Interacting With Design Alternatives

by

Siniša Kolarić
M.Sc., Pontifical Catholic University of Rio de Janeiro, Department of Informatics, 2008
B.Sc., University of Zagreb, Department of Mathematics, 1998

Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Doctor of Philosophy

in the
School of Interactive Arts and Technology
Faculty of Communication, Art, and Technology

© Siniša Kolarić 2016
SIMON FRASER UNIVERSITY
Summer 2016

All rights reserved.
However, in accordance with the Copyright Act of Canada, this work may be reproduced without authorization under the conditions for "Fair Dealing." Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.
APPROVAL

Name: Siniša Kolarić

Degree: Doctor of Philosophy

Title of Thesis: Interacting With Design Alternatives

Examining Committee: Tom Calvert, Professor Emeritus
SIAT, Simon Fraser University, Canada
Chair

Dr. Halil Erhan, Associate Professor
SIAT, Simon Fraser University, Canada
Senior Supervisor

Dr. Robert Woodbury, University Professor
SIAT, Simon Fraser University, Canada
Co-Supervisor

Dr. Carman Neustaedter, Associate Professor
SIAT, Simon Fraser University, Canada
Internal Examiner

Dr. Reinhard König, Junior-Professor
Bauhaus University at Weimar, Germany
External Examiner

Date Approved: May 26th, 2016
The author, whose name appears on the title page of this work, has obtained, for the research described in this work, either:

a. human research ethics approval from the Simon Fraser University Office of Research Ethics

or

b. advance approval of the animal care protocol from the University Animal Care Committee of Simon Fraser University

or has conducted the research

c. as a co-investigator, collaborator, or research assistant in a research project approved in advance.

A copy of the approval letter has been filed with the Theses Office of the University Library at the time of submission of this thesis or project.

The original application for approval and letter of approval are filed with the relevant offices. Inquiries may be directed to those authorities.

Simon Fraser University Library
Burnaby, British Columbia, Canada

Update Spring 2016
Abstract

Many designers tend to work with multiple, simultaneously-available design alternatives. The evidence for this fact can be found in many different domains of design, such as architectural, product, industrial and mechanical design, but also in drawing, painting, sculpting, and fine arts in general. Numerous empirical studies, likewise, report instances of designers developing, and then simultaneously working with multiple design solutions in parallel. The need for alternatives-enabled work can be further confirmed by instances of expert opinion, as well as explained by theoretical accounts based on first perceptual principles, such as those governing the human visual system, or human perception in general.

Yet, despite the available evidence confirming the importance of multi-state work, most contemporary computational tools, including computer-aided design (CAD) tools, their important subvariant (parametric CAD, or pCAD tools), as well as other types of computational tools, are steeply immersed in the single-state paradigm, whereby the user can work with, and modify, just a single computational model at a time. Consequently, since the overwhelming majority of users nowadays still employ single-state tools, they necessarily have to recur to various ad hoc methods when engaging in multi-state work, in order to circumvent the limitations inherent to single-state tools. However, such workaround methods are inefficient, and error-prone.

The aim of this work is to address the aforementioned shortcoming of single-state computational tools, while focusing on pCAD tools. The thesis of this work, accordingly, is that “alternatives-enabled (i.e. multi-state) pCAD tools, designed in concordance with the interaction design guidelines and principles presented in this dissertation, support designing”. I break down my thesis statement further into the following three main research questions, each addressing one separate aspect of the statement: (1) What are the essential system features of novel multi-state pCAD tools that can support design work? (2) What are the design guidelines and principles for building and evaluating multi-state pCAD tools? and (3) What methodological approach can be followed that can help system designers build multi-state pCAD tools?

In order to support my thesis, I first conceptualize (or design, develop) the class of multi-state pCAD tools, by first employing both (1) various pre-design methods (such as literature review, and probing of the “design space” of feasible interaction designs), as well as (2) by employing various
research-through-design methods characterized by the development of numerous models, interactive research prototypes, and design artifacts in general. All these activities provide evidence about the effectiveness of specific, individual features of multi-state pCAD tools. The two main research methods that I use include (a) an expert study whereby six experts voice their opinion on the CAMBRIA 1 prototype and its features, and (b) a cognitive analysis which provides scores of the CAMBRIA 2 prototype evaluated along a number of “cognitive dimensions”. The experience that I acquire by performing pre-design and research-through-design activities allows me to synthesize high-level bodies of knowledge such as design guidelines and principles, as well as to describe the method that I followed in order to conceptualize multi-state pCAD tools. These, in turn, answer postulated research questions, and thus support the thesis.

Siniša Kolarić
Spring 2016.
To my family and friends.
“Beauty will result from the form and the correspondence of the whole, with respect to the several parts, of the parts with regard to each other, and of these again to the whole; that the structure may appear an entire and complete body, wherein each member agrees with the other, and all necessary to compose what you intend to form.”

— Andrea Palladio (1508–1580)
Acknowledgements

Many thanks to my doctoral advisors Dr. Halil Erhan and Dr. Robert Woodbury for introducing me to the topic of multi-state parametric CAD tools, and for their continuous academic and financial support over the years. Likewise, many thanks to my examiners Dr. Carman Neustaedter and Dr. Reinhard König to take the time to read my thesis, and provide valuable feedback.

My research was generously funded by the Canada’s *Networks of Centres of Excellence* (NCE) program through GRAND (Graphics, Animation and New Media) NCE, within the ENCAD (Enabling Technologies for CAD Systems) project. The Faculty of Communication, Art and Technology (FCAT) as well as the School of Interactive Arts and Technology (SIAT) supplemented this support with matching postgraduate scholarships. Special thanks to Brad Holtz, president & CEO of Cyon Research, who made my participation at COFES 2013 possible, and who provided valuable expert feedback on CAMBRIA.

The School of Interactive Arts and Technology (SIAT) at Simon Fraser University provided technical and administrative support that made my learning and research possible. All of this support is gratefully acknowledged.
Contents

Approval ii
Ethics Statement iii
Abstract iv
Dedication vi
Quotation vii
Acknowledgements viii
Contents ix
List of Tables xvii
List of Figures xviii

1 Introduction 1
  1.1 Why Work With Many Designs at Once? ......................... 1
  1.2 Research Overview ............................................. 5
    1.2.1 Situation of Concern ................................... 5
    1.2.2 Research Problem Statement ............................ 5
    1.2.3 Research Intent Statement ............................... 6
    1.2.4 Research Contributions ................................. 6
    1.2.5 Thesis Statement ....................................... 7
  1.3 Dissertation Outline ....................................... 8
    1.3.1 Part I: Preliminaries .................................... 8
    1.3.2 Part II: Background and Related Work .................. 8
    1.3.3 Part III: Stages of Research ............................. 8
    1.3.4 Part IV: In Closing ..................................... 9
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.2.1 Research in HCI Design: A Knowledge-Centric View</td>
<td>46</td>
</tr>
<tr>
<td>3.2.2.2 Research in HCI Design: An Activity-Centric View</td>
<td>49</td>
</tr>
<tr>
<td>3.2.3 Evaluating HCI Designs</td>
<td>55</td>
</tr>
<tr>
<td>3.2.3.1 HCI Evaluation Methods</td>
<td>56</td>
</tr>
<tr>
<td>3.2.4 Stopping Point: When is an HCI Research Problem Considered ‘Solved’?</td>
<td>58</td>
</tr>
<tr>
<td>3.3 Research Approach Adopted for This Thesis</td>
<td>60</td>
</tr>
<tr>
<td>3.3.1 Phases of Research Performed</td>
<td>62</td>
</tr>
<tr>
<td>3.4 Chapter Summary</td>
<td>64</td>
</tr>
<tr>
<td>4 Background: Many Design Alternatives at Once</td>
<td>66</td>
</tr>
<tr>
<td>4.1 Evidence for Concurrency in Design</td>
<td>67</td>
</tr>
<tr>
<td>4.1.1 Alternatives in Architectural Design</td>
<td>68</td>
</tr>
<tr>
<td>4.1.2 Alternatives in Product Design</td>
<td>69</td>
</tr>
<tr>
<td>4.1.3 Alternatives in Fine Arts</td>
<td>70</td>
</tr>
<tr>
<td>4.1.4 Alternatives in Industrial Design</td>
<td>71</td>
</tr>
<tr>
<td>4.1.5 Evidence from Empirical Studies</td>
<td>72</td>
</tr>
<tr>
<td>4.1.6 Expert Opinion</td>
<td>74</td>
</tr>
<tr>
<td>4.1.7 Theoretical Justification for Multiple Design Alternatives</td>
<td>76</td>
</tr>
<tr>
<td>4.1.8 Ambiguity in Design</td>
<td>77</td>
</tr>
<tr>
<td>4.2 Fixation on a Single Design Solution</td>
<td>78</td>
</tr>
<tr>
<td>4.3 Discussion</td>
<td>79</td>
</tr>
<tr>
<td>4.3.1 A Way to Conceptualize Design Multiplicity</td>
<td>81</td>
</tr>
<tr>
<td>4.4 Chapter Summary</td>
<td>82</td>
</tr>
<tr>
<td>5 Related Work</td>
<td>83</td>
</tr>
<tr>
<td>5.1 Interactive Visualization in General</td>
<td>84</td>
</tr>
<tr>
<td>5.1.1 Taxonomies of Interaction Tasks and Techniques</td>
<td>85</td>
</tr>
<tr>
<td>5.1.1.1 Interaction Typology by Brehmer and Munzner (2013)</td>
<td>87</td>
</tr>
<tr>
<td>5.1.1.2 Interaction Taxonomy by R. E. Roth (2013)</td>
<td>89</td>
</tr>
<tr>
<td>5.2 Multi-State Interactive Visualization</td>
<td>92</td>
</tr>
<tr>
<td>5.2.1 Parallel Visualization</td>
<td>93</td>
</tr>
<tr>
<td>5.2.2 Subjunctive Interfaces (A. Lunzer)</td>
<td>94</td>
</tr>
<tr>
<td>5.2.3 Set-Based Interaction (M. Terry)</td>
<td>99</td>
</tr>
<tr>
<td>5.2.4 Parallel Text Editing</td>
<td>100</td>
</tr>
<tr>
<td>5.2.5 Other Reported Approaches to Multi-Interaction</td>
<td>102</td>
</tr>
<tr>
<td>5.2.6 Commercial Tools and Systems</td>
<td>103</td>
</tr>
<tr>
<td>5.3 Histories and Undo/Redo</td>
<td>107</td>
</tr>
</tbody>
</table>
6 Phase A: Pre-Design

6.1 Domain Vocabulary

6.2 Pre-Design Activities

6.2.1 Cataloguing Candidate Approaches Through Literature Review

6.2.1.1 Interaction Paradigms and Interaction Styles

6.2.1.2 Interaction Techniques

6.2.2 GUI Sketches and Mockups

6.2.3 Throwaway Interactive Prototypes

6.2.3.1 Prototype #1: “2 × 2 Parametric Designer”

6.2.3.2 Prototype #2: “2D Font Designer”

6.2.3.3 Prototype #3: “Subjunctive 3D Points”

6.2.3.4 Prototype #4: “Taming Combinatorial Explosion”

6.2.4 Discussion

6.3 Updated Design Requirements

6.4 Chapter Summary

7 Phase B: Conceptual Design

7.1 Metaphors Chosen

7.1.1 Metaphor #1: Design Gallery

7.1.2 Metaphor #2: Sketching and Tracing Paper

7.1.3 Metaphor #3: Evolutionary Tree

7.2 Main Concepts and Their Relationships

7.2.1 Variables

7.2.2 Items

7.2.3 Constraints

7.2.3.1 One-Way Constraints

7.2.3.2 Two-Way Constraints

7.2.3.3 Constraints Constrain Items

7.2.3.4 Constraint Solver

7.2.4 Design Model = Items + Constraints

7.2.5 Design Model Instantiation

7.2.6 Design Model Modification

7.2.7 Design History of a Model

7.2.8 Set of Alternatives for a Model

7.3 Chapter Summary
8 Phase C: CAMBRIA 1 Prototype

8.1 Design of CAMBRIA 1

8.1.1 Collections View: Managing Collections of Alternatives

8.1.2 Alternatives View: Selecting and Managing Alternatives

8.1.3 Parallel Editing View: Modifying Selected Design Alternatives

8.2 Evaluating CAMBRIA 1: Expert Review

8.2.1 Feedback by Theoretical Concepts

8.2.1.1 Alternatives, and Their Structure

8.2.1.2 Duplicating Alternatives

8.2.1.3 Collections of Alternatives

8.2.1.4 Sequential Single Alternative Editing

8.2.1.5 Editing of Many Alternatives Simultaneously

8.2.1.6 Moving Alternatives Among Collections

8.2.1.7 “Pass” and “Link” Interaction Features

8.2.1.8 See and Compare Many Alternatives at Once

8.2.1.9 3D

8.2.1.10 Collaboration

8.2.2 Feedback: Post-Interview Questions

8.2.2.1 What Are The Most Interesting Features of This Prototype?

8.2.2.2 What Are The Least Interesting Features of This Prototype?

8.2.2.3 Envision This in a Much More Developed and Polished Version. How Would You Work With It in Your Practice?

8.2.2.4 What Barriers Does It Present?

8.2.2.5 How Would You Change It?

8.2.3 Other Feedback

8.2.3.1 Confusing, Non-Intuitive and Unsatisfactory Features

8.2.3.2 Intriguing Remarks

8.3 Discussion

8.4 Chapter Summary

9 Phase D: CAMBRIA 2 Prototype

9.1 Main Changes from CAMBRIA 1

9.2 Design of CAMBRIA 2

9.2.1 Managing Alternatives and Related Constructs

9.2.2 Arranging Alternatives for Viewing

9.2.3 Navigating Alternatives

9.2.4 Editing Alternatives
9.3 Evaluation of CAMBRIA 2 ......................................................... 194
  9.3.1 Visibility ................................................................. 195
  9.3.2 Juxtaposability ......................................................... 197
  9.3.3 Hidden Dependencies .................................................. 197
  9.3.4 Viscosity ............................................................... 199
  9.3.5 Terseness .............................................................. 200
  9.3.6 Proneness to Error .................................................... 201
  9.3.7 Hardness of Mental Operations ...................................... 202
  9.3.8 Abstraction Gradient .................................................. 204
  9.3.9 Premature Commitment .............................................. 205
  9.3.10 Progressive Evaluation .............................................. 206
  9.3.11 Secondary Notation .................................................. 206
  9.3.12 Custom Dimension: “Alignment to Chosen Metaphors” ....... 207
9.4 Discussion ............................................................................. 210
9.5 Chapter Summary ............................................................... 215

10 Phase E: Consolidation ......................................................... 216
  10.1 Classifying Interaction Primitives for Multi-State Systems .... 217
    10.1.1 High-Level Tasks ..................................................... 218
    10.1.2 Interaction Methods .................................................. 222
  10.2 Design Guidelines for Alternatives-Enabled pCAD Tools .... 228
    10.2.1 CHOOSE Methods ..................................................... 232
      10.2.1.1 Search (CHOOSE-SEARCH) .................................. 232
      10.2.1.2 Preview (CHOOSE-PREVIEW) ............................ 234
      10.2.1.3 Select (CHOOSE-SELECT) ............................... 236
    10.2.2 EXPRESS Methods ................................................... 237
    10.2.3 ARRANGE Methods .................................................. 240
      10.2.3.1 Automatic ARRANGE Methods ............................... 242
      10.2.3.2 Manual ARRANGE Methods ................................. 250
    10.2.4 NAVIGATE Methods .................................................. 251
      10.2.4.1 Panning ............................................................ 251
      10.2.4.2 Zooming .......................................................... 253
      10.2.4.3 Overview+Detail ................................................ 254
      10.2.4.4 Focus+Context ..................................................... 256
    10.2.5 EXAMINE Methods .................................................... 257
      10.2.5.1 Details-on-demand ............................................. 257
      10.2.5.2 Filter-in and -out .............................................. 258
### Appendix A: Interviews

<table>
<thead>
<tr>
<th>Participant</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>320</td>
</tr>
<tr>
<td>P2</td>
<td>330</td>
</tr>
<tr>
<td>P3</td>
<td>343</td>
</tr>
<tr>
<td>P4</td>
<td>367</td>
</tr>
<tr>
<td>P5</td>
<td>378</td>
</tr>
<tr>
<td>P6</td>
<td>390</td>
</tr>
</tbody>
</table>

### Appendix B: A Formalism for Subjunctive pCAD Models

409

### Appendix C: Glossary

414

### Bibliography

420
List of Tables

2.1 Various definitions of the activity of designing, as found in the literature. ........................ 11
2.2 Comparing well- and ill-defined problems. Table adapted from [146]. .......................... 19

3.1 Comparison of conventional research, and research in arts and design, according to Frayling [121]. ............................................................................................................. 42
3.2 Juxtaposition of types of research in arts and design [121, 113]. ................................. 44

8.1 Experts who participated in the study. .............................................................................. 153
8.2 Most interesting features of CAMBRIA 1. ................................................................. 162
8.3 Envisioned uses for a CAMBRIA-like system. ............................................................ 163
8.4 Changes suggested for CAMBRIA 1. ............................................................................ 166
8.5 Expert feedback per major functionality of CAMBRIA 1. ............................................ 169
8.6 Feedback on CAMBRIA 1: features desired by experts. ............................................. 173

9.1 Evaluating cognitive dimension of CAMBRIA 2: Summary table. ............................. 210

11.1 Learning preferences and inclinations [195, 115]. ....................................................... 305
# List of Figures

1.1 A schematic depiction of conventional (single-state) vs. alternatives-enabled (multi-state) software tools. In the conventional approach, an user interacts with one computational model at all times. In the alternatives-enabled approach, an user interacts with, and manages, multiple model alternatives in parallel.  

1.2 Multiple design alternatives in product design, arranged in a rectilinear grid. Drawings of earbuds © Patrick Murphy [259]. Reused under CC BY 2.0 license.  

1.3 Multiple, cognate “massing” models for a building designed by architect Frank Gehry, and built in New York City in 2011. Reproduced by permission © Vitra Design Museum. Photo credit: Bettina Matthiesen [242].  

1.4 Microsoft Surface Hub, an 84”, wall-mounted, touch-enabled display. In this depiction, two designers collaborate on a chair design, with four alternative chair designs shown at the top of the screen. Image credit: Microsoft Corporation promotional materials [249].  

1.5 Research overview.  

2.1 Attributes (dimensions) of representations [391, 50, 271].  

2.2 The “abstract ↔ specific” continuum for visual-spatial representations according to Fish and Scrivener [118].  

2.3 An instance of problem solving by searching in a “problem space”.  

2.4 High-level functional design requirements for a patient information system, subdivided into two functional areas. Example based on [268, p. 80].  

2.5 When developing design requirements, the designer moves in a three-dimensional space, from the initial idea (which is mostly opaque, informal, and lacks agreement), to the desired output (i.e., design requirements) which maximizes the three principal dimensions (completeness of specification, formality of representation, and full degree of stakeholders’ agreement). Based on [288].  

2.6 An early GUI sketch.  

2.7 A design problem and any of its solutions should be regarded as an unit.
2.8 Co-evolving design problems and design solutions using genetic algorithms. Based on Figure 1 in [233]. .............................................................. 29

2.9 Action cycle mediated through a cognitive artifact. Note that in this scenario actions are exerted through the artifact, and that the designer perceives the world indirectly, through a representation generated by the artifact. Based on [273, 271]. 31

2.10 A checklist. .............................................................. 33

3.1 Research methods, usually aggregated into research approaches or methodologies, are the means to augment existing (or create new) bodies of knowledge. . . . 35

3.2 Ackoff’s “data-information-knowledge-wisdom” (DIKW) model of knowledge [3]. . . 38

3.3 Any instance of interaction design research lies on a point in the triangle defined by: professional practice, scientific studies, and playful exploration. Adapted from [113, 114]. .............................................................. 43

3.4 A schematic visualization of a design, or a “design state”, marked by the identifier “1”. .............................................................. 44

3.5 The knowledge-centric view evolves from a design state, and from all the knowledge representations constituting the state, now understood as a “design ensemble”. Further, design ensemble representations are augmented by both the lower-level research artifacts (for instance, the domain vocabulary), as well as higher-level research artifacts (such as e.g., design guidelines) that are derived by the HCI researcher through the knowledge accumulated by repeated co-evolution of the ensemble. .............................................................. 45

3.6 The activity-centric view evolves from a design state, and from the activities (such as design, build, and evaluate) needed to create and understand that state. The total of all activities, iterated and repeated over time, ultimately results in the rich structure of design history shown at the right end of the diagram. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46

3.7 Every design artifact opens up its own, associated design sub-space. Many such sub-spaces can, when unified, lead to a larger design space [357]. In this schematic example, design artifacts $a_0, a_1, \ldots, a_8$ opened up their own design subspaces $D_0, D_1, \ldots, D_8$ which, when unified, may lead to a larger design space $D = D_0 \cup D_1 \cup \ldots \cup D_8$. .............................................................. 46
Some bodies of knowledge (i.e., research artifacts) developed in the course of solving an IxD/HCI research problem. The design problem representation (i.e., the requirements document) and design solution representations co-evolve as an ensemble, and build upon the concepts found in the vocabulary. The co-evolution of the ensemble leads to the designer's deep understanding of the target research problem, allowing him to synthesize “design guidelines”, or “principles”, shown at the top, as well as to fully describe the custom design development followed.

In design, artifacts are developed iteratively, over a period of time. The global design development process periodically releases revisions of design artifacts.

In research through design, design development activities are interwoven with design research activities. In other words, we research as we simultaneously develop design solutions, and other research artifacts. Correspondingly, design artifacts are now also seen as a kind of “embodied knowledge” [357, 121].

An iterative model of HCI designing. Based on [328].

A simplified iterative model of HCI designing.

Left: long HCI evaluation cycle, featuring a formal evaluation study. Right: short (rapid) HCI evaluation cycle, in order to quickly obtain design feedback for the purposes of learning (unused parts of the large cycle greyed out for easier comparison).

The design development activities of forward iteration, backtracking, and merging result in a structure that can be represented by a directed acyclic graph (DAG). Each node in this structure is actually a composite one, consisting approximately of the design-build-evaluate (DBE) activities (cf. Fig. 3.15). Furthermore, some nodes (indicated by black colour) feature a long evaluation test (cf. Fig. 3.13, left), while all the other nodes feature a rapid evaluation test (cf. Fig. 3.13, right).

Two ways to further abstract and visualize the model of iterations in HCI design research. Left: the “activity-centric” sequence of DBE iterations, where each B activity (Build) ends in the creation of a design solution artifact (other research artifacts not shown for simplicity). Right: the sequence of design states produced through DBE iterations.

Evaluation methods help the HCI researcher to choose among two or more design solution alternatives for a given design design requirement. In this example, design proposal C wins over design proposals A and B, because the evaluation method has indicated that the design proposal C is more suitable in some way for the design requirement in question, than either design proposals A and B.

An overview of the research described in this dissertation, divided into five consecutive phases.
3.18 The “double-diamond” model of design as proposed by the British Design Council. Based on [51]. 63

4.1 Example of multiple design alternatives used “in the wild” in architectural design: “Elephant House” by Brady Peters / Foster + Partners. Source: Brady Peters [48]. 68

4.2 “Elephant House” by Brady Peters / Foster + Partners. Source: Brady Peters [48]. 69

4.3 Example of multiple design alternatives used in product design: (a) Design alternatives developed (hand-drawn sketches), and (b) “Juicy Salif” citrus squeezer. 70

4.4 Examples of alternatives “in the wild” in the field of visual arts: “Study for the Trivulzio Equestrian Monument” [213] by Leonardo da Vinci. Different parts of this sketch feature multiple alternative outlines, with different positions and orientations. I isolated three parts of the sketch (see the cutouts on the left side of the figure, from top to bottom): (1) different rider’s head positions, (2) multiple positions of the horse’s hind legs, and (3) multiple positions of the horse’s hind legs. 71

4.5 Point-based approach to car design. Based on [396]. 71

4.6 Set-based, concurrent approach to car design is based on the gradual narrowing of entire sets of possible (i.e., alternative, potential) design solutions. Based on [396]. 72

4.7 Having multiple design alternatives in the field of view facilitates their comparison and recombination. The retrieve-recombine-store process (left), where the designer retrieves alternatives one by one and is forced to remember the details of previous alternatives in the retrieve-store sequence, gets replaced by the faster scan-recombine process (right) in which the designer can readily pinpoint, select, and evaluate individual parts, and recombine them into a new alternative state. 77

4.8 Design multiplicity graphs for two (hypothetical) designers, are in general different for the same design project. 81

5.1 Seven fundamental interaction tasks by Shneiderman and Plaisant [333, p. 539]. 85

5.2 Taxonomy of interaction tasks by Amar et al [16]. 86

5.3 Taxonomy of interaction tasks by Gotz and Zhou [135]. 86

5.4 “Why will this task be performed?”; according to the typology by Brehmer and Munzner [49]. 87

5.5 “How will this task be performed?”; according to the typology by Brehmer and Munzner [49]. 88

5.6 Linked views showing a “shape space” (top left), “object space” (top right), and “design history” (bottom). Figure created after [56]. 94
5.7 In the coding editor *Sublime Text*, multiple cursors (in this case, three) allow the user to edit different parts of the textual file at the same time. Left: three cursors inserted into the document. Right: one character deleted to the left of each cursor, after the user presses the Backspace or Delete key. ........................................ 102

5.8 Previews in MS Office Word 365. ................................................................. 104

5.9 Viewing two visualizations in parallel in Autodesk ShowCase. Image credit [106]. . 104

5.10 Three different “configurations” of a SolidWorks model. Image credit [350]. . . . 105

5.11 The 3DVIA commercial tool by Dassault Systèmes. .................................................. 106

5.12 Alternatives in video gaming. In this example, the user picked three different options for the part of the avatar’s armour protecting torso. Images credit [386]. . . . 106

5.13 Visually comparing two 3D wireframe models, each within its own (independent) design history timeline. Image credit [75]. ....................................................... 107

6.1 Diagram of the main concepts. ................................................................. 112

6.2 Initial (seed) design requirements on alternatives-enabled tools. ......................... 113

6.3 Multiple alternatives arranged as a sequence in a rectilinear grid, with two different expression types for alternatives (thumbnails and glyphs). ......................... 114

6.4 Early-stage interaction sketches. In (a), parallel editing of analogous parts for two different yet related designs: a bicycle design, and a wheelbarrow design. In (b), a number of different bicycle design alternatives, as well as ways to pass analogous bicycle parts from one design to another. ......................................................... 115

6.5 Early-stage storyboard (hand sketching + raster image editing), frames 1–24. . . . 117

6.6 (Continued from Fig. 6.5.) Early-stage storyboard (hand sketching + raster image editing), frames 25–36. .............................................................. 118

6.7 Early wireframe mockups of graphical user interfaces. (a) Explorer view (latest versions), (b) Explorer view (version history), (c) Parallel editor (juxtaposition mode), and (d) Parallel editor (superposition mode). ....................................... 119

6.8 A sample of rapidly-developed, low-fidelity prototypes. (Listed in the chronological order of development). .............................................................. 120

6.9 An early, throwaway prototype nick-named “2 × 2 parametric designer”, featuring: (1) multiple views (one per design alternative), which can be active or inactive (as indicated by the toggle button in top left corner), (2) items linked through unique identifiers, (3) brushing (mouse-over) of all the linked instances (as shown in the case of the curve, highlighted in bright red), and (4) selection of multiple graphical items in parallel (the curve selected in all three active views, indicated by dashed line contour). ................................................................. 121

6.10 Interactive prototype for 2D parametric design (“Font Designer”). ......................... 123
6.11 Interactive prototype for 3D modelling (“Subjunctive 3D Points”).
6.12 Interactive prototype based on Zwicky’s “morphological analysis” [421, 422, 305].
6.13 Updated design requirements.

7.1 Metaphor #1: Design gallery. Reproduced by permission © Vitra Design Museum. Photo: Bettina Matthiesen [242].
7.2 Metaphor #2: Tracing paper with a drawing, on top of another drawing. Image credit [301].
7.3 An early evolutionary tree created by E. Häckel in 1866 [147].
7.4 An abstracted visualization of an item.
7.5 Simple item types, along with their axis-aligned minimum bounding boxes (BB): line, curve, rectangle, text.
7.6 Items can be composite i.e., modular. In this example, item 3 comprises three sub-items 3.1, 3.2, and 3.3, which can be either simple or composite.
7.7 One-way (unidirectional) constraints are indicated by one arrowhead.
7.8 A directed acyclic graph (DAG) is a directed graph with no directed cycles. In this example, adding an one-way dataflow constraint from item 2 to item 6 would create a cycle (infinite loop) $2 \rightarrow 6 \rightarrow 4 \rightarrow 2 \rightarrow 6 \rightarrow \ldots$ and this graph would therefore cease to be a DAG.
7.9 Bidirectional constraints are indicated either by two arrows, or no arrow.
7.10 The behaviour and geometric configuration of items is governed by constraints (here depicted as red dotted lines).
7.11 An abstraction of a design model consisting of five items.
7.12 An abstract design model's history can be represented as an acyclic directed graph, whose nodes (i.e., design states) are further distinguished by developmental branches, merge events, and revision numbers.
7.13 Three sets A, B, C of alternatives, for a design model. For instance, set A contains alternatives 0, 1, 2, 6, 7, and 9.

8.1 Screenshot of CAMBRIA 1 prototype, with its three main constituent parts: Collections view (left), Alternatives view (middle), and the Parallel Editing view (right).
8.2 Collections view.
8.3 “Alternatives” view in CAMBRIA 1. In the mode shown ("manual arrangement" mode), the user can select multiple alternatives (i.e., thumbnails thereof) at once, zoom in and out of them, and arrange them into proximity-based clusters.
8.4 Ways to arrange alternatives for viewing in CAMBRIA 1: (a) “Thumbnails” (grid of thumbnails), (b) “Data Grid”, and (c) “Manual Arrangement” view.
8.5 Two ways of presenting alternatives in the parallel editing area of CAMBRIA 1:
(a) juxtaposition, and (b) superposition.

8.6 Linking and de-linking of alternatives in CAMBRIA 1 occurs by toggling the check-
box at the top of the view. Here, both A-3 and A-4 are linked.

8.7 Parallel selection in CAMBRIA 1.

8.8 “Passing variable” mechanism in CAMBRIA 1.

8.9 “Passing variable value” mechanism in CAMBRIA 1.

8.10 In CAMBRIA 1, each design alternative features a simple “editing toolbar” at its top.

8.11 Parallel editing in CAMBRIA 1.

8.12 Coding of an interview transcript.

8.13 Collections view (outlined in red).

8.14 Clustering results in two new entities, or “gestalts”.

9.1 CAMBRIA 2 screenshot, showing multiple bicycle designs derived from each other.

9.2 Schematic depiction of the hierarchy of constructs in CAMBRIA 2.

9.3 File menu in CAMBRIA 2.

9.4 Collections are implemented as “tabs” that can be selected at the bottom of the
window.

9.5 Arrange menu with options for presenting a set of alternatives in CAMBRIA 2.

9.6 Arrangement modes available in CAMBRIA 2.

9.7 Panning in parallel across multiple alternatives in CAMBRIA 2, using mouse press
and drag, with the key Alt depressed.

9.8 Multi-state (interactive) zooming in parallel in CAMBRIA 2.

9.9 Edit menu in CAMBRIA 2.

9.10 Parallel (item-by-item) selection in CAMBRIA 2.

9.11 Rubberband selection of items in CAMBRIA 2 (BEFORE).

9.12 Rubberband selection of items in CAMBRIA 2 (AFTER).

9.13 Contextual menus in CAMBRIA 2.

9.14 Two point types in CAMBRIA 2.

9.15 Various graphical items than can be created in CAMBRIA 2.

9.16 Various constraints available for a free point in CAMBRIA 2.

9.17 CAMBRIA 2 screenshot, with alternatives’ undo views shown.

9.18 Pass items mechanism in CAMBRIA 2.

9.19 Pass property (in this case, pen colour) mechanism in CAMBRIA 2.

9.20 Point snapping in parallel across multiple alternatives in CAMBRIA 2.

9.21 Point snapping in parallel across multiple alternatives in CAMBRIA 2.
9.22 A way to increase visibility of constraints in CAMBRIA 2, by superposing constraint “boxes” (here coded in red) connecting input to output items/variables, whenever an item is selected in the alternative. .............................................. 196
9.23 A mockup of the interaction functionality for detecting linkage among instances of item #6. ................................................................. 198
9.24 Visual summary of interaction levels, as the user drills in and out of data structures: (1) projects, (2) collections within the open project, (3) view and select multiple alternatives within multiple collections, (4) edit selected alternatives, and (5) working with selected items within alternatives. Of course, the user should be able to switch levels easily and rapidly. .............................................. 214

10.1 A schematic model of high-level tasks synthesized from Norman’s, Brehmer’s, and Schönh’s models, with three highest-level tasks (Perceive, Reflect, and Act) highlighted in light blue. .......................................................... 219
10.2 Different motivations for interacting with a computational tool. ................................................................. 220
10.3 Classification of high-level tasks for working with multi-state pCAD tools. .................................................. 222
10.4 Classification of interaction methods for multi-state pCAD tools. ............................................................. 228
10.5 Design guidelines for attracting user’s attention [403][333, p. 60]. .............................................................. 229
10.6 Summary of design guidelines for multi-state pCAD systems. ................................................................. 231
10.7 An example of “parallel” or “subjunctive” search functionality. ............................................................ 233
10.8 Interactive prototype for 2D parametric design (“Font Designer”), with 15 preview states. ............................................................. 235
10.9 A mockup of preview functionality based on effects modifying multiple selected items. ............................................................. 236
10.10 Multi-state selection of multiple (linked) items in CAMBRIA 2. ................................................................. 237
10.11 Expressing a collection graphically. ........................................................................................................... 238
10.12 Expressing an alternative (purely) graphically. .......................................................................................... 239
10.13 Expressing an alternative alphanumerically. ............................................................................................ 239
10.14 Some ways to express an alternative using a hybrid method. ............................................................... 239
10.15 A mockup of eight different ways to arrange design alternatives. ......................................................... 241
10.16 Multiple alternatives arranged as a sequence in a rectilinear grid, with two different expression types for alternatives (thumbnails and glyphs). ............................................................. 243
10.17 Multiple alternatives arranged into (optionally transparent) layers. ....................................................... 244
10.18 Superposing multiple alternatives on top of each other in CAMBRIA 1. .................................................. 244
10.19 Mockup of multiple alternatives arranged as a slideshow. .................................................................. 245
10.20 Multiple alternatives arranged as a sequence of rows displaying alphanumeric data. .......................... 246
10.21 A mockup of “lifestream” or “coverflow” arrangement. ........................................................................ 247
10.22 Multiple design alternatives arranged automatically by applying a distance metric function. ................................................................. 248
10.23 A mockup of multiple design alternatives arranged according to their lineage. .... 249
10.24 “Design history” arrangement mode available in CAMBRIA 2. .............................. 249
10.25 A mockup of multiple alternatives arranged manually. ...................................... 251
10.26 Multi-state (interactive) panning in parallel in CAMBRIA 2. ............................... 252
10.27 Multi-state (automatic) panning “to an item” in parallel in CAMBRIA 2. ................. 253
10.28 Multi-state (interactive) zooming in parallel in CAMBRIA 2. ............................ 254
10.29 A mockup of a strip of thumbnails for overview+detail navigation of multiple design alternatives within a collection. .............................. 255
10.30 A mockup of an inset overview pane for overview+detail navigation of an alternative. 256
10.31 A mockup of multi-state focus+context navigation functionality, based on a combination of “subjunctive cursors”, and fisheye lenses. ...................... 257
10.32 A mockup of multi-state focus+context navigation functionality, based on a fisheye lens applied to a sequence of rows displaying alphanumeric data. ............... 257
10.33 Details-on-demand: tooltip (on mouse-over) and “detail” pane (populated with data from the selected object). ........................................ 258
10.34 Filter out items within active alternatives. Note that alternative #2 is inactive, therefore filtering does not apply to it i.e., items 1, 2, 4, 6 are not suppressed. .......... 259
10.35 Highlight items within alternatives. ................................................................. 260
10.36 Brushing of linked (associated) items in CAMBRIA 2 using colour coding. (All four rear wheel instances rendered in red.) .................................. 261
10.37 A mockup of the Spot-the-Difference mode (based on local focus) for three design alternatives. Here, item 6 matched and highlighted in all three design alternatives. 262
10.38 Differencing (mode 2, colour-coded) for three alternatives, using “parallel coordinates” connecting centers of bounding boxes for items 1, 2, and 3. By observing slopes of lines connecting parallel coordinates, the user can easily perceive differences in placement along the vertical axis. ................................. 262
10.39 Differencing three alternatives using superposition (layers). Here, items 1 and 4 highlighted in green and red, respectively, indicating their differences in placement, size, and shape. .................................................. 263
10.40 View decoration and management. .................................................................... 265
10.41 Solo mode activated for an alternative, as indicated by the depressed (red) “S” button. All the remaining alternatives automatically enter into “mute” mode (blue “M” button), thus not receiving any changes. ................................. 265
10.42 “Viewlets” within an alternative’s view. In this example, alternative #3 has three viewlets opened. ............................................................... 266

xxvi
10.43 An illustrative depiction of some important ways to generate new objects within multi-state tools. ................................................................. 269
10.44 “Pass items” mechanism in CAMBRIA 2. ................................................................. 271
10.45 “Pass property” (in this case, pen colour) mechanism in CAMBRIA 2. ............. 272
10.46 A simple shape rule. ................................................................. 273
10.47 Substituting a component of the model (torso armour) in-place, with three different component-level alternatives. Images credit [386]. .................................................................................. 274
10.48 A mockup of substitute-part functionality, based on pop-up “coverflow” lists of alternatives for a part. ................................................................. 275
10.49 Point snapping in parallel across multiple alternatives in CAMBRIA 2. ........... 276
10.50 Interface design principles (“Golden Rules”) by Shneiderman and Plaisant [333, p. 70]. .................................................................................. 280
10.51 System design principles for tools that support “creative” problem solving [175]. .................................................................................. 281
10.52 Seven “lessons” for computational tools supporting multiple “ideas” by Smith et al [345]. .................................................................................. 281
10.53 Ten “usability heuristics” for user interface design by Nielsen [269]. .............. 282
10.54 High-level design principles for alternatives-enabled pCAD tools, as proposed in this thesis. .................................................................................. 282
10.55 CAMBRIA 2 running on a 3 × 2 multi-monitor setup, showing a collection of boat designs. .................................................................................. 288

11.1 Student’s IxD before-and-after expertise developed during doctoral studies. ................................................................. 292
11.2 The three-level hierarchical model of design knowledge: a methodological “mental map”. .................................................................................. 292
11.3 Bodies of knowledge generated in this work. ................................................................. 293
11.4 Five major consecutive phases of the research described in this dissertation, with pragmatic alternation between phases. .................................................................................. 296
11.5 Designer-researcher “traverses” design states in the IxD design space (a), thus generating design history (b). .................................................................................. 297
11.6 Pragmatic alternating of the activities of updating various bodies of design knowledge. .................................................................................. 297
11.7 A visual metaphor depicting a progressively “clearer” (i.e., better structured, understood, and comprehended) body of knowledge, closely following K. Weick [401]. Note that both the structure as well as the details of this clockwork become progressively more distinguished and discernible. .................................................................................. 299
11.8 Progressive build-up and clarification (“sharpening of focus”) of hierarchical bodies of IxD knowledge developed in this thesis. .................................................................................. 299
11.9 Cognitive processes within the Schöen’s “reflective practice” model of design. Figure created after [323]. 301
11.10 OODA loop with its four main cognitive processes: observe, orient, decide, and act. Figure created after [383]. 302
11.11 Cognitive processes within the Kolb’s “experiential learning cycle” model for learning [195]. 304
11.12 Within an organization, knowledge is shared through communication. 307
12.1 The hierarchical model of design knowledge proposed in this work. 311
12.2 Progressive build-up and clarification of hierarchical bodies of design knowledge. 312
12.3 Cyclic models of cognition. 318
13.4 Event categories for participant P1. 329
13.5 Event categories for participant P2. 342
13.6 Event categories for participant P3. 366
13.7 Event categories for participant P4. 377
13.8 Event categories for participant P5. 389
13.9 Event categories for participant P6. 408
14.10 A simple parametric model based on data propagation. Data emanates from node p0 and is being consumed by nodes q0, q1 and q2. 409
14.11 A simple subjunctive parametric model SPM. 409
14.12 All three possible realizations of the subjunctive parametric model SPM shown in Figure 14.11 are the three concrete parametric models CPM1, CPM2 and CPM3, depending on which alternative node (p0, p1 or p2) has been selected in the subjunctive node S. 410
14.13 A subjunctive node can contain subjunctive nodes and/or concrete nodes. In this example, subjunctive node S0 contains concrete nodes p0, p1 and p2. Subjunctive node S1, on the other hand, contains one subjunctive node S0, and two concrete nodes p3 and p4. 411
14.14 The three possible partial realizations of the subjunctive model SPM00 shown in Fig. 14.15. 411
14.15 Further realization of the subjunctive model SPM00 into three concrete models CPM000, CPM001, and CPM002, depending on which alternative node (p0, p1 or p2) has been selected in the subjunctive node S0. 411
14.16 Design states in subjunctive parametric graphs are created through the combined process of realization and instantiation. 412
15.17 Is-a thesaurus graphs of important concepts, as per WordNet [251] and WordVis (wordvis.com). 419
Chapter 1

Introduction

1.1 Why Work With Many Designs at Once?

In this work, I explore ways to enable computational support for managing *multiple, simultaneously-available computational design models* so that users can *view, compare, and edit many such models in parallel*. I depict this schematically in Figure 1.1 by juxtaposing: (a) the conventional approach to computing where the user can access and modify just one model at a time (as is common with the nowadays prevalent *single-state* systems), and (b) the approach developed in this work where the user manages, and interacts with, multiple models in parallel (thus using a *multi-state* system).

![Figure 1.1: A schematic depiction of conventional (single-state) vs. alternatives-enabled (multi-state) software tools. In the conventional approach, an user interacts with one computational model at all times. In the alternatives-enabled approach, an user interacts with, and manages, multiple model alternatives in parallel.](image)

The question naturally poses itself, why would one want or need to work with multiple, simultaneously available design models in our software applications of choice? After all, the users managed to *make do* with single-state systems for quite some time now. My answer to that question is four-fold, as follows:
I. Designers frequently work with multiple design alternatives in parallel. The first, and arguably the most compelling reason, is the evidence that many designers work with multiple design alternatives in parallel. I will further elaborate this evidence in Chapter 4, however here I will mention that this evidence can be derived from scholarly studies (see e.g., [11, 10, 208, 26, 216, 347]), works of art and design (see e.g., Figures 1.2, 1.3, as two representative examples), and expert opinion\(^1\).

For instance, in sketching and painting, artists might come up with *multiple variants* of a sketch or drawing (or parts thereof). They might sketch a part of the scene multiple times, each time slightly differently and superposed on top of each other, in order to evaluate what might work best in terms of composition or overall visual effect; alternatively, a number of different *variations* of a design might be sketched, and arranged in a rectilinear grid or some other layout configuration (Figure 1.2).

![Figure 1.2: Multiple design alternatives in product design, arranged in a rectilinear grid. Drawings of earbuds © Patrick Murphy [259]. Reused under CC BY 2.0 license.](image)

As another representative example, architects frequently create multiple *massing models* of a building design, using materials such as wood, cardboard, clay, and acrylic, in order to evaluate and choose the most appropriate overall form of the building (Figure 1.3).

For the sake of completeness, I should mention that designers at present also can, and occasionally do, utilize their current computational tools (i.e., CAD applications, vector graphics editors, raster image editors, text editors, ...) for the purpose of managing alternatives, but at the cost of

\(^1\)Linus Pauling, an American quantum chemist and biochemist, stated that “If you want to have good ideas you must have many ideas. Most of them will be wrong, and what you have to learn is which ones to throw away” [82].
1.1. WHY WORK WITH MANY DESIGNS AT ONCE?

Figure 1.3: Multiple, cognate “massing” models for a building designed by architect Frank Gehry, and built in New York City in 2011. Reproduced by permission © Vitra Design Museum. Photo credit: Bettina Matthiesen [242].

labour-intensive and time-consuming ad hoc workarounds\(^2\). The jury is still out on why, how, and in what design phase designers develop multiple design variants, but later on in this dissertation (in Chapter 4) I present concrete evidence that this particular aspect of design work is real, and should be taken into account as a potentially valuable and fruitful avenue of research, as well as an incentive to augment our tools with the corresponding, alternatives-aware capabilities.

II. Intellectual curiosity. The second part of the answer to the question “why enable simultaneously available, multiple design models in our computational tools?” is simply due to the fact that we might, after all, just be satisfying our intellectual curiosity and the scholarly urge to discover what kind of tools would one get if we just go ahead and explore computational tools for managing multiple design models—even if we completely disregard the fact that many designers indeed do develop (and historically have always been developing) multiple design alternatives. In other words, exploring “computational support for many design alternatives at once” just for the sake of “exploring computational support for many design alternatives at once” is a valid and worthwhile research cause in and of itself, with or without the evidence from professional and art practices that many designers frequently work with a number of designs at once.

III. Increase in computing power available. The third part of the motivation for this work is, simply, the fact that contemporary personal computing devices nowadays boast sufficient computing power to, at last, handle multiple computational models at once, at comfortable interactive rates and with a relative processing ease. In fact, one may claim that practically all software tools have historically been conceptualized and implemented as single-state systems also (partly) due to the

\(^2\)Such ad hoc workarounds include saving different model versions under different file names, or taking multiple raster image screenshots of those models so that many might be compared at once, and so on.
fact that previous generations of personal computing devices were relatively underpowered, thus unable to effortlessly and comfortably handle multiple computational models at a time. This much computing power, which will undoubtedly increase further in the future, translates directly into the capability to maintain multiple computational models in a consistent state, and at interactive rates, for model sizes and model complexities much greater than the ones typically used in the past.

IV. Novel display technologies. Last but not least, planar computer displays continue to grow in total surface area available, thus allowing the users to work with many more views to display design alternatives with, as compared to conventional, relatively small computer displays which could fit much less information on their screens. Additionally, large planar screens offer increasingly higher resolutions, and alternative interaction modalities, such as touch-based and gesture-based interaction (Figure 1.4). If I extrapolate a little bit, one might surmise that, in the future, wall-sized screens will be a common occurrence, thus potentially allowing simultaneous interaction with thousands of design alternatives in parallel.

Figure 1.4: Microsoft Surface Hub, an 84”, wall-mounted, touch-enabled display. In this depiction, two designers collaborate on a chair design, with four alternative chair designs shown at the top of the screen. Image credit: Microsoft Corporation promotional materials [249].

Other classes of display devices that are at the time of this writing experiencing revival and reawakened interest, and which might provide additional impetus to future alternatives-enabled applications, include various immersive Virtual Reality (VR), Augmented Reality (AR), and hologram-based head-mounted displays (HMD), thus potentially enabling interaction with multiple design alternatives in three-dimensional spaces. Commercial devices available at the time of writing include for example Oculus VR Rift [394], and Microsoft HoloLens [248].
1.2 Research Overview

Previous section provided the motivation and rationale for carrying out the research described in this dissertation. This leads us directly to the causal chain of derived entities depicted in Figure 1.5.

1.2.1 Situation of Concern

Research problems, including the one addressed in this work, come into existence due to some situation of concern that needs resolving. In this work, the situation of concern is that users of computational tools cannot easily view and compare multiple digital models at once, as they can with analogue models, sketches, and other expressions of design in the physical world. This forces the users to adopt workflows and methods which do not actually fix the problem, but merely offer alternative ways to avoid it. However, such methods are lengthy, inefficient, and error-prone. The overall goal of this research is to address and mitigate this situation of concern.

1.2.2 Research Problem Statement

The situation of concern described above applies to most types of computational tools used nowadays: computer-aided design (CAD) tools, word processors, spreadsheets, raster and vector image editors, and so on. In this thesis, however, I focus on a particular class of computational tools, parametric CAD (pCAD) tools, or CAD tools that put a particular emphasis on the relations (constraints) among the elements of the geometric model. From now on, I will use either of the acronyms, pCAD and/or CAD, to indicate such tools.

I can now formally put forward the research problem statement, derived from the situation of concern that needs resolving:

**Research problem.** Current pCAD tools lack adequate support for working with multiple geometric design alternatives.
1.2.3 Research Intent Statement

From the research problem statement, I can now proceed directly to the research intent statement:

**Research intent.** The intent of this research is to systematically inquire into the nature of pCAD tools that allow their users to quickly and easily view, edit, compare, and manage multiple geometric design alternatives in parallel.

From the research intent statement above I can set apart the following four central components of research conducted in this work, and which help address the research problem:

1. **Form of solution:** a computational (pCAD) tool. The purpose of a solution is to mitigate the situation of concern, by providing the target users with an interactive system which does not force them to use aforementioned ad hoc workarounds, but to allow them to interact with, and manage, multiple design alternatives in a more direct fashion.

2. **Target users:** designers using pCAD tools in their work. While the limitations of single-state systems apply to most classes of software applications and their respective user communities, in this thesis I focus on users who commonly and frequently use pCAD tools in order to solve geometric design problems they are interested in.

3. **Human activity:** view, edit, compare, and manage multiple geometric design alternatives at once.

4. **Levels of support:** direct, quick, and easy. The direct level of support refers to the fact that the users should be able to work with many designs in parallel, without having to recur to workarounds employed with single-state systems. In other words, the users should be able to work directly with entire sets of design alternatives, and with many designs in parallel. I define the other two levels of support (quick and easy) relative to existing (single-state) systems. In a way, these levels of support follow immediately from the direct level of support, because instead of executing a series of roundabout tasks using a single-state tool, those tasks would be implemented natively in a multi-state pCAD tool.

1.2.4 Research Contributions

The main research contributions of this work are as follows:

1. **Interaction design guidelines for building alternatives-enabled pCAD tools.** While the existing literature reports several proposals of systems supporting alternatives, no comprehensive
set of design guidelines for specifying richly-interactive tools has yet emerged, particularly in the field of pCAD. In this work, I propose such a set of guidelines, or principles, which a HCI expert or practitioner should consider when designing alternatives-enabled pCAD tools.

2. **Domain vocabulary.** As another research contribution, I present a vocabulary of concepts pertinent to alternatives-enabled tools, such as what is an “alternative”, what are “collections of alternatives”, and so on.

3. **Conceptual model for alternatives-enabled computational tools.** A high-level, conceptual model capturing the notion of alternatives, collections thereof, as well as tasks related to managing and manipulating such models.

4. **Prototypes and other “design solution” artifacts.** I also describe implemented prototypes, wireframe mockups, GUI sketches, and other artifacts produced in the course of conducting this research, as a type of “embodied knowledge”.

5. **Novel interaction techniques in parametric CAD.** Specific, novel interaction techniques pertinent to parametric CAD, such as subjunctive snaps and subjunctive editing operations.

6. **Design development process.** I further describe specifics and particulars of the design development process that was adopted in order to come up with these research contributions.

### 1.2.5 Thesis Statement

I can now formally state my thesis statement:

**Thesis statement.** Alternatives-enabled pCAD tools designed in concordance with the interaction design guidelines and design principles presented in this dissertation support design work.

In order to validate this thesis, and support the claim of the thesis, in this work I endeavour to answer the following main research questions:

RQ1. What are the essential system features of novel multi-state pCAD tools that can support design work?

RQ2. What are the interaction design guidelines and principles for specifying, building and evaluating multi-state pCAD tools?

RQ3. What methodological approach can be followed that can help system designers build multi-state pCAD tools?
1.3 Dissertation Outline

This dissertation is structured into Parts and Chapters as follows:

1.3.1 Part I: Preliminaries

- **Introduction.** Chapter 1, the section you are currently reading, introduces this work in a concise manner.

- **Theoretical Framework.** Chapter 2 gives the theoretical framework within which I situate this work, by describing terms such as design, designer, plan, problematic situation, artifact, implementation, tool and computational tool, and ways of how we can conceive them and think about them.

- **Research Approach.** Chapter 3 describes issues related to research approach and methodology, including research outcomes i.e., the bodies of new knowledge generated through this work.

1.3.2 Part II: Background and Related Work

- **Background.** Chapter 4 provides the background for this work by listing empirical evidence (both from research and from professional practice) that designers indeed do work with multiple design alternatives in parallel. It also offers a theoretical justification for why designers would do so.

- **Related Work.** Chapter 5 describes past research pertinent to this work, i.e., work related to computational support for managing and interacting with multiple alternative design solutions, organized into appropriate categories.

1.3.3 Part III: Stages of Research

- **Pre-Design Phase.** Chapter 6 describes the pre-design phase i.e., how I went about to collect, and then select, feasible and promising approaches to interacting with multiple design alternatives.

- **Conceptual Design.** Chapter 7 describes the conceptual model i.e., a very high-level, abstract representation of the concepts (constructs, entities) present in the solution, as well as their relationships. A major part of the conceptual model are metaphors which I chose to follow in order to cast the UI and associated tasks in terms of familiar environments and objects.
• **CAMBRIA 1 Prototype.** Chapter 8 describes the first higher-fidelity prototype application, named CAMBRIA 1, for handling multiple design alternatives in the domain of 2D parametric pCAD. It also describes the findings of a qualitative study of the prototype based on expert feedback.

• **CAMBRIA 2 Prototype.** Chapter 9 describes an extended version of the higher-fidelity prototype, named CAMBRIA 2, as well as findings obtained from a case study.

• **Consolidation.** Chapter 10 proposes a taxonomy of interaction primitives, a list of interaction design guidelines, and a list of design principles for building alternatives-enabled pCAD systems.

### 1.3.4 Part IV: In Closing

• **Research and development process.** Chapter 11 describes the process that I followed, or went through, in order to generate the contributions of this work.

• **Conclusions.** Chapter 12 lists the contributions of this work, presents conclusions, and suggests open research questions as well as future work.

### 1.4 Chapter Summary

This chapter introduced, in a concise manner, the motivation behind this work, an overview of the research conducted, thesis statement, as well as the dissertation outline by parts, and chapters.
Chapter 2

Theoretical Framework

Human beings actively perceive and understand their surroundings, achieve goals, and perform the activity of designing, that is, make plans for creating new artifacts, or creatively influence undesirable, problematic situations into more desirable or favourable ones.

Human beings are further characterized by their extensive use of tools, or implements which help them to achieve their goals in a faster, easier, or simply more convenient fashion, as compared to achieving those same goals un-aided. One important class of tools are computational tools, of which one kind (i.e., alternatives-enabled computational tools) is the main focus of this work.

In this thesis, I view “designing” from two different yet equally important perspectives:

1. **Tool designer’s perspective.** The activity of designing a novel class of computational tools is in and by itself an activity of designing, which falls with the subject field of interaction design (IxD), or HCI design.

2. **User’s perspective.** On the flip side, the users of the computational tool developed in this work attack their own design problems in a specific application domain. Thus, to build effective computational tools, it helps to understand the nature of the problems that such tools are meant to solve: design problems.

Given the importance, then, of the high-level concepts such as design, designer, plan, problematic situation, artifact, implementation, tool and computational tool within the context of this work, in this Chapter I give a succinct summary of these terms, and describe ways of how we can conceive them and think about them. This, in turn, will provide us with a general theoretical framework within which to think about the research outputs produced by this work.
2.1 Designing: The Activity of Design

The type of computational tool being developed in this work helps its users to design. Further, the work described in this dissertation is itself concerned with the activity of designing a particular artifact i.e., a novel pCAD tool. However, many different definitions of the notion of “designing”, or the activity of design, exist. Table 2.1 contains a number of such definitions, listed in the chronological order of their publication.

---

Table 2.1: Various definitions of the activity of designing, as found in the literature.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Design is the process of inventing physical things which display new physical order, organization, form, in response to function.”</td>
<td>C. Alexander [13]</td>
</tr>
<tr>
<td>“Everyone designs who devises courses of action aimed at changing existing situations into preferred ones.”</td>
<td>A. Newell and H. Simon [264]</td>
</tr>
<tr>
<td>“Design is the creation of artifacts that are used to achieve some goal.”</td>
<td>W. Mayall [243]</td>
</tr>
<tr>
<td>“Designing is a reflective conversation with the materials of a situation.”</td>
<td>D. Schön [323, p. 78, p. 94]</td>
</tr>
<tr>
<td>“Design is structuring argumentation to solve ‘wicked’ problems.”</td>
<td>H. Rittel [306]</td>
</tr>
<tr>
<td>“Design is initiating change in man-made things.”</td>
<td>J. Jones [179]</td>
</tr>
<tr>
<td>“Designing is an activity that consists in specifying an artifact product, given requirements on the artifact.”</td>
<td>W. Visser [391]</td>
</tr>
</tbody>
</table>

---

What it means “to design”, therefore, differs from one scholar to the next, from one problem domain to another, and on the scope of consideration. Considering this variability of the notion of “designing” and its dependency on context, I give not one but three different (yet equally valid) working definitions of designing, along with the explanation of where a particular definition applies best:

(i) Designing as a general cognitive activity,
(ii) Designing as a reflective conversation with the materials of the situation, and
(iii) Designing as the activity of creating a new product or service.
2.1.1 Designing as a General Cognitive Activity

If we first consider designing to be a general, pervasive human activity which can be applied to many different contexts and different problems to be solved (from e.g., finding the best route through a city, winning a game of chess or tennis, creating a house plan, writing a letter, or baking a cookie), I adopt the rather general definition of designing by A. Newell and H. Simon [264, 343]:

Definition 2.1. Designing.

To design means to devise courses of action aimed at changing existing situations into preferred ones.

In this sense, the act of designing is a general mental activity which “is not the monopoly of those who call themselves designers”[306]. Thus, any person (or intelligent agent) designs, plans, and devises.

2.1.2 Designing as a Reflective Conversation

Next, when I consider designing to be an activity of working out the form of some artifact, for instance by sketching, modelling, or writing, and using some medium, representation or technique, I adopt the definition by D. Schön [323]:

Definition 2.2. Designing.

To design means to have a reflective conversation with the materials of a problematic situation.

In this case, an analog (e.g., paper-based) or digital canvas might serve as the “material of a situation” that contains marks, objects, or abstract symbols describing a problematic situation. The designer has a “reflective conversation” with the material (i.e., the canvas) by using the pencil, mouse, or some other device, and making and erasing marks (either analogue or digital ones) on it. The type of computational tool being developed in this work, then, can be seen as providing a “material” with which the users will be able to have a meaningful reflective conversation.

I should note that Schön’s definition can be reinterpreted in the terms of the Newell and Simon’s definition, if we consider the situation of “not having the design” of the planned artifact as the undesired one, which we would like to transform into the desirable situation of “having a design” of the artifact. And vice versa, if we would like to “devise courses of action”, this implies some medium (or material) in which we devise such courses of action, whether it’s merely in our memory, or on paper, or some other medium.
2.1.3 Designing as the Creation of an Artifact Design

Finally, I will also now give a third definition of designing, which best captures the notion that when we *design* something, we usually go about creating specifications for a new *product, artifact, or service*. While the definitions by Newell and Simon (the problematic situation of not having such product or service) and Schön (having a reflective conversation with a material in order to come up with the blueprint for such product or service) still hold, the following definition by W. Visser [391] which is more specific in that regard:

**Definition 2.3. Designing.**

To design means to *specify* an artifact product, given requirements on the artifact.

In addition, and in accordance with this definition, a design *project* might be considered to be a “*temporary endeavor undertaken to create a unique product, service, or result*”[294].

2.2 Designers: The Practitioners of Designing

Within the context of this work, which is exploring computational support for multiple design alternatives, I presume that a *designer* is a person (or even an abstract *agent*) who uses or takes advantage of all available resources, tools, and media, including the computational tools developed as a part of this research, in order to perform his or her work. Such a person might alternatively take on other names, such as: *user, expert, professional, creative worker, artist,* or even an abstract *agent*.

Reflecting the definitions of the activity of designing given in Section 2.1, I thus arrive at the following three equally valid yet increasingly specific definitions of the concept of designer:

**Definition 2.4. Designer.**

A designer is a person or an agent who *(in increasing order of specificity)*:

1. Devises courses of action aimed at changing existing situations into preferred ones.
2. Has reflective conversations with the materials of a problematic situation.
3. Specifies an artifact product, given requirements on the artifact.
2.3 Designs: The Outcomes of Designing

Equivalently, the following definition of the concept of design, or the product of the activity of design, echoes the three definitions of designing given in Section 2.1:

**Definition 2.5. Design.** (noun)

A design is \((\text{in increasing order of specificity})\):

1. A course (or plan) of action.
2. The materials (or representations, see Section 2.4) of a problematic situation.

2.4 Design Representations

The tool being developed in this work will allow their users to, among other tasks, represent their designs. The concept of representation is a central one in cognitive science\(^1\) in general, and in designing in particular. Many definitions of “representation” exist, but perhaps the most general one is given by Greco [137]:

**Definition 2.6. Representation.**

A representation is anything that stands for something else.

For example, a rectangle sketched on a napkin, or a rectangle drawn using the computational tool developed in this work, might represent the outline of a room in a house design. While useful, the above definition is perhaps overly general. I will now give a second, more specific definition of representations, following Palmer [280] and Newell [265]:

**Definition 2.7. Representation.**

A representation is a \(\text{triple} \langle W, R, W \rightarrow R \rangle\) consisting of the following:

\(^1\)Cognitive science is ‘the interdisciplinary study of mind and intelligence, embracing philosophy, psychology, artificial intelligence, neuroscience, linguistics, and anthropology’ [372]. Since designing is fundamentally a cognitive activity involving high-level thought processes such as decision making, comprehension, perception, evaluation, reasoning, memory storage, memory retrieval, and learning, cognitive science provides a suitable framework to reason about, and study design.
2.4. DESIGN REPRESENTATIONS

1. \( W \): the set of elements that are being represented, or the *represented world*.

2. \( R \): the set of *symbols* that denote elements from \( W \), or the *representing world*.

3. \( W \rightarrow R \): the *interpretive process*, or the process executed by an actor and which maps the represented world \( W \) onto the representing world \( R \).

According to the definition above, for instance, some real-world building might be mapped into symbol \( b \), or into some group of hand-drawn lines i.e., a sketch.

Examples of representations, internal (i.e., mental) or external, include: hand-drawn or imagined floor layouts, sketches, images, signs, software prototypes, physical prototypes, written statements, literature, verbal statements, models\(^2\), and others.

Representations should fulfill the following three principal purposes [268, p. 274]:

1. **Exploration.** In the course of designing, a designer usually comes up with many different ideas, thus exploring the design problem at hand. A representation should allow and empower the designer to easily, quickly and reliably record those alternative ideas.

2. **Communication.** No designer works in isolation, and a representation should therefore be convenient and adequate to communicate ideas to other individuals. Even a solo designer should be able to “communicate with himself”, i.e., able to understand a representation that was created by himself some time in the past, for instance.

3. **Analysis.** A representation must support the analysis and evaluation of a specific design, in order to predict the performance of the final artifact. For instance, a designer may visually evaluate a sketch and estimate the appropriateness of the design (represented by this sketch) for a context, or a requirement. As another example, an interactive artifact might undergo a number of different inner states; a representation (such as e.g., *state diagrams* or *activity sequences*) might describe these states and help the designer choose the most appropriate alternative.

2.4.1 Attributes of Representations

In terms of classifying representations, different authors describe different (yet overlapping) sets of attributes of representations [391, p.119][165, 50, 271]. Merging these classifications results in a superset of attributes (or dimensions) of representations (Figure 2.1).

\(^2\)A model can be defined as “a purposeful abstraction that allows one to reduce complexity by focusing on certain aspects” [189].
### Attributes (dimensions) of representations [391, 50, 271].

| (1) Internal | ←→ | External (Mind) |
| (2) Internal | ←→ | External/Surface (Device) |
| (3) Initial | ←→ | Final |
| (4) Shareable | ←→ | Non-Shareable |
| (5) Transient | ←→ | Durable |
| (6) Self-Generated | ←→ | Ready-Made |
| (7) Abstract | ←→ | Specific |

**Figure 2.1:** Attributes (dimensions) of representations [391, 50, 271].

As another characterization of representations, this time according to their **abstractness**, Fish and Scrivener [118] propose the following continuum (Figure 2.2), from most abstract to least abstract:

\[
\text{(abstract descriptions)} \\
\downarrow \\
\text{abstract signs} \rightarrow \text{verbal descriptions} \rightarrow \text{mathematical descriptions} \rightarrow \text{diagrams} \rightarrow \text{icons} \\
\rightarrow \text{maps} \rightarrow \text{sketches} \rightarrow \text{illustrations} \rightarrow \text{photographs} \rightarrow \text{films and solid models} \rightarrow \text{moving 3D models} \\
\downarrow \\
\text{(concrete and spatially specific depictions)}
\]

**Figure 2.2:** The “abstract ←→ specific” continuum for visual-spatial representations according to Fish and Scrivener [118].

The novel, alternatives-enabled computational tool developed in this work utilizes the following representation types:

- A specific, hybrid, and custom device-internal representation combining (i) a geometric (i.e., visual-spatial) device-internal representation encoding geometric strokes, lines, curves, shapes, colours, pen thickness, . . . , and (ii) textual device-internal representation for indicating element names, numerical values, and other non-graphical displays.

- Device-external representations (renderings) such as lines and curves shown on computer display.

In terms of attributes of representations shown in Figure 2.1, the user can use this tool to create his or her own external representations, which can be shared or not (as a file), are durable,
2.5. DESIGN PROBLEMS

that can be self-generated or ready-made (by importing an existing file), and can be positioned at the middle of the “abstract ←→ specific” continuum shown in Figure 2.2.

2.5 Design Problems

The notion of design problem is a very important and central one within the context of this work. As I have already mentioned at the beginning of this Chapter, in terms of design, in this work we operate at the following two levels:

(i) The level of coming up with alternatives-enabled pCAD tools.
(ii) The level of addressing geometric design problems using the pCAD tool developed in this work.

In order to define “design problems”, I build upon the definition of designing and design given in Section 2.1 and 2.2, and give the following two definitions as follows.

Definition 2.8. Design problem.

A design problem is (in increasing order of specificity):

1. An existing, unsatisfactory situation that a designer desires to change into a preferred one.

2. The problem of devising an artifact product, given initial requirements on the artifact.

Non-routine design problems are considered to be hard, convoluted, and complex. As distinguished from the so-called “well-defined”, “well-structured” or “tame” problems [90, 337] for which we know how to solve them and when did we find the solution, most non-routine design problems are considered to belong to any of the following “hard” problem classes:

• “Ill-defined” problems by W. Reitman\(^3\) [299, 300],

• “Ill-structured” problems by H. Simon [337],

• “Wicked” problems by H. Rittel and M. Weber [307, 308, 53, 78], or

• “Messy” problems by D. Schön [323].

In the following, I will succinctly describe each of these problem classes, starting with the easiest class of problems to solve: well-defined problems.

\(^3\)The first appearance of expressions “well-defined” can be traced back to McCarthy [244] who called well-defined problems those for which a test, which can evaluate a proposed solution in a finite number of steps, exists. The expression was later used by Minsky in his classification of “well-defined” and “ill-defined” domains [253].
2.5.1 Well-Defined Problems

Well-defined problems [90, 337] are considered “tame” because they are structured in advance (by “structured” I mean that all its constituent states and feasible solving strategies are known in advance). Formally, Simon [337] gives a list of six requirements that a problem must satisfy in order to be considered a well-structured one:

1. **Objective solution tests exist.** Any solution can be tested by the application of a well-defined, objective criterion. Furthermore, an automated process for applying said criterion exists.

2. **A problem space exists.** At least one formal problem space (Figure 2.3) exists where the pertinent knowledge states (initial states, goal states, and intermediary states) can be represented.

3. **Transitions can be represented.** All legal (i.e., permitted) moves (i.e., state changes) between knowledge states can be represented in the problem space above. Transitions are achieved through the use of operators, which have the power to legally transform one state into another. Figure 2.3 shows transitions \( a \rightarrow c \), \( c \rightarrow e \), \( e \rightarrow h \), \( h \rightarrow i \), and \( i \rightarrow l \).

4. **New knowledge can be represented.** Any new knowledge acquired by the problem solver can be represented in at least one problem space.

5. **A problem space for the physical world exists.** If the problem solver also acts upon (in) the external world, such actions must be reflected in a problem space that models natural laws and represents the external world.

6. **Computable.** All five preceding points must be achievable using only practicable amounts of computation.

![Figure 2.3: An instance of problem solving by searching in a “problem space”.](image-url)
To solve a well-defined problem, i.e., reach one of its goal states, the problem solver might apply a number of well-defined problem solving (algorithmic) strategies, such as for example exhaustive (brute-force) search, adversarial search, difference reduction, means-ends analysis, and heuristics. Examples of well-defined problems include solving an algebraic equation for an unknown variable $x$, solving a logical puzzle, or checking one’s email. As I have already mentioned at the beginning of this Section, non-routine design problems are, however, inherently and qualitatively much harder and more complex than well-defined problems.

### 2.5.2 Ill-Defined Problems

“Ill-defined” problems can be defined simply as a residual of well-defined problems i.e., we say that a problem is ill-defined when it is not well-defined [338]. Different characterizations of ill-defined problems exist in the literature; in this section I review several of them. For a quick overview, Table 2.2 lists some major differences between well- and ill-defined problems.

<table>
<thead>
<tr>
<th>Category</th>
<th>Well-defined problems</th>
<th>Ill-defined problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem specification</td>
<td>complete, unambiguous</td>
<td>incomplete, ambiguous</td>
</tr>
<tr>
<td>Solution evaluation</td>
<td>definite criteria</td>
<td>no definite criteria (no stopping rule)</td>
</tr>
<tr>
<td>Knowledge utilized</td>
<td>represented in one or more problem spaces in advance of problem solving</td>
<td>many sources (problem spaces) that cannot be determined in advance and need to be integrated</td>
</tr>
<tr>
<td>Operators</td>
<td>enumerable, all known in advance</td>
<td>no exhaustive, enumerable set</td>
</tr>
<tr>
<td>Solution path</td>
<td>exists, predetermined (but has to be found)</td>
<td>no predetermined solution path exists</td>
</tr>
<tr>
<td>Examples</td>
<td>chess, Towers of Hanoi, theorem-proving</td>
<td>design, planning, management, document and music composition</td>
</tr>
</tbody>
</table>

Table 2.2: Comparing well- and ill-defined problems. Table adapted from [146].

Thus, most non-routine design problems can never be fully and unambiguously specified. Further, they have no definite design solution “goodness” criteria i.e., for most design problems, one can always come up with “better” design solutions. They require knowledge from many different sources, have no definite set of well-defined design transitions, and have no pre-defined series of steps in order to arrive at an appropriate design solution.
Again, remembering the two levels of designing pertinent to this work (design of a tool, and designs created using this tool), both can be said to embody properties of ill-defined problems: in other words, both levels are “messy”, ambiguous, and with no predetermined solution path.

### 2.5.2.1 Ill-Defined Problems (Reitman)

As probably the first historically and contextually relevant treatment of “hard” problems, Reitman [299, 300] describes so-called “ill-defined” problems\(^4\), that is, those characterized around the notion of constraints, or conditions that control all the aspects of the problem. An “open” constraint is the one that has \textit{“one or more parameters the values of which are left unspecified”} [299, p. 31], and illustrates this by giving an example of composing a fugue (a type of musical form) for which, according to Reitman, at the beginning there is only one (implied) constraint:

\textit{The composition produced has to be a fugue.}

Thus, the composition produced has to follow the rules and the structure of this particular musical form, which for all practical purposes implies another, relatively large set of open (sub-)constraints. Designers (or problem solvers in general) become aware of these constraints gradually, as they proceed in their work: “\ldots constraints are generated from one transformation of the problem to the next” [299, p. 296]. As the designing proceeds, the constraints must at some time be “decided on” i.e., their values specified by the designer. Further, designing is a social activity and an open constraint might thus be satisfied in many different ways: “\ldots no solution to an ill-defined problem can count on universal acceptance” [299, p. 153]. One particular designer will, in general, satisfy an constraint differently from another designer. Therefore, in order to successfully solve a realistic ill-defined problem, an agreement (consensus) has to be reached among all the designers (or problem solvers in general) involved in a problematic situation.

### 2.5.2.2 Ill-Structured Problems (Simon)

Inspired by Reitman’s notion of “ill-defined” problems, Simon proposes the related notion of “ill-structured” problems [337]. According to Simon, all ill-defined problems can become “well-structured” in the course of problem solving. In other words, in ill-structured problems, some part of well-structuredness is missing and the problem has to be “structured” first (before any problem-solving is attempted); ill-structured problems are simply those which are not well-structured i.e., have no well-defined knowledge states, operators, and/or solving strategies. According to Newell, only humans are able to structure ill-structured problems into well-structured ones [266].

Once an ill-structured problem becomes well-structured, it is relatively easy to solve using standard, well-known techniques, such as exhaustive (brute-force) search, adversarial search,

\(^4\)Reitman uses the expression “ill-defined problems”, but I also use it in this work to denote all problems which are not well-defined.
difference reduction, means-ends analysis, and heuristics. Thus there is no inherent difference
between “ordinary” problem solving and design problem solving, only that design problems might
require more structuring beforehand. To illustrate how ill-structured problems might become well-structured, Simon [337] gives an example problem of how to design a house which, while initially
being ill-structured, can gradually be decomposed into a set of well-structured sub-problems.

2.5.2.3 Wicked Problems (Rittel and Weber)

As another influential characterization of “hard” problems, Rittel and Weber [307] describe so-called wicked problems, a subtype of ill-defined problems commonly encountered in social planning, but now universally understood to describe most real-world problems, including non-routine design problems. Wicked problems are characterized by the following ten prominent features:

1. Wicked problems elude definitive formulations.
2. Wicked problems have no stopping rule.
3. Solutions to wicked problems are not true or false, but good or bad.
4. There is no immediate and no ultimate test of a solution to a wicked problem.
5. Every solution to a wicked problem is a “one-shot operation” because there is no opportunity
to learn by trial and error, thus every attempt counts significantly.
6. Wicked problems do not have an enumerable or an exhaustively describable set of potential
solutions, not is there a well-described set of permissible operations that may be incorpo-
rated into the plan.
7. Every wicked problem is essentially unique.
8. Every wicked problem can be considered to be a symptom of another problem.
9. The existence of a discrepancy representing a wicked problem can be explained in numerous
ways. The choice of explanation determines the nature of the problem’s resolution.
10. The designer (i.e., user, planner, . . . or the person trying to solve a wicked problem) has no
right to be wrong.

2.5.2.4 Ill-Definedness Revisited (Lynch et al)

Finally, in a recent (2009) synthesis of prior characterizations, Lynch et al [231] suggest that ill-defined problems and domains:

1. Involve “open-textured” concepts (i.e., context-dependent concepts which require human
judgment [36, 37]) and competing domain principles that are subject to debate;
2. Lack widely accepted domain theories identifying relevant concepts and functional relations;
3. Cannot be readily partitioned into independent subproblems;
4. Have prior cases that seem, at the face value, to be inconsistent;
5. Involve the need to reason analogically with cases and examples;
6. Have a large or complex solution space that prohibits one from enumerating all possible characterizations or solutions;
7. Lack formal or well-accepted methods to verify solutions;
8. Lack clear criteria by which solutions are judged;
9. Are not considered to be solved when one solution is presented but may be readdressed by multiple, often distinct, solutions;
10. Involve disagreements among domain experts regarding the adequacy of the solutions; and
11. Require solvers to justify their solutions through argument.

This concludes the overview of design problems according to their complexity or hardness, and I will now proceed to describe ways to operationally represent such problems.

### 2.5.3 Representing Design Problems Through Design Requirements (DRs)

In addition to the broad definitions of ‘design problems’ given on page 17, I now give yet another (operational) definition based on their set of (explicitly or implicitly stated) design requirements. In other words, I define design problems extensionally\(^5\). I will define design requirements formally later on, but for the time being I will just state that a ‘requirement’ is anything that is essential for a design. We thus arrive at the following definition of design problems:

\[
\textbf{Definition 2.9. Design problem. (operational definition)}
\]

A design ‘problem’ is a set of design requirements specifying an artifact.

Note that the set of design requirements is a dynamic one, in the sense that it fluctuates during the course of designing [384]: some requirements might be discovered as the designers proceeds in his or her work, with new requirements added to the set, some requirements removed from the set, and so on. In other words, and as has already been mentioned in previous sections, non-routine design problems are being continuously “discovered” i.e., their corresponding sets of requirements are being continuously modified as well. In the words of D. Schön [323, p. 40]:

\[
\textit{In real-world practice, problems do not present themselves to the practitioners as given. They must be constructed from the materials of problem situations which are puzzling, troubling, and uncertain.}
\]

\(^5\)Enumerating all possible designs for a general design problem, thus defining it intentionally, is for all intents and purposes impracticable, except for the simplest design problems.
2.5. DESIGN PROBLEMS

Design problems can therefore be represented by their design requirements (DRs). I define design requirement as follows:

**Definition 2.10. Design requirement.**

A requirement is [167]:

(i) A condition or capability needed by a user to solve a problem or achieve an objective.

(ii) A condition or capability that must be met or possessed by a system, system component, product, or service to satisfy an agreement, standard, specification, or other formally imposed documents.

(iii) A documented representation of a condition or capability as in (i) or (ii).

Although taken from the domain of software engineering, Definition 2.10 can be applied to most types of design problems. DRs can be specified using either natural language and/or formal requirements models [289]. In fact, a set of DRs is simply just another form of representation, so all what was written about representations in Section 2.4 applies to DRs as well. So for example, DRs can be represented in a designer's mind only as well. In many cases though, requirements are represented in a written form, as shown in Figure 2.4.

1. **Patient registration.**
   
   1.1 The system must support the entry of patient information on arrival at the hospital, including name, address, name of physician, date and time of arrival.
   
   1.2 The system must allocate a patient ID to be quoted in all transactions while the patient is registered at the hospital.
   
   1.3 Patient registration should take a trained operator no more than five minutes.

2. **Patient record update.**

   2.1 The system must enable subsequent update and correction of patient information by nursing staff on the ward.

**Figure 2.4:** High-level functional design requirements for a patient information system, subdivided into two functional areas. Example based on [268, p. 80].

Low-level requirements can be progressively aggregated into requirements at a higher level, these into yet higher levels, and so on [13, 105, 214]. For example, in architectural design, DRs
can be decomposed into [105]: Mission, Goals, Performance Requirements (PR), and Concepts. In software engineering design, DRs can be decomposed into [214]: System Purpose, System Design Principles (SDP), Blackbox Behavior (BB), Design Representation (DP), Physical Representation (PR). The higher the level of a DR, the more pertinent the question ‘why do we need this feature?’ is. And vice versa, the lower the level of a DR, the more the question ‘how do we implement this feature?’ applies.

Pohl [288] suggests that the process of developing and refining design requirements moves along the axes of the following three dimensions:

1. Completeness of specification dimension,
2. Formality of representation dimension, and
3. Degree of stakeholders’ agreement dimension.

The designer starts near the origin of this three-dimensional space (see Figure 2.5), where he has ‘opaque personal views’ of the system being designed. The overall goal, therefore, is to move this process to the desired output (top right corner of the cube) which is indicated by the following three coordinates: (1) completely specified requirements, (2) formal representation prepared, and (3) agreement reached by all the stakeholders in the process.

![Figure 2.5](image)

*Figure 2.5: When developing design requirements, the designer moves in a three-dimensional space, from the initial idea (which is mostly opaque, informal, and lacks agreement), to the desired output (i.e., design requirements) which maximizes the three principal dimensions (completeness of specification, formality of representation, and full degree of stakeholders’ agreement). Based on [288].*

Due to the scope and space limitations of this work, I will not touch upon many other important concepts related to requirements, such as requirements elicitation, analysis, specification, traceability, modeling, and management. You may refer e.g., to [164] for further information. I note here that the subject field of “design requirements” is a relatively large one, with devoted journals and conferences, in multiple areas of application. For instance, the study and application of design
requirements in the field of architectural design is typically called *architectural programming*. In the areas of mechanical and software engineering, it is commonly called *requirements engineering*, and so on.

### 2.5.4 Summary on Design Problems and DRs

Non-routine design problems are a type of ill-defined problems. Novel computational tools being developed in this work will help their users to address such problems. On another level, the design of alternatives-enabled pCAD tools, the main focus of this work, is of course in itself an ill-defined problem as well. A design problem can be represented by its corresponding set of design requirements or DRs.

### 2.6 Design “Solutions”

We have seen that design problems can be represented by its (dynamic, ever-evolving) set of design requirements. What about design “solutions”? Obviously, a design solution should be defined as “something” that satisfies the design problem’s set of DRs.

That “something”, i.e., any and all design solutions, come in the form of a representation, so Section 2.4 on page 14 applies to design solutions too. Some important classes of design solution representations include sketches, models, and prototypes.

#### 2.6.1 Sketches

Sketches are the *quintessential* external representation for design [323, 112, 404, 155, 206]. Sketching is an indispensable part of the process of designing itself as well. For instance, Ullman [385] intends to “*show the necessity of drawing during all the developmental stages of a mechanical design*”. In architectural design, Schön and Wiggins [325] state that “*the designer sees what is there in some representation of a site, draws in relation to it, and sees what has been drawn, thereby informing further designing*”.

Sketches are especially useful in the early, conceptual phases of design, when the design problem is still largely unknown (i.e., design requirements are usually still imprecise and incomplete). For example, Figure 2.6 shows an early graphical user interface for the prototypes developed in this work. Due to not knowing the problem, designing the right artifact is very difficult [298]. It is thus in the designer’s best interest to co-evolve [102] the problem-solution ensemble (for a discussion on this ensemble, see Section 2.6.3 on page 28) as quickly and thoroughly as possible; “*each (design) move is a local experiment which contributes to the global experiment of reframing the problem*”[323] i.e., to the understanding the design problem at hand; in other words, it is beneficial
to undertake design moves quickly. And indeed, sketches fit the bill perfectly, because they are:

1. Quick,
2. Timely,
3. Inexpensive,
4. Disposable,
5. Plentiful,
6. Suggestive,
7. With minimal detail needed,
8. With appropriate level of refinement, and

Furthermore, the study by Black [39] suggests that prolonging the use of sketches, due to their inherent ambiguity, contributes to a broader exploration of the design solution space. With regard to conventional CAD, sketches are graphic metaphors [155] for both 1) the real artifact being designed, and 2) drafted digital CAD object. Therefore, they act as a bridge (or intermediary representation) between mental representations, and the CAD drafting representation.

### 2.6.2 Prototypes

According to Drexler [111, p. 220], in engineering, it is the prototyping, and not planning of engineering designs, that brings most advances. Brooks [52] states that prototyping is necessary for

---

6In other words, the conversation with the “materials of the situation” (in this case, with sketches) is faster.
eliciting the requirements of the computational tool under development, since design problems are in general under-constrained. In other words, using a fixed set of requirements will as a rule lead to incongruities, which can then consequently be dealt with only by further requirement elicitation i.e., further prototyping.

In HCI, a prototype is a design solution proposal, i.e., ‘the representation of the interface par excellence, reflecting both its appearance and behavior’ [371]:

Prototypes emulate the intended design as faithfully as the implementation resources and technology permit. Interface design can be very difficult to picture and evaluate in the abstract or from stimulus and behaviorally impoverished representations such as static drawings or verbal description.

If we view the process of eliciting design requirements as a learning process (‘learning the design problem’), Argyris and Schön [24] state that the process of learning generally involves the detection and correction of errors through feedback loops, which in design are afforded through the construction of prototypes. Any ‘problematic’ design situation is by definition uncertain, and an effective approach to deal with its uncertainty is to build a prototype [323].

2.6.2.1 Shortcomings of Prototyping

For the sake of completeness, despite their significant advantages, prototypes, especially high-fidelity prototypes, carry some significant potential problems of their own [371, 303]:

- **Concern shifting (from interface design to programming).** The designer’s attention is dangerously shifted away from interface design, to the specifics of the implementation, dealing with a variety of development-related time-consuming tasks. To remedy this, authors [371] suggest ‘explicit interventions’ and techniques to recover the designer’s attention back towards interface design, such as group design, and using environments for rapid, visual construction of user interfaces.

- **Expensive to build (time, costs).** Prototypes by definition are supposed to demonstrate a good part of the intended functionality, which requires significant expenditure of total human effort in terms of time and financial cost.

- **Having to deal with non-essential aspects of the interface.** The implementer of the prototype inevitably deals with many non-essential aspects of interaction design.

- **Testers’ comments are superficial.** Testers of the prototype frequently comment on its superficial features, instead of on its “deep features”.

• *Developers’ inertia.* Prototype implementers are frequently unwilling to make changes to the existing code base due to “sunk costs” i.e., total hours spent developing the prototype.

• *Unreasonable expectations.* A well-executed prototype could incite big appetites and lead to overly optimistic expectations or excessive demands on the final, “release” revision of the application.

In a nutshell, then, prototyping is an immensely useful and effective technique in the context of design, however it must be approached with care.

### 2.6.3 Problems and Solutions: Two Sides of The Same Coin

A design problem, and any of its solutions, can be regarded as two sides of the same coin, i.e., they can be considered as belonging to a unit (ensemble), as depicted in Figure 2.7.

![Diagram of design ensemble](image)

**Figure 2.7:** A design problem and any of its solutions should be regarded as an unit.

In other words, designing consists of both the *problem analysis* (i.e., the exploration of what to design), and the *solution synthesis* (i.e., figuring out what are the parts of the solution, and structuring those parts together) [342, 343]. I am thus in the position to postulate another (this time more operational) definition of designing, in addition to the existing definitions given in Sections 2.1.1, 2.1.2, and 2.1.3:

**Definition 2.11.** *Designing. (operational definition)*

To design means to co-evolve the design problem and its associated set of design solutions.

Further, while in this Chapter I have opted to treat design problems first (in Section 2.5), and design solutions second (in this section, Section 2.6), this does not imply that designers proceed
in this manner, from first defining the problem, and then coming up with the solution. Indeed, as we will see in this section, a design problem and any of its solutions can be considered to co-evolve in parallel.

For instance, "designers move rapidly to early solution conjectures, and use these conjectures as a way of exploring and defining problem-and-solution together" [85]. In [102] Dorst and Cross confirm the hypothesis that design problems and their solutions co-evolve in parallel, by conducting protocol studies involving experienced designers. In this study, the authors observed moments when evolving and unstable problem and solution spaces are “fixed” by moments of creative insight; on those moments, the problem has been “framed” (see Schön’s notion of problem framing and reframing, e.g., [323, pages 40, 93, 129, and 132]). In mechanical engineering design, Ullman et al. [384] give evidence that designers introduce new, additional DRs as they proceed with design, (1) those based on the designer’s pre-existing domain knowledge, and (2) those derived by evaluating potential design solutions. In a recent study, Yu et al. [415] confirm the co-evolution hypothesis by doing a protocol study involving architectural designers performing tasks in parametric design environments. Maher and Poon propose a model [233] for “creative design” based on exploring the problem-solution tandem in parallel, using genetic algorithms to evaluate the fitness of individual design solutions (Figure 2.8).

On a related note, designers are, generally speaking, inherently solution-oriented, instead of problem-oriented. For instance, Lawson [208] found out that designers employ “solution-focused” solving strategies, as distinguished from “problem-focused” solving strategies usually employed by scientists. In [83] Cross likewise claims that expert designers appear to focus on problem generation, instead of on problem analysis. In [199] Kruger and Cross claim that “the 'generate' activity

![Figure 2.8: Co-evolving design problems and design solutions using genetic algorithms. Based on Figure 1 in [233].](image)
was the most frequently occurring single activity, thus tending to confirm the solution-focused nature of design thinking”. In a study of engineering designers [216], the authors found that the designers tried to approach the design problem in a “generative” way by creating multiple solutions, instead of trying to understand the problem through problem analysis. In a similar vein, Hillier et al [159] suggest that designers often “conjecture” a solution by proposing a design, in order to learn more about the design space i.e., uncover design requirements and design constraints.

To conclude, while a design problem and any of its solutions do evolve in parallel, we can say that most (but not all) designers tend to generate design solution candidates in order to “feel out” the design problem, i.e., elicit corresponding design requirements. This elicitation and augmentation of the corresponding set of DRs, in turn, informs the designer’s generation of subsequent design solution proposals, and so on, which thus closes the “problem ←→ solution” feedback loop.

2.7 Computational Tools are “Cognitive Artifacts”

The principal objective of this work is to develop a novel computational tool, by solving the corresponding design problem. How can we conceptualize and define “computational tools”?

Humans possess the remarkable ability to modify their environment through the conscious creation of man-made (“artificial” [343]) objects, or artifacts. Artifacts range from the simplest ones, such as those protecting us against cold (clothing, camp fire, hut), improve our eyesight (prescription eyeglasses) or restore our health (drugs, antibiotics), all the way to very complex ones such as skyscrapers, spacecrafts, computer chips, or large software packages.

One special class of artifacts are so-called cognitive artifacts [271], or those artifacts which in some way enhance human performance while performing various cognitive tasks, such as designing a house, summing up numbers, or creating a new drug. Cognitive artifacts store, retrieve, process, and represent information. They are called “cognitive” because they aid with and enhance cognitive processes such as decision making, comprehension, perception, evaluation, reasoning, memory storage and retrieval, and learning. Examples of cognitive artifacts include written language, mathematics, pencil on paper, computational/CAD tools, and the novel computational tool being developed in this work.

**Definition 2.12. Cognitive Artifact.**

A cognitive artifact is an artificial device that maintains, displays, or operates upon information in order to serve a representational function, and that affects human cognitive performance [271, p. 17].
**Figure 2.9:** Action cycle mediated through a cognitive artifact. Note that in this scenario actions are exerted through the artifact, and that the designer perceives the world indirectly, through a representation generated by the artifact. Based on [273, 271].
Figure 2.9 illustrates what happens when an individual sets a goal, and then tries to achieve that goal through the use of a cognitive artifact, according to Norman’s seven-step action cycle [273, 271]. Accordingly, tying into previous sections on design, and viewing designing as the activity of reaching the goal of solving a design problem (page 17), I now define the concept of “goal” as follows:

**Definition 2.13. Design goals.**

Design goals are to (in increasing order of specificity):

1. Reach a preferred situation, starting from an existing, unsatisfactory one.
2. Devise an artifact product, given initial requirements on the artifact.

Cognitive artifacts thus *mediate* between their users, and the situation (i.e., the world being sensed and represented by the cognitive artifact), by enabling the activities of “execute action” and “evaluate the situation” which have to bridge two gaps or gulfs, the gulf of execution and the gulf of evaluation:

1. **Gulf of execution.** Cognitive artifacts enable users to execute *actions*, which in turn affect the situation (i.e., either the representation of the situation, or the situation itself). The *gulf of execution* refers to how difficult it is for the user to make those changes, using the cognitive artifact.

2. **Gulf of evaluation.** Cognitive artifacts enable users to perceive and thus to evaluate the situation (again, either the representation of the situation, or the situation itself). The *gulf of evaluation* refers to how difficult it is for the user to evaluate the situation using the cognitive artifact, as well as to how well and how accurately does the artifact measure (i.e., detect, observe, sense), interpret, and represent the actual situation.

Although cognitive artifacts are supposed to enhance human cognitive abilities in some way, this does not of course imply that cognitive artifacts actually *modify* an individual’s innate cognitive abilities; rather, in Norman’s own words [271], they change the structure of the task at hand so that the total three-part “human+task+artifact” system (i.e., the “human+task+tool” system) appears to be more efficient than the unaided human (i.e., the “human+task” system):

*Rather, [cognitive artifacts] change the nature of the task performed by the person. When the informational and processing structure of the artifact is combined with the task and the informational and processing structure of the human, the result is to expand and enhance cognitive capabilities of the total system of human, task, and artifact.*
Within the context of this work, this implies that a multi-state tool will change the current “human + task $T$ + single-state tool” system to a more efficient “human + task $\overline{T}$ + multi-state tool” system, where $\overline{T}$ stands for task(s) made possible through the application of multi-state tools.

Cognitive artifacts can change the nature of the task that has to be performed by doing the following [271]:

1. Pre-compute all the needed entities and/or actions (such as preparing a checklist in advance),
2. Distribute needed entities and/or actions across people (such as instructing other people to remember specific items), and
3. Somehow change the entities and/or actions that have to be used or performed.

**Example: Checklists** Norman further gives [271] a concrete example of cognitive artifacts: checklists, which are a “list of items (names or tasks etc.) to be checked or consulted” [412] and which “aid in reducing human error” [148]. From the system view, checklists appear to enhance the memory of the person using them, while from the personal view, they actually change the task from:

(i) Needing to plan and remember all the actions on the list, to a much easier one of
(ii) Remembering to use the list and then simply going sequentially through individual actions on the list.

**Four types of representations associated with cognitive artifacts** As stated by Norman [271], a cognitive artifact might operate on situations that exist not only on the real world but also in virtual worlds only; objects populating such “virtual” situations then might or might not be mapped into real objects at a later time. For instance, a sketch of a house or a 3D model of a house might or might not be mapped into a real, built house later on. There are four levels of representation present in cognitive artifacts:

1. The represented world (real or virtual). Example: a constructed, real house (real world), or a 3D model of a building in a video game.
2. Artifact-internal representation, i.e., the representation within the artifact. Example: a 3D geometric model within a CAD tool.
3. Artifact-external (i.e., surface) representation obtained by transforming the artifact-internal representation. Example: a graphical rendering of a 3D geometric model within a CAD tool.
4. *Mind-internal representation*, i.e., representation internal to the user’s mind. Example: the internal representation acquired by a designer when he or she views a graphical rendering of a 3D geometric model within a CAD tool.

### 2.7.1 The Significance of “Cognitive Artifacts” for this Work

Alternatives-enabled computational tools are a type of cognitive artifacts, and as such they are presumed to take on a certain role in the overall design process, thus modifying the original task \( T \) to some transformed task \( \tilde{T} \). The exact nature of the original task is elaborated further in this dissertation, but here I note that the original task \( T \) is, generally speaking, the task of managing multiple design alternatives when using conventional, non-alternatives-enabled (i.e., single-state) computational tools.

### 2.8 Chapter Summary

This Chapter presented a whirlwind, concise summary of fundamental concepts related to the activity of design and providing computational support for it, such as who are designers, what are design representations and cognitive artifacts, and what are design problems and design solutions.

An extended overview might include a discussion on *design paradigms* or *general perspectives on design* such as the *symbolic information-processing* approach [264], the *situated-action* approach [360, 272], the *reflective practice* approach [323], and the *construction of representations* approach [391], as well as a discussion on *design process models*, such as the propose-critique-modify (PCM) family of design methods [67], the UK Design Council’s “double-diamond” model of design [51], and others. In the next chapter discussing the research approach and methodology, Chapter 3, I will touch upon several design process models specific to HCI design.
Chapter 3

Research Approach

The overall goal of research, or the systematic quest for knowledge, is to generate, evaluate, and validate new contributions to knowledge in some given subject domain. Such contributions are made by practitioners of research, or researchers, who in many instances follow specific processes called research methods. Methods, in turn, are commonly aggregated into research approaches [173] or methodologies\(^1\) [327, 88]. I depict these concepts schematically in Figure 3.1, showing the creation of (or augmentation of existing) bodies of knowledge through the application of research methods\(^2\) aggregated into a larger research approach or methodology.

![Figure 3.1: Research methods, usually aggregated into research approaches or methodologies, are the means to augment existing (or create new) bodies of knowledge.](image)

In order to describe the research approach that I adopted in this thesis, in this chapter I structure my exposition in three parts, as follows:

---

\(^1\)“Methods are like ingredients, whereas methodology is like a recipe” [327].

\(^2\)I should note at the outset that research methods can be considered to be a type of knowledge as well. Thus a method not only directs and helps us to arrive at new knowledge in a systematic manner, but can also be researched, studied, and published like any other body of knowledge. See e.g., [43] for an overview of how to develop and evaluate research and development processes (methods) in the subject field of HCI.
(i) **An overview of traditional research (§ 3.1).** I begin with an overview of research as understood in traditional, long-established fields of inquiry such as philosophy and sciences, including the corresponding notion of “knowledge” (§ 3.1.1), the main output of research, as well as a summary of the cognitive processes and research methods commonly found in traditional research (§ 3.1.2). I follow immediately by a brief discussion of why the framework of traditional research is too limited and restrictive in and for design research (§ 3.1.3).

(ii) **An overview of research in HCI design (§ 3.2).** I then describe what it means to conduct research in HCI design, beginning with a review of related work on research in general design (§ 3.2.1). I continue by providing a knowledge-centric view of HCI design research (§ 3.2.2.1) as well as an activity-centric view of HCI design research (§ 3.2.2.2), followed by an overview of the important topic of evaluation in HCI design (§ 3.2.3), and a description of criteria of “success” in HCI research (§ 3.2.4).

(iii) **My research approach (§ 3.3).** Finally, I describe the research approach that I adopted in this work, in order to answer the posited research questions, and support my thesis.

### 3.1 Traditional Research

In this section, I provide a concise overview of the nature of knowledge produced in traditional (i.e., non-design) fields of inquiry such as philosophy, natural sciences, and social sciences, as well as typical cognitive processes and activities employed by researchers in order to arrive at such knowledge.

#### 3.1.1 Knowledge: The Output of Research

Many definitions of the concept of “knowledge”, the principal output of research, exist; you may refer to e.g., [314] for a good number of such definitions. In this work, I adopt the following working definition [2] of knowledge as understood in traditional sense:

<table>
<thead>
<tr>
<th>Definition 3.1. Knowledge.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge is facts, informa-</td>
</tr>
<tr>
<td>tion, and skills acquired by</td>
</tr>
<tr>
<td>a person through experience</td>
</tr>
<tr>
<td>or education.</td>
</tr>
</tbody>
</table>

Therefore, as per definition above, knowledge is relative to a person: knowledge is something that one believes, and that belief should at the same time be true and justified\(^3\). The definition

---

\(^3\) JTB, or “justified true belief” account of knowledge, as first postulated by Plato [353] in *Theaetetus* and *Meno* [74].
above also represents a nod towards the fact that an individual can “know something” not only in
the sense of possessing facts or information about “something that is the case”, but also how to
“do something”. In fact, knowledge can be classified and juxtaposed in a number of different ways;
some of the most important classifications include:

(i) **Conceptual vs. procedural knowledge.** Conceptual or “know-that” knowledge [319] knowl-
edge can be thought of as a “connected web of concepts”, or a network in which the links
(relationships) among discrete pieces of information are as prominent as those pieces them-
selves [157, Ch. 1]. On the other hand, procedural or “know-how” knowledge [319] can be
thought of as procedures, rules, algorithms, a series of steps, manipulations, operators, or ac-
tions\(^4\). This classification of knowledge is further illustrated by the existence of structural and
behavioural models in software engineering [352], or even by the existence of two distinct,
prominent Internet encyclopedias, one specializing in conceptual knowledge (wikipedia.org)
and another one specializing in procedural knowledge (wikihow.org).

(ii) **Representable vs. tacit knowledge.** A person can be in possession of tacit [290, 395, 44]
or “knowing-in-action” [324] knowledge, or the type of practical, know-how knowledge that
usually is not (or cannot be) easily expressed or represented. Such knowledge is typically
gained through “informal learning, either from one’s own experience or from that of a mentor
or colleague” [395, p. 173].

(iii) **Knowledge by scope.** Knowledge by scope or purview (from wider to narrower scope) in-
cludes the concepts of:

- Research **paradigm** [200] or research **program** [204],
- **Theory**, or “an organized system of accepted knowledge that applies in a variety of
circumstances to explain a specific set of phenomena” [413],
- **Model**, or “a purposeful abstraction that allows one to reduce complexity by focusing on
certain aspects” [183] of the problem, and
- **Law**, or small theories that underwent extensive, repeated observation and thus consid-
ered to be universally true, e.g., Newton’s Second Law of Motion \(F = m \cdot a\), Moore’s
Law [255], and Fitts’ Law [119].

A belief, in order to qualify as knowledge, must necessarily be **true** because an individual can, as we all know, believe
in something that actually is not true; and a true belief must in addition also be **justified**, because if a person believes in
something that is indeed true, that something still has to be defended, explained, and argued to be true.

\(^4\)Consider also the ancient Greeks’ distinction between **epistêmê** (‘knowledge’) and **theoria** (‘viewing’, ‘contemplat-
ing’), as well as **technê** (‘craft’, or ‘art’) and **praxis** (‘action’, ‘doing’) [5, 96].
(iv) **Knowledge by hierarchical levels.** Knowledge by hierarchical levels, such as e.g., Ackoff’s “DIKW” model of knowledge ([3] and Figure 3.2), or Newell’s knowledge model [265], where one level of knowledge builds upon, and is emergent from, the previous one.

![Diagram of Ackoff's DIKW model](image)

**Figure 3.2:** Ackoff’s “data-information-knowledge-wisdom” (DIKW) model of knowledge [3].

Hierarchical models of knowledge provide the first hint of *cognitive processes* involved in the generation of knowledge; namely, a layer of knowledge *emerges* from a lower layer through certain cognitive activities and processes. I provide some detail regarding such processes in the next section, § 3.1.2.

Therefore, knowledge, the principal output of research, can be understood, classified, and interpreted in many different ways. This section, § 3.1.1, provided just a very compact and brief overview of knowledge in order to delineate its role in the overall research process; as I will show in the sections that follow, aforementioned classifications of knowledge will help us to better understand the types of knowledge generated in HCI research, and in the research described in this thesis. For further information on the nature of knowledge, the reader may refer to a number of standard texts on epistemology, for example [70, 211, 27, 302, 212, 354].

### 3.1.2 The Activity of Research

As for the *activity* of research which results in knowledge, and as traditionally understood, numerous definitions exists in the literature e.g., [186, 262, 387, 91, 80]. To best cover the research described in this dissertation, I adopt the following definition:

---

**Definition 3.2.** *Research.*

Research is any activity or process which increases one’s knowledge or understanding of a topic or issue.
3.1. TRADITIONAL RESEARCH

Cognitive processes through which one arrives at knowledge, in the most general sense include [353]: (1) perception, (2) introspection, (3) memory recall, (4) reasoning, and (5) receiving testimonials (for example, by attending a lecture, or by reading a book). Similarly, C. Peirce [283] suggested the following four primary methods to “fixate one’s beliefs”: (1) “a priori” method (based on logic, deductive reasoning, and intuition\(^5\)), (2) through an authority on knowledge, (3) via tradition (facts believed and justified on the basis of successful, long-standing practice), and (4) via science (i.e., through empirical evidence). From the considerations above, as well as directly [378], we can thus discern among two principal (traditional) meta-methods for generating new knowledge, the reflective method (also sometimes called the theoretical method), and the empirical method:

(i) **Reflective method.** Based on reflection, contemplation, introspection and abstract reasoning. Within Western thought, this method can be traced back to ancient Greece, with thinkers such as Plato, Aristotle, Euclides, and many others\(^6\). As a representative illustration, Plato (428–348 BC) states in *Meno* and *Phaedo* [74] that all human beings are already born with inner sources of knowledge, and thus any instance of “knowledge learning” is therefore merely the act of recognizing pre-existing, intrinsic knowledge, through a process called *anamnesis* [326]. The reflective method, as the principal way to generate and validate new knowledge, remained dominant until the appearance of the empirical method during the Renaissance.

(ii) **Empirical method.** Some authors [189] credit G. Galilei as the “father of modern science”, being one of the first thinkers to advocate and employ the empirical method, based on the availability of empirical evidence to justify beliefs, and as a rigorous, objective way to produce and validate new knowledge. Succinctly, a study utilizing the empirical (or “scientific”) method consists of the following principal steps [62]:

(i) Form a hypothesis.

(ii) Identify variables (independent and dependent) present in the problem; usually just a few independent variables are considered in an empirical study.

(iii) Decide how independent variables will be varied (or “controlled”).

(iv) Eliminate complexity and confounding effects so that it is clear that it is actually the change in independent variables that causes changes in dependent variables, not some other factors.

(v) Measure dependent variables (or “metrics”), as they change as a function of varying independent variables.

---

\(^5\)Cf. for example Descartes’ *cogito ergo sum* [91, p. 4].

\(^6\)For instance, both Aristotle [25, Book 1, Part 3] and Bertrand Russell [315] considered Thales of Miletus (624–546 BC) to be the primogenitor and initiator of Western philosophy.
(vi) Apply statistics to analyze results, and derive findings that support or refute the hypothesis.

Modern notion of “research”, of course, subsumes both the reflective and the empirical method in order to generate and justify new knowledge. Furthermore, a large number of different incarnations (or subtypes) of both the reflective and the evidence-based method exist; for this, I refer the reader to standard texts on research methods and research methodology [186, 262, 378, 304, 91, 80, 81], as well as to various sources on the “science of science” [283, 204, 364, 200, 291].

3.1.3 Limitations of Traditional Research Methods in Design

However, as I will describe in some detail in the next section, § 3.2, the traditional research methods (reflection, and the empirical method), while still useful, are of limited utility and scope in design research. Designing is at a fundamental level primarily concerned with the creation of future states (i.e., future artifacts, or future situations) [121], while traditional disciplines (i.e., philosophy, sciences) are fundamentally concerned with the study of what already is. This leads to the following important distinctions:

(i) Firstly, the working definition of knowledge (p. 36) has to be augmented so as to include the notion of design artifacts, such as e.g., design prototypes, as significant carriers of knowledge i.e., as a type of “embodied knowledge”[86, 357] or “the thinking embodied in an artifact” [121]. Design artifacts or parts thereof, understood as knowledge, can be copied, modified, and varied [86]. Traditionally, design artifacts have not been deemed to be on equal footing with well-established and long-used carriers of knowledge such as e.g., books, documentaries, or scientific reports. Furthermore, as distinguished from sciences which are very positivistic7 in their outlook, design artifacts are also much more subject to aesthetic judgement, and personal tastes.

(ii) Reflective method alone cannot fully encompass, grasp, and predict the full scope and richness of many designed artifacts, non-interactive and interactive ones. Design artifacts, and designs themselves, are frequently very complex in terms of possible interactions and relations among their constituent parts [12]. Thus, to deal with this complexity, designers in general have to offload their thinking to external representations characteristic of design, such as sketches, models, and interactive prototypes. In other words, designers (1) generally cannot easily “design in their head” only, and (2) use “designerly” external representations in order to facilitate the design process.

7Positivism denotes the philosophical stance that the only valid knowledge is only that derived and verified through the application of the empirical (i.e., scientific) method [207, p.179].
(iii) The empirical (scientific) method, likewise, by its very nature is usually limited to confirming only very narrow conjectures, involving at most a handful of independent and dependent variables, thus leading to overly simplified (i.e., reduced\textsuperscript{8}) situations. However, no non-routine design problem can be reduced to just a handful of variables, and must due to many interactions of its constituent parts be approached holistically. When applied indiscriminately in designing, the empirical method can lead to the elimination of promising branches of design early on in the creative process [419].

3.2 Research in HCl Design

Considering the limitations of traditional research in designing, how can we think about, conceptualize, and operationalize research pertinent to design, of which HCl design is but one sub-field? How does research in HCl design differ from traditional research? I begin with a review of related work on what it means to conduct research in design-based fields of inquiry.

3.2.1 What is “Design Research”? 

Design problems\textsuperscript{9} are associated with the planning, conception, and actualization of increasingly complex man-made objects, or artifacts. The phenomenon of artifact complexification has over the years led to the professionalization of many traditional lines of work and trade, which have previously been considered to belong to vernacular arts, or crafts [14]. Non-routine design problems have been characterized as being especially hard and complex, and have interchangeably been called and classified as “ill-defined” [299, 300], “ill-structured” [337], “wicked” [307, 308, 53, 78], or “messy” [323]. Design problems, in turn, gave rise to new considerations and efforts to generate, validate, and systematize knowledge associated to solving such problems, leading to the development of various design-based research methods and approaches [18, 121, 419, 113, 114, 357].

The apparently “special” nature of design problems, and of the activity of designing, has also led to claims that design requires a “new kind of science” [342] or that successful instances of designing require special, “designerly” ways of knowing [86, 84]. Design is presumed to be based on abductive reasoning [87, p. 27][237], as distinguished from traditional fields of inquiry in which researchers primarily employ inductive and deductive modes of reasoning. In the words of L. March [237]:

“Logic has interests in abstract forms. Science investigates extant forms. Design initiates novel forms. A scientific hypothesis is not the same thing as a design hypothesis."

\textsuperscript{8}Reductionism denotes the mistaken belief, or a philosophical stance, that complex phenomena can be completely understood in terms of their components [126, 414].

\textsuperscript{9}See Chapter 2, § 2.5 on page 17 on the particulars of design problems.
A logical proposition is not to be mistaken for a design proposal. A speculative design cannot be determined logically, because the mode of reasoning involved is essentially abductive.”

March further argues that the deductive or “top-down” reasoning (that is, reaching a certain conclusion from a set of premises, within a closed domain of discourse), and the inductive or “bottom-up” reasoning (reaching a weak or strong conclusion by generalizing or extrapolating from bits of evidence) are suitable for analysis and evaluation, while design is intrinsically about synthesis. In other words, deduction is all about what must be; induction is all about what actually is; while abduction suggests that something may be [237].

On the other hand, in a more recent development [116, 117, 127] it has been argued that the boundary between design and sciences might not be as clearly delineated as previously thought, and that design and science might even be considered to be “identical twins” [127]. As the argument goes, sciences are also concerned with creation of a number of different “artificial” objects or artifacts, such as scientific models, theories, and other cognitive artifacts. If supported by further research, this hypothesis could lead to wide-ranging implications and could “open up new vistas for synergetic cross-disciplinary discussion of design and methodology” [116].

Next, Frayling [121] inquired about what it actually means to conduct research in arts and design, and suggested the following division of concerns between traditional research, and research in arts and design (Table 3.1).

<table>
<thead>
<tr>
<th>Traditional Research</th>
<th>Research in Arts and Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>going over old territory</td>
<td>concerned with the new</td>
</tr>
<tr>
<td>arms-length</td>
<td>engagement</td>
</tr>
<tr>
<td>artefacts</td>
<td>artefacts</td>
</tr>
<tr>
<td>words</td>
<td>deeds</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison of conventional research, and research in arts and design, according to Frayling [121].

Traditional research thus goes over “old territory” in the sense of being concerned with what already is, while in design we are primarily concerned with the creation of future artifacts or states. Furthermore, in traditional research we study our objects of inquiry in an detached and objective manner i.e., by keeping them at “arms-length”, while in design we are actively engaged with the situation, and its materials, through the hands-on creation of artifacts. Frayling then goes on and suggest that design research can be divided into three types: research through design, research for design, and research about design:

---

10 It appears that the spelling “artifact” is favoured in North America, and “artefact” in the UK. In this dissertation, I use the first spelling.
• **Research through Design (RtD)** in which researchers develop artifacts iteratively, thus creating novel products and services which can then be used as a “stepping stone” to subsequent products and theories.

• **Research for Design (RfD)** in which researchers come up with ways that can help designers to reframe (and thus help them solve) the problems they commonly face in their work. Typical outputs from RfD include design patterns [14], frameworks [63, 130], guidelines [154, 175, 269, 268, 333], recommendations, implications and methods [13, 43].

• **Research about Design (RaD)** is concerned with the scientific inquiry on how humans actually perform the activity of design, and is as such closest to natural and social sciences. Typical outputs from RaD are scientific theories, for instance theories accounting for cognitive processes found in the activity of designing.

According to Frayling, any design research project will in general involve all three types of design research (RtD, RfD, RaD) to a lesser or greater degree, and in different proportions.

![Figure 3.3: Any instance of interaction design research lies on a point in the triangle defined by: professional practice, scientific studies, and playful exploration. Adapted from [113, 114].](image)

Echoing Frayling’s three-way classification of research types in generic design, Fallman and Stolterman [113, 114] suggest a similar classification in the field of interaction design (IxD), and propose that any instance of IxD research can be placed at any point within their triangular model, as depicted in Figure 3.3.
Fallman’s triangular model, then, congruently with the Frayling’s model, partitions design research into the areas of (1) explorative (speculative) design, where the designer playfully explores what is desirable and possible, (2) design practice, where the designer honours the contractual agreement with a business client and solves context-driven, particular design problems in a synthetic fashion, and (3) design studies, where the researcher utilizes the empirical method in order to support or refute scientific hypotheses related to the processes and entities found in designing. I summarize and juxtapose the two classifications of design-based research in Table 3.2.

<table>
<thead>
<tr>
<th>Frayling [121]</th>
<th>Fallman [113, 114]</th>
<th>Main question:</th>
<th>Context:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research-through-design</td>
<td>Explorative design</td>
<td>“What is possible?”</td>
<td>Playful experimentation</td>
</tr>
<tr>
<td>Research-for-design</td>
<td>Design practice</td>
<td>“What is real?”</td>
<td>Professional practice</td>
</tr>
<tr>
<td>Research-about-design</td>
<td>Design studies</td>
<td>“What is true?”</td>
<td>Sciences</td>
</tr>
</tbody>
</table>

Table 3.2: Juxtaposition of types of research in arts and design [121, 113].

### 3.2.2 Two Ways to View HCI Design Research

In this section, I present two ways to look at HCI design research: a knowledge-centric view (§ 3.2.2.1), and an activity-centric view (§ 3.2.2.2). Both views can be considered to come forth and evolve from the notion of design state, or the totality of all the variables defining it: the set of all the representations\(^{11}\), bodies of knowledge, design artifacts, and research artifacts created in the course of solving the associated design problem. Ullman [385] for instance considers that a “design state” consists of the designer’s inner, mental representations, notes, drawings, sketches, computer records, accepted design proposals, constraints, and strategies. I visualize a design state by a simple circle, labeled with an identifier or name (Figure 3.4).

\[\text{Figure 3.4: A schematic visualization of a design, or a “design state”, marked by the identifier “1”.}\]

**Knowledge-centric view.**

In the knowledge-centric view of HCI research, I examine how various research artifacts and knowledge representations originate from, and are developed around, the design ensemble which itself springs from a design state, as shown in Figure 3.5.

\(^{11}\text{Please refer to Chapter 2, Section 2.4, page 14 for an overview of the concept of “representations”.}\)
A design ensemble\(^{12}\) consists of a design problem and its associated set of design solutions. Within the co-evolution hypothesis [323, 384, 85, 102, 415], to design means to co-evolve both (1) the design problem representations and (2) design solution representations in parallel. In general, research artifacts related to the co-evolution of the design ensemble build upon the lower-level research artifacts (in the first place, upon the domain vocabulary), and higher-level constructs such as design guidelines are then synthesized by the researcher based on his accumulated knowledge and experience as he proceeds to solve the problem. To better illustrate this point, Shneiderman and Plaisant [333, p. 40] claim that HCI design guidelines are usually derived from “experimental results, experience with existing interfaces, and knowledgeable guesswork”. I thus adopt the position that higher-level knowledge types, such as design guidelines, can thus be best understood as a kind of understanding, as per Ackoff’s DIKUW hierarchical model of knowledge (cf. Fig. 3.2). In other words, design guidelines are not generated randomly or in some haphazard, undefined manner; instead, they emerge and are specified by the researcher only after sufficient experience has been acquired and accumulated by said researcher through extensive design development and evaluation of design artifacts, as well as subsequent introspection, reflection, and sense-making activities.

**Activity-centric view.**

On the other hand, in the activity-centric view of HCI research, I examine how research artifacts and knowledge representations originate from, and are developed around, various research and development activities as shown in Figure 3.6. In research through development (RtD), the structure of design history provides a kind of scaffolds for all the research activities taking place in a given research project; artifacts thus built then lend themselves to other forms of research (RfD, RaD, and others).

---

\(^{12}\)See Chapter 2, Section 2.6.3 on page 28 for a discussion on design ensembles.
CHAPTER 3. RESEARCH APPROACH

3.2.2.1 Research in HCI Design: A Knowledge-Centric View

In this section I detail the knowledge-centric view of HCI design research shown in Figure 3.5 above, i.e., I provide a picture and a justification of HCI research viewed through the lens of bodies of knowledge (cognitive artifacts, knowledge representations) usually encountered and generated in HCI research.

To start with, designers are inherently solution-oriented [108, 208, 159, 216, 83, 199]. That is, most (but not all) designers are “solution-focused” who prefer to synthesize solutions in order to explore the design space, as distinguished from scientists who are “problem-focused” [208] i.e., prefer analysis over synthesis.

Figure 3.7: Every design artifacts opens up its own, associated design sub-space. Many such sub-spaces can, when unified, lead to a larger design space [357]. In this schematic example, design artifacts $a_0, a_1, \ldots, a_8$ opened up their own design sub-spaces $D_0, D_1, \ldots, D_8$ which, when unified, may lead to a larger design space $D = D_0 \cup D_1 \cup \ldots \cup D_8$.

Design artifacts are any bodies of knowledge or representations that help solve a particular design problem. Any design artifact can be considered to be defining and opening its own design
sub-space [357]. Further, many such artifacts (relative to a design problem) can therefore be considered to open its associated design sub-spaces which, when unified, may lead to a larger, global design space. Each coordinate in this space contributes to the understanding of what is possible and desired in a design solution (Figure 3.7).

I thus augment the definition of knowledge as understood in traditional research (see § 3.1.1, page 36) with the fact that in design research, the researcher also by necessity interacts with design artifacts who might have been (or not\textsuperscript{13}) developed and/or used by the researcher during the overall research process.

\begin{definition}
Knowledge.
Knowledge is facts, information, artifacts, and skills acquired, developed and/or used by a person through experience or education.
\end{definition}

As has been mentioned elsewhere in this dissertation, design artifacts thus can be understood as “embodied knowledge” [86, 357] or “the thinking embodied in an artifact” [121]. Considering the expanded definition of knowledge above, research in HCI design and related fields also produces a number of design artifacts now understood as research artifacts, including but not limited to: target domain vocabularies [238], HCI theories, HCI models, HCI laws (for instance, Fitts’ Law [119]), design requirements (functional or non-functional), design specifications, design instantiations (e.g., design mockups, and prototypes), design guidelines and principles\textsuperscript{14} [154, 175, 269][268, p. 374][333, p. 40], and HCI design development and/or evaluation methods [43]. Figure 3.8 builds on, and expands the abstract model of HCI knowledge representations shown in Figure 3.5, and explicitly shows a number of HCI research artifacts, with an emphasis on the mutual relationships of those artifacts in terms of (1) containment, and (2) mutual influence.

\textsuperscript{13}For example, in research-about-design and research-for-design. Also, certain forms of design research do not necessarily require the development of design artifacts beforehand. For example, Norman [270] in his book “The Design of Everyday Things” reflects on, comments, and evaluates the design of a number of commonplace, “everyday” artifacts as he encounters them in his immediate environment.

\textsuperscript{14}See also C. Alexander’s influential notion of “design patterns” [14] which, similarly to design guidelines and principles, facilitate and guide design process for solving complex problems in architectural and urban design.
Figure 3.8: Some bodies of knowledge (i.e., research artifacts) developed in the course of solving an IxD/HCI research problem. The design problem representation (i.e., the requirements document) and design solution representations co-evolve as an ensemble, and build upon the concepts found in the vocabulary. The co-evolution of the ensemble leads to the designer’s deep understanding of the target research problem, allowing him to synthesize “design guidelines”, or “principles”, shown at the top, as well as to fully describe the custom design development followed.
To summarize, as shown in Figure 3.8, HCI research artifacts and representations can be roughly organized into the following three subsets:

(i) *Low-level representations.* Includes foremostly domain vocabularies, the absolute basis and initial point for deriving, generating, and modifying any other research artifacts, thus also design research artifacts. Such vocabularies represent a conceptualization of the research domain in question, and can be regarded as an agreed-upon set of *constructs* for a domain, thus forming the “foundational” layer for any further research activity [200].

(ii) *Mid-level (design ensemble) representations.* Designing can be regarded as the activity of co-evolving the design problem and the set of associated design solutions\(^{15}\), as shown in the central part of Figure 3.5. A design ensemble can thus be represented by the pair of:

(a) *Design problem representations.* Any representation that captures the design problem, for example a set of design requirements (DRs)\(^{16}\).

(b) *Design solution representations.* Any design artifact which matches a set of problem representations\(^{17}\). For example, a design proposal that satisfies the corresponding set of design requirements.

(iii) *High-level representations.* From the maturation of the design ensemble, and acquired experience and knowledge, a designer-researcher can at some point synthesize higher-level (i.e., derived) bodies of knowledge (or *understanding* [3]), such as design principles or guidelines, the overall design development procedure followed in any particular instance of design research, lessons learned, and other research artifacts.

### 3.2.2.2 Research in HCI Design: An Activity-Centric View

Equivalently, in this section I further elaborate the general activity-centric model of HCI design research shown in Figure 3.6 which ultimately results in design history, thus providing a scaffold for both (1) concurrent (intertwined) research activities, and (2) subsequent research activities which commence when the design development process has been completed. Thus, when viewed through the lens of *activities* commonly encountered in HCI design research, I adopt the following definition by B. Archer [18]:

---

\(^{15}\) See also Section 2.6.3 on page 28 for a more detailed discussion on co-evolving the design ensemble, as well as the corresponding definition of ‘designing’.

\(^{16}\) See an overview of design requirements in Chapter 2, Section 2.5.3, page 22.

\(^{17}\) See an overview of design solutions in Chapter 2, Section 2.6, page 25.
**Definition 3.4.** *Design research. (activity)*

Design research is a systematic inquiry whose goal is knowledge of, or in, the embodiment of configuration, composition, structure, purpose, value, and meaning in man-made things and systems.

Design artifacts (including GUIs and interaction designs, the main focus of this work) “typically emerge through an extended process of development and redevelopment” [64] as well as “copy-and-modify” [57, p. 347]. I depict this process schematically in Figure 3.9, in which design development activities take place over the lifetime of a design project. As the process unfolds through time, it results in a number of *design artifacts* (that is, different revisions of such artifacts) being produced, refined, and released by the designer.

![Design Development Timeline](image-url)

**Figure 3.9:** In design, artifacts are developed iteratively, over a period of time. The global design development process periodically releases revisions of design artifacts.

However, in research, we are not interested in only coming up with new design artifacts; in research we are also interested in generating, validating, and systematizing new knowledge. This can be presented schematically as shown in Figure 3.10, where design activities are interwoven with research activities, and where design artifacts are now understood as research artifacts as well i.e., as a type of embodied knowledge or the thinking embodied in an artifact.
3.2. RESEARCH IN HCI DESIGN

**Figure 3.10:** In research through design, design development activities are interwoven with design research activities. In other words, we *research* as we simultaneously *develop* design solutions, and *other research artifacts*. Correspondingly, design artifacts are now also seen as a kind of “embodied knowledge” [357, 121].

Some characteristic activities in HCI design research include *identify needs and requirements*, *design*, *build or modify*, and *evaluate*. In HCI literature, such activities are usually arranged and depicted as closed loops (or “lifecycle models”), as shown in Figure 3.11.

**Figure 3.11:** An iterative model of HCI designing. Based on [328].

If I abstract the iterative loop from Figure 3.11 further, I end up with the following simplified yet still informative model of designing in Figure 3.12.
Next, March and Smith [238] further propose that the outputs generated in IT and HCI research may be subjected to any of the following four research activities: build, evaluate, theorize, and justify.

- **Build.** The activity of demonstrating the feasibility of an artifact performing a task\(^{18}\). Its main outputs are constructs, models, methods, and implementations. The main question is: *Does the artifact work?*

- **Evaluate.** This activity measures how well an artifact works using some criteria (e.g., metrics, benchmarks, criteria, expert opinion, and so on). The main question is: *How well does the artifact work?*

- **Theorize.** The main output of this activity are, as the name says, theories, as well as akin bodies of knowledge such as laws and models. The main question is: *Why does the artifact work?*

- **Justify.** The main output of this activity are the justifications (i.e., explanations and evidence gathered by collecting and analyzing data, or positing and proving theorems) for the theories developed in the ‘theorize’ activity. The main question is: *What is the evidence for theories?*

Note that the activities of build and evaluate are two activities that occur naturally in IT/HCI design, and indeed in design in general, as distinguished from sciences; within the Frayling’s classification, this modality of HCI design research clearly falls under RtD. The activities of theorizing and justifying are, on the other hand, indispensable and essential in RfD and RaD.

As another distinction of the HCI design development activities, an iteration can be considered to be either (1) a long one, and (2) a rapid one:

\(^{18}\)It appears that March and Smith [238] consider the activity of “design” as an integral part of the “build” activity.
3.2. RESEARCH IN HCI DESIGN

(a) *Long and formal* one (Figure 3.13, left), such as e.g., a formal evaluation study, perhaps involving a number of co-researchers, participants, and a substantial amount of equipment and/or system development, ranging over months or even years [332].

(b) *Rapid* one (Figure 3.13, right), i.e., a very simple and quick one (for instance, changing the way some data is visualized on screen, like e.g., substituting a list view with a thumbnail view), and carried out for the purposes of learning on the part of the HCI designer.

![Long evaluation cycle](image1)
![Short evaluation cycle](image2)

*Figure 3.13:* Left: long HCI evaluation cycle, featuring a formal evaluation study. Right: short (rapid) HCI evaluation cycle, in order to quickly obtain design feedback for the purposes of learning (unused parts of the large cycle greyed out for easier comparison).

During the design development (i.e., RtD), the designer-researcher:

(1) Deals with *design states*,

(2) *Transitions* from one design state to another (also called *copy-and-modify*),

(3) Makes *incremental* changes to the copied design state,

(4) Copies and modifies a design incrementally not only once but number of times, thus *iterating* or repeating such incremental modification of a prior design over a number of *iterations*,

(5) Transitions *forward*,

(6) *Backtracks* i.e., encounters dead-end or unpromising designs and then returns to an earlier design and continues from there, and

(7) *Merges* i.e., decides to produce a new design recombining elements of two or more prior designs.
These design development activities, when taken together, in general result in a rich structure called design history (Figure 3.14).

Figure 3.14: The design development activities of forward iteration, backtracking, and merging result in a structure that can be represented by a directed acyclic graph (DAG). Each node in this structure is actually a composite one, consisting approximately of the design-build-evaluate (DBE) activities (cf. Fig. 3.15). Furthermore, some nodes (indicated by black colour) feature a long evaluation test (cf. Fig. 3.13, left), while all the other nodes feature a rapid evaluation test (cf. Fig. 3.13, right).

Every node shown in a design history (Figures 3.14 and 3.6) is, in fact, a composite one, consisting of a number of contained activities that consume and produce various research artifacts. Thus, any such node can be seen as either:

(i) A “DBE” (Design-Build-Evaluate) sequence of activities, or simply as
(ii) A sequence of design states produced through a number of DBE iterations (Figure 3.15).

Figure 3.15: Two ways to further abstract and visualize the model of iterations in HCI design research. Left: the “activity-centric” sequence of DBE iterations, where each B activity (Build) ends in the creation of a design solution artifact (other research artifacts not shown for simplicity). Right: the sequence of design states produced through DBE iterations.

In general, an instance of solving a design problem can produce a design history ranging over a
large number of basic DBE (design-build-evaluate) iterations, and thus correspondingly over thou-
sands of revisions of associated design artifacts, such as design implementations, requirements, and design specifications. More specifically, this also means that a design history can contain thousands of instances of evaluations or tests of designs which, as I have had the opportunity to learn and experience during the development process, can be imagined as decision points at which a designer can make up his or her mind about future courses of actions, by answering any or all of the following questions:

(a) Should I continue with the present design?
(b) Or should I backtrack?
(c) Should I merge two existing designs?
(d) How should I modify or fulfill the design requirements, based on the evaluation test?

Another view of design evaluations, according to D. Schön’s theory of reflection-in-action [323], is as an opportunity to step back from the previously undertaken “design move” (made by acting on the previously “framed” design problem), and then reflect on such a move i.e., assess the effects and consequences of the move. Given, then, the supreme importance of design evaluations (which is not to diminish the importance of other design activities, such as design or build), and given that they occur within every DBE iteration in the design development and research process, in the next section I will go into some detail regarding the HCI evaluation methods.

### 3.2.3 Evaluating HCI Designs

As already mentioned above, depending on the outcome of the evaluation or the test of a partic-
ular design (or just a part of the design), the designer either chooses to continue developing the current design, or abandons it and returns to a previous design (i.e., the designer “backtracks”), or chooses to create a new design either from scratch or by merging two existing design revisions. Independently from the type of the test applied, however, every test will produce some feedback, whether in the form of the designer simply learning more about the design solution being developed, or in the form of extensive data which then has to be analyzed and studied, in order to extract findings in the case of long evaluations. Such findings are then written up in a report, and the findings/report are then used in turn to modify the design specification.

Thus, in a way, an evaluation method provides the designer with the means to discern among “good” and “bad” design alternatives [327]. An evaluation method helps the designer to decide which particular design (or a smaller or larger component of such a design) is viable and which is not; which part of the design should be retained and further improved, and which should be discarded (Figure 3.16).
Figure 3.16: Evaluation methods help the HCI researcher to choose among two or more design solution alternatives for a given design design requirement. In this example, design proposal C wins over design proposals A and B, because the evaluation method has indicated that the design proposal C is more suitable in some way for the design requirement in question, than either design proposals A and B.

Besides helping to choose among multiple design solution alternatives, evaluation methods can also attribute some fitness measure to a single design solution; for example, if the evaluation method chosen is “expert feedback”, a group of experts testing a prototype could deem a feature to either be “acceptable”, or “very effective”, or “unnecessary”. In this sense, the evaluation is deemed to be “formative” because evaluation results are then fed back into the next round of designing in the subsequent DBE iteration.

3.2.3.1 HCI Evaluation Methods

No universal agreement on how to evaluate HCI research artifacts exists; in other words, a plethora of different methods exists and one should approach the selection of an appropriate method with care. For example, Fallman and Stolterman [114] warn about the current “disciplinary anxiety” in the field of interaction design, due to organizations not being in possession of research tradition, nor “experienced supervisors to build and guide students and inexperienced researchers”. Furthermore, available publication venues “allow for substantial latitude in terms of acknowledged research methods and approaches, ranging from quantitative empirical lab studies and qualitative observational studies to research-through-design and critical design experiments”, thus frequently leading to confusion or anxiety regarding the selection of appropriate research methods, for a given HCI design problem.

In order to select the most appropriate HCI design evaluation methods for this work, I first decided to obtain a high-level, global view of available methods. For example, Sharp et al [328] provide a classification of evaluation approaches into three major categories (analytical, empirical, and hybrid), as follows.
Analytical Evaluation Methods These methods do not necessarily require end users in order for the HCI design to be tested, i.e., they can be performed by the HCI designer himself, and are thus usually quick to perform. Major subclasses include:

1. *Inspection-based methods.* Inspections are good for evaluating full systems, and are further classified into
   
   (a) *Walkthrough methods.* Here designers and experts walk through scenarios using a prototype, or a sequence of screen mockups. Two major kinds include
      
      i. *Cognitive walkthroughs.* Evaluates ease of learning, or *exploratory learning*, for novice users.
      
      ii. *Pluralistic walkthroughs.* Involve users, developers, and usability specialists.
   
   (b) *Heuristic evaluation (HE) methods.* These methods use heuristic rules of thumb, principles, guidelines, and platform standards for evaluating the fitness of a HCI design. Besides being quick and cheap, another advantage of HE is that it applies to users that are neither novices nor experts, which is predominantly the case in real-life situations.

2. *Theoretical predictive models.* Predictive models predict performance of expert users, for specific parts of the system. Such models include e.g. GOMS (Goals, Operators, Methods, Selection Rules), as well as its concrete instantiation KLM (Keystroke-Level Model) containing standard times for basic operators such as clicking a mouse button or pressing a key. Disadvantages of these models include highly limited scope (they support only small set of highly routine, short, and well-defined tasks), usefulness for determining expert performance only, and does not allow for modelling distractions usually encountered in everyday life and work, thus leading to unreliable predictions. As a consequence, GOMS is not used often for evaluation nowadays. Another predictive model includes Fitts’ Law [119].

Empirical Evaluation Methods These methods require collection of observational data. Major kinds include:

1. *Usability testing methods.* At the peak of popularity in the 1980s, however still being used nowadays. These methods are appropriate for later stages of design, and they measure user’s performance while executing a task (e.g., speed, error rates). Data is commonly collected through system logging, in a controlled environment.

2. *Field studies.* Done in ecologically valid settings, with data collection methods such as interviews and user observation. Typical data collected include notes, questionnaires, and audio or video recordings.
3. Case studies. This is a method that can be considered to be primarily a qualitative [304] or sometimes a mixed [364] research method, in order to provide insight into a particular instance of a research problem in rich detail.

Hybrid Evaluation Methods  This is some combination of analytical and empirical evaluation methods, combined into an overall, custom research approach. For example, several experts could be asked to execute cognitive walkthroughs (analytical) while trying out an interactive prototype in a controlled environment and being video-recorded; the interviews could later on be transcribed, and transcripts coded into categories in order to discover regularities within such transcripts i.e., within the feedback provided by the experts.

Discussion. Of the three main categories of HCI design evaluation methods (analytical, empirical, and hybrid) described above, I will note that I used walkthrough tests for rapid evaluation of low-fidelity and high-fidelity prototypes and mockups developed. In other words, I underwent rapid evaluation cycles as shown in Figure 3.13 (right) in order to obtain quick feedback on the prototypes and mockups being developed, as well as to learn more about the problem-solution ensemble.

In addition, I evaluated two higher-fidelity prototype revisions (CAMBRIA 1 and 2) using long, formal evaluation cycles illustrated in Figure 3.13 (left). I evaluated CAMBRIA 1 using the hybrid evaluation method described in the paragraph above, i.e., by interviewing several experts performing cognitive walkthroughs with the prototype, and video-recording their “talk-aloud” statements. The interviews were later transcribed and coded into categories. I chose this particular (hybrid) evaluation method because it offers means to obtain valuable critical assessments or suggestions for improvement from participants with specialized knowledge or skills pertinent to the prototype in question, and then being able to feed such assessments back into the design development process.

I evaluated the second prototype (CAMBRIA 2) by conducting a case study, which can be considered an empirical evaluation method. I selected this particular method because it offers an analysis of a particular use situation in rich detail, thus contributing to further improvement, refinement, and elaboration of subsequent research artifacts, such as for example design guidelines and principles.

3.2.4 Stopping Point: When is an HCI Research Problem Considered ‘Solved’?

How can we know that we have reached the “stopping point” in HCI development and research? When do we know that we have succeeded in designing a “good” HCI solution, and that we have “solved” the HCI research problem?
Reminding ourselves about the nature of design, non-routine design problems have no stopping rule [307]. Further, the associated sets of potential, “good enough” design solutions are considered to be innumerable [307, 231]. Thus, in theory, one could be traversing the space of “good” HCI design states (for a given HCI problem) in perpetuity. We can reasonably assume that each evaluation study (within the design history as shown in e.g., Figure 3.14) would keep a continuously-developed HCI design within the bounds of such a space of “good” designs; such an evaluation, done during the design process, is also commonly called a formative evaluation [328, p. 589].

Of course, the ultimate evidence that any given HCI design is, indeed, “good” and “successful” is when it has achieved wide adoption by its intended user community. However, such adoption frequently takes place over a prolonged period of time, measured in years or even tens of years. Clearly, to judge the quality of HCI research, including the HCI research I describe in this thesis, we are forced to employ some intermediate measures of “goodness” (stronger than basic evaluations, and necessarily weaker than the definitive acceptance by an user community). Such intermediary evaluations, which should be done with real users and real tasks [62, 327], are then usually called “summative” evaluations [328, p. 589], “validation” studies, or “testing in the wild” (in further text, I will call such methods “validation methods”).

Validation methods in HCI design and related fields include case studies [304, 364], field studies [366] and field experiments [62], in-situ interviews [420], and observational studies [365]. Further, Zimmerman et al [419] proposed a novel validation method or approach according to which a particular HCI design can be considered successful if it features the following four ingredients, or characteristics:

(1) **Significant invention.** A quality HCI design should be a novel, important, and well-reasoned recombination of existing techniques, elements, and approaches.

(2) **Design process description.** An instance of a successful HCI design research should be accompanied with a full description of the design process followed by the researcher.

(3) **High relevance.** A HCI design solution should also exhibit “high relevance”, which is demonstrated by detailing how the solution (1) affects the real world, (2) what is the preferred state that the HCI design should ultimately converge to, and (3) provide an explanation as of why the research community should consider this state to be preferred.

(4) **High extensibility.** Exhibited and demonstrated “high extensibility” i.e., that the HCI design is generalizable and applicable in other domains, and adaptable for different uses.

In addition, Munzner [257] claims that a quality of a contribution in the field of information visualization should be judged by (1) how well are visual encodings and interaction mechanisms
justified, in terms of how well they fulfill the design requirements, and that (2) such justifications should cogently refer to the first (i.e., perceptual) principles, as well as to currently prevalent theories in the field. Taken together, then, both the Munzner's and Zimmerman's criteria can be considered to make up an effective roadmap for what constitutes successful HCI research; in this work, I strive to meet such criteria, and anticipate for the work described in this dissertation to represent a significant invention, provide a process description, exhibit high relevance, and high extensibility; further, I strive to justify design decisions, and refer to the first principles when making such decisions.

3.3 Research Approach Adopted for This Thesis

Shortly, the research approach adopted in this thesis is an instance of the general design-based research process described in all the preceding sections of this chapter.

As per Research Problem Statement postulated in Chapter 1, single-state systems, by their very design, do not natively support interaction with multiple design alternatives in parallel, and are thus exhibiting definite limitations regarding this particular aspect of design work. Merely enhancing current, single-state tools cannot lead to a tool design which successfully resolves the situation of concern\(^{19}\), nor fulfills the research intent\(^{20}\) of this thesis. The fact that merely enhancing existing, single-state pCAD tools cannot lead to a successful tool design, implies that this thesis calls for HCI research concerned with innovative or exploratory design. Exploratory design, in turn, implies that we are dealing with original artifacts whose design requirements must be developed completely anew, and “must be defined largely from first principles” [268, p. 158]. This, in turn, means that the design problem and its set of associated design solutions must be co-evolved from scratch.

In exploratory design, one does not seek to improve on the existing task structure, as is frequently the case in HCI design, and as commonly recommended in reference texts. Instead, because the current task structure comprises an ineffectual mixture of ad hoc methods using single-state systems, I see to actually replace such mixture with a new task structure, through the development of new design solution artifacts (thus being involved with what can be best described by Frayling’s notion of RtD [121] or research-through-development). Only after a number of instantiations (prototypes, release-quality implementations, and so on) have been developed, perhaps over the course of several years, in other contexts and with other researchers, will other forms of research methods, such as usability studies, be applicable and suitable for generating new knowledge in the selected target domain.

Exploratory designing is to be contrasted with non-explorative designing (i.e., enhancement

---

\(^{19}\)Section 1.2.1.

\(^{20}\)Section 1.2.3.
of existing designs), whose functional requirements as well as existing usability levels (speed, error rates, and so on) are relatively easy to identify. The requirement to conduct exploratory design also implies that one cannot employ late-stage evaluation techniques (such as GOMS, or controlled quantitative studies) in order to assess the feasibility of new designs; indeed, such late-stage techniques require relatively mature design artifacts which by definition cannot exist in early-stage design, nor is there the baseline to compare the new tasks (enabled by the novel computational tool) against. Furthermore, using late-stage evaluation techniques have the potential to cull promising lines of design; “premature usability evaluation of early designs can eliminate promising ideas or the pursuit of multiple competing ideas”[141].

Thus, from the very beginnings of the research described in this dissertation, I was predominantly involved with the development of a good number of different design artifacts, whether those were simple HCI sketches, storyboards, wireframe GUI mockups, and prototypes of various levels of fidelity (low-fi, and hi-fi), culminating in the design and development of the two higher-fidelity prototypes. Therefore, and especially during the early phases of research, I remained within the limits of what Frayling calls research-through-development.

However, as soon as the stream of continuously-developed design solution artifacts allowed for an initial understanding of what is possible and desirable, I was able to gradually transition into a more reflective (introspective, rational) mode of thinking (overt of covert, conscious or subconscious) related to the generation of the following two research artifacts: (a) interaction design guidelines and principles for building alternatives-enabled pCAD systems, and (b) the custom design development method utilized throughout this research.

As I will further indicate in Section 3.3.1, reflective activities were intensifying as a function of the number and the refinement level of research artifacts developed throughout the development; in a way, then, such reflective activities can be considered to build “atop” research-through-development. In Frayling’s terms, the experience and knowledge acquired during RtD has allowed me to gradually become involved with RfD and, to a degree, RaD activities. Therefore, although constituting a smaller part of the design-based research described in this dissertation, I also had the opportunity to conduct research “about” designing as well (RaD), through the following activities: (a) observing how participants go about designing, when given the chance to use both prototypes developed in this work, and (b) self-observing myself as I conduct the studies, and perform the activity of designing all the artifacts developed in this work.
3.3.1 Phases of Research Performed

The overall research described in this dissertation was roughly divided into six consecutive\(^{21}\) phases shown in Figure 3.17.

![Diagram showing six phases of research](image)

**Figure 3.17:** An overview of the research described in this dissertation, divided into five consecutive phases.

The overall *development process* (i.e., phases A–E) seemed to loosely follow the UK Design Council’s process model of designing [51] which consists of the following two major stages, each divided into phases (see below):

1. *Problem discovery stage.* This stage was marked by attempts to better understand and define the research problem, and includes the Phase A and Phase B of research (see below).

2. *Design solution stage.* This stage was marked by the development and evaluation of higher-fidelity prototypes (Phases C and D of research, see below). I note here that the “deliver” part was missing from the research described in this thesis\(^ {22}\) and was instead supplanted by Phase E of research (consolidation and reflection).

Finally, in addition to detailed descriptions in the chapters that follow, I will now succinctly describe the work done in each phase of research:

---

\(^{21}\)This is not to be confused with the waterfall model of designing, because each phase contains multiple continuous developmental iterations of multiple research outputs; furthermore, some research activities extended over, and were being performed continuously, throughout the research, such as e.g., continuous updating of the literature review, and of the design requirements.

\(^{22}\)This thesis is not concerned with the delivery of a commercially available product.
3.3. RESEARCH APPROACH ADOPTED FOR THIS THESIS

Figure 3.18: The “double-diamond” model of design as proposed by the British Design Council. Based on [51].

Phase A: Pre-Design This phase was characterized by the creative exploration of the space of possible design solutions for the problem of alternatives-enabled computational tools, as well as by opportunistic “switching” among several iterative and incremental activities, such as requirements identification and refinement, and iterative creation of several types of early-stage design “solutions” (GUI and interaction sketches, storyboards, GUI wireframe mockups, and low-fidelity GUI throwaway prototypes) which together collectively made up a “design specification” DS-A at the end of the phase (Figure 3.17).

Phase B: Conceptual Model This phase was characterized by my attempts to define the “conceptual model” of the computational pCAD tool under development. A conceptual model is a very high-level, abstract representation of the concepts (constructs, entities) present in the solution, as well as their relationships, and should [177] at a minimum include (1) metaphors (or analogies) used, (2) concepts, and (3) the relationships among concepts. In this work, the conceptual model captures the notion of alternatives, collections thereof, as well as tasks related to managing and manipulating such models. This phase was also marked by the beginning of research-through-development in its earnest, as well as the onset of research-for-design activities through thinking, contemplating, and reflecting on what would constitute effective design guidelines for alternatives-enabled computational tools.

Phase C: CAMBRIA 1 Prototype This phase was characterized by the development and testing of the first higher-fidelity working prototype, named CAMBRIA 1, as well as with the further refinement of design requirements, and the design specification. During the development, I developed a fair number of different versions, culminating in a formative evaluation study of the prototype, involving six CAD and HCI expert participants. The study resulted in six video-recorded interviews of participants, each about 1.5 hours long; the video recordings were then transcribed by the author using a commercial coding package. The transcripts
were analyzed, coded into categories. Findings were written up and published [194, 409, 410, 191].

**Phase D: CAMBRIA 2 Prototype** The phase which ended in the development of another, second higher-fidelity prototype (CAMBRIA 2), its validation, as well as in further refinement of design requirements, and the design specification. This phase also marked the end of RtD activities, since designing and building novel instantiations ceased at the end of this phase. The method chosen to evaluate the CAMBRIA 2 prototype was the *case study* method, or a method that can be considered to be primarily a *qualitative* [304] or sometimes a *mixed* [364] research method, in order to provide insight into a particular instance of a research problem in rich detail.

**Phase E: Consolidation** General consolidation and reflection phase, which resulted in the appraisal of the total work done up to that point, and in the postulation of a *taxonomy of interaction primitives* as well as *interaction design guidelines and principles*, which will assist HCI design practitioners and researchers to approach, frame, specify, and evaluate future alternatives-enabled pCAD tools. This phase was characterized by contemplation, introspection, integration, and overall sense-making activities.

I will note here that the phases listed above are, of course, *approximate* phases of research arranged in chronological order, in the sense that they are intertwined with continuous and concurrent activities of uninterrupted sense-making of the entire problem complex, as well as constant refinement of relevant bodies of knowledge, such as the domain vocabulary, and the related work (bibliography).

### 3.4 Chapter Summary

In this Chapter, I first presented an overview of the nature of research as a systematic quest for knowledge, in order to provide a basis for comparing traditional research with research in design-based fields of inquiry, such as HCI and IxD. In this sense, I first provided an overview of the types of knowledge produced in traditional research, as well as a concise description the two major traditional research meta-methods (the reflective method, and the scientific method), and explained why these methods are of limited power and applicability in design, as well as why the definition of “knowledge” as understood in traditional sense has to be augmented in order to encompass the artifacts typically developed in design as well.

Having this general basis to think about, and understand design-based research, I then described two ways to view such research: a *knowledge-based view* (expressed in terms of research artifacts generated in design research, as well as the mutual relationships of such artifacts), and an
activity-based view (expressed in terms of activities commonly found in design-based research). I continued with a discussion on the topic of evaluation in HCI research, how such evaluations can be understood as decision points in the process of designing, and how to judge the overall quality and success of HCI research. Finally, the last section of this chapter described the research approach that I adopted in this thesis, a research-through-design approach combined with the methods of expert study and case study, immediately followed and concluded by research-for-design activities, reflection, and integration.
Chapter 4

Background: Many Design Alternatives at Once

Do designers work with multiple design solutions at a time? Or do they fixate on one particular design solution, and then hone it incrementally until it satisfies the requirements and constraints of its associated design problem?

In this chapter, I will delve into this issue, and investigate whether designers indeed do, or do not, work with multiple design alternatives “in the wild”. That way, I will be providing the overall background picture, as well as a part of inspiration, for the work described in this thesis. And as the reader will soon find out, ample evidence exists that some (or many, or most) designers indeed tend to work with multiple, simultaneously-available design alternatives. I will provide examples, cite empirical studies, and describe expert opinion lending support to this claim, as well as provide an overview of what it means to “fixate” on a single design solution instead.

There are multiple ways to examine the issue of multiplicity of design solutions, and related questions. For instance, one might suppose that there is some optimal, eigen-number of multiple, simultaneously available design solutions preferred by a designer, similarly to the “magical” number seven of items (or chunks) that an average human can retain in his or her short-term memory [250]. One might also conjecture that such a number $m$ lies within some interval, ranging from 1 (thus indicating that this particular designer prefers to work with a single design), to some as of yet unknown number $m_{\text{max}}$ (for instance, an hypothetical designer who prefers to, somehow, work with hundreds of designs in parallel, perhaps following some highly customized and specific design procedure, or simply using a novel computational tool). Such a number $m$ might additionally be dependent on the designer’s personal preferences and work style, or on the nature of design problem at hand, or on the current stage in the design process (conceptual, early, or later design stages?), or even on the design domain (architectural, mechanical, or product design?). There are, surely, other dimensions along which one might consider and determine this number $m$. 
To examine this matter further, and support the claim that (most, many, or just some) designers indeed do prefer or tend to work with multiple design alternatives, in this Chapter I reference:

(i) **Artifacts and projects from real-world practice utilizing alternatives.** I furnish several illustrative and representative instances of multiple design alternatives being used “in the wild” i.e., in architectural design (§ 4.1.1), product design (§ 4.1.2), fine art (§ 4.1.3), and industrial design (§ 4.1.4),

(ii) **Existing scholarly studies.** I reference existing academic studies that describe and report instances of designing involving multiple design solutions (§ 4.1.5),

(iii) **Expert statements.** I quote experts from diverse fields stating the need for generating and evaluating multiple design alternatives (§ 4.1.6),

(iv) **Theoretical justification.** I explain the designers’ need for working with multiple design alternatives by applying theoretical accounts of design (§ 4.1.7), and

(v) **Ambiguity.** I interpret the phenomenon of multiple design alternatives in terms of ambiguity and other types of indeterminacy, which thus naturally leads to concurrency in design (§ 4.1.8).

In addition to the above, and in order to provide a more balanced picture of relevant issues, in § 4.2 I will review claims that some designers actually prefer to “fixate” on a single design solution and then fine-tune such a solution until it is deemed satisfactory. Finally, I discuss possible ways to explain and reconcile this apparent contradiction in § 4.3.

### 4.1 Evidence for Concurrency in Design

Examples from real-world art and design practices demonstrating the use of multiple design variants are plentiful and easy to find. In this section, § 4.1, I showcase several such representative examples from different areas of design, such as architectural design, product design, fine arts, and industrial design. I further reference expert opinion, and provide a theoretical justification as of why designers would want, or need, to work with multiple design alternatives in parallel. Before I do that, however, I provide the following definition:

**Definition 4.1. Design multiplicity.**

Design multiplicity is the count \( m \) of design alternatives that a designer is simultaneously viewing, or comparing, or editing, at a certain point of time.
The following examples and considerations, then, cover the cases where design multiplicity \( m \) happens to be greater than one. In Section 4.2 I will explore the case when designers fixate on a single design, i.e., the case of \( m \) being equal to one.

### 4.1.1 Alternatives in Architectural Design

In Chapter 1, Figure 1.3, I have already provided one example of a collection of multiple massing models of a building designed by Frank Gehry, and constructed in 2010 at 8 Spruce Street, in New York City.

As another example demonstrating the use of multiple design alternatives, Figure 4.1 shows nine alternative models of the “Elephant House” (Figure 4.2) designed by the architectural firm Forster + Partners, and constructed from 2002–2008 at Copenhagen Zoo, Copenhagen, Denmark.

![Figure 4.1: Example of multiple design alternatives used “in the wild” in architectural design: “Elephant House” by Brady Peters / Foster + Partners. Source: Brady Peters [48].](image)

Traditional materials, such as wood, corrugated fiberboard, and acrylic, were used to build this collection of solid models or maquettes. The designer can walk around the table stocked with such models, view them, compare them, and evaluate them according to different criteria, such as for example the overall aesthetic impression, the sense of proportion, and their suitability for the environmental context. Based on such comparison and evaluation, the designer is in position
4.1. EVIDENCE FOR CONCURRENCY IN DESIGN

4.1.2 Alternatives in Product Design

As another representative example, this time from product design, Figure 4.3(a) shows around thirty hand-drawn sketches (i.e., different design alternatives) by Philippe Starck, an expert product designer, produced in the course of developing the “Juicy Salif” citrus squeezer design shown in Figure 4.3(b).

According to Starck's own statements, he produced the final design while waiting for a meal in a restaurant: once the food arrived, it had a a baby squid in it. “This vision of a squid-like lemon came upon me, so I started sketching it” [87, p. 7]. The designer began drawing his sketches in the lower right corner of the paper mat, and made his way counterclockwise on the mat, finally ending up with drawings located in the lower left corner [217]. The sequence of sketches in this arc shows Starck's initially unsuccessful attempts to combine a lemon with a squid, which ended with a (this time successful) merging of both the lemon and the squid with a third concept: the concept of a 1960s spaceship. One may thus write:

\[
\text{citrus squeezer} = \text{lemon} + \text{squid} + 1960's \text{ rocket.}
\]

Therefore, in this example, the designer arrived at the final product design by working his way through a sequence of around thirty simultaneously visible, closely related, mutually-derived, and cross-pollinated ideas or designs (the lemon, the squid, and the 60's rocket/spaceship), sketched rapidly on a simple, paper table mat. The designer thus devised the final design while being able
to view and compare a number of alternatives at once, and then recombine elements of several promising designs into the final (and commercially successful i.e., widely accepted, popular) product design.

As another example of alternatives in product design, the reader may refer back to Figure 1.2 on page 2, showing eighteen related yet different designs of in-ear headphones, also known colloquially as “ear buds”, arranged into a rectilinear grid.

### 4.1.3 Alternatives in Fine Arts

As another example, this time from fine arts, Fig. 4.4 shows a sketch study “Study for the Trivulzio Equestrian Monument”, made by Leonardo da Vinci in 1508-10, with a total of four different revisions of the sketch, all drawn by the artist on a single sheet of paper [213].

As can be observed on the sketch, its different parts feature multiple overlaid alternative outlines, with differing positions and orientations (cf. for example horse’s legs, or the rider’s head). This is a good example of how a sketch can be suggestive and ambiguous [404, Ch. 5], i.e., lending itself to multiple possible interpretations, and with multiple final design solutions. I will add that sketches are also quick to make, inexpensive, and disposable (§ 2.6.1), thus allowing for easy and rapid generation of alternatives, as demonstrated in this drawing.
4.1. EVIDENCE FOR CONCURRENCY IN DESIGN

4.1.4 Alternatives in Industrial Design

Toyota Motor Corp., which is at the time of this writing the world’s largest and by some measures the most successful [132] car manufacturer, employs an approach to car design that has been the subject of a number of academic studies in the past, e.g., [396, 348]. In those studies, the authors investigated what makes Toyota’s industrial design process, despite its apparent complexity, capable to “make better cars more quickly and cheaply”, when compared to other large car manufacturing companies.

In the conventional approach to car design, the designer (or the design team) moves through a series of steps, from one point to another, following the so-called point-based model of designing [396]. Figure 4.5 depicts such a process in a simplified, abstracted fashion.

![Figure 4.5: Point-based approach to car design. Based on [396].](image-url)
Toyota Corp., on the other hand, follows the so-called set-based (or "concurrent engineering") model of designing (Figure 4.6), which is characterized by the availability of not one but a whole set of design solutions at various stages in the design process, both at the whole car level, and at the subsystem level [396].

![Figure 4.6: Set-based, concurrent approach to car design is based on the gradual narrowing of entire sets of possible (i.e., alternative, potential) design solutions. Based on [396].](image)

Therefore, instead of focusing on just one or two prototypes as is the case in point-based approaches to design, Toyota Corp. develops and “filters down” entire sets of the same i.e., applying the concurrent, progressively “set-narrowing” approach to designing. Additionally, in the early, prototyping phase of car design, Toyota’s designers prepare comparatively many more clay-based physical prototypes than their competitors, which are then further filtered into partial prototypes, to be combined into a set of full, final-stage prototypes. Toyota also allows for significant early-stage design ambiguity expressed in terms of comparatively large tolerances of part dimensions, thus providing higher levels of flexibility, as well as the opportunity to defer decisions for as long as possible regarding the selection of appropriate components for the final car design [396].

### 4.1.5 Evidence from Empirical Studies

According to a body of peer-reviewed empirical studies, some (or most) designers avoid premature commitment to specific design variants i.e., they develop and work with multiple design solutions at once.

To begin with, designers are in general solution-focused, instead of problem-focused [85, 108]. In other words, designers tend to generate and then evaluate a specific design proposal, instead of focusing on problem analysis. From this perspective, then, computational tools that handle multiple potential design solutions might potentially help and boost the design process, because such tools would mirror, and support, designers’ natural tendency to be solution-oriented i.e., to
generate multiple design solution proposals in order to better understand the design problem at hand.

Next, in a study by Ömer [11], expert architectural designers generated, on the average, 4.13 design solution proposals (versus 1.83 by students of architecture, and 1.253 by complete novices). The author claimed that architects, as distinguished from other designers such as mechanical engineering designers, generate comparatively many more design alternatives due to the supposedly more open-ended (i.e., “creative”) nature of architectural design problems. The author conjectured that architects naturally tend to generate multiple design solutions due to their training and design habits, which in turn rewards creative design strategies, in contrast with electrical engineering where engineers frequently employ “routine design”, thus not having to generate more than one or two design alternatives. In another study by the same author [9], architects were found to develop multiple design solutions to constrained, closed design problems. In yet another study [10], architectural designers kept searching for alternatives, even if they have already found a satisfactory conceptual design.

In [208], advanced students of architecture proceeded to generate entire sequences of design solutions until one proved acceptable. In [347] the authors, in order to study iteration processes within the context of team-oriented engineering design, asked the participants grouped into nine different teams, to design a two-dimensional structure. The findings indicate that three out of nine teams reconsidered early design (conceptual) design solutions i.e., developed multiple alternative conceptual designs. In [26] the authors studied the design processes of freshman and senior engineering students, and found out that both the freshmen and senior students spent most of their time (79% and 76%, respectively) in the “developing alternative solutions” activity, out of which significant amounts of time (56% and 54%) were spent to actually model those alternatives. The number of possible alternative solutions considered was 11.2 and 11.5, respectively. The experiment also demonstrated that the participants who considered a larger number of design alternatives produced a higher-quality final design.

In another study on how designers create graphic online advertisements [103], it was found that parallel prototyping, as opposed to serial prototyping, leads to better design results, more divergence, and increased self-efficacy\(^1\). In [267], the authors observed and interviewed eleven professional website designers, and found that they followed a design process characterized by the generation of multiple design alternatives, using multiple representations. Said designers utilized various, heterogeneous media and techniques such as pen and paper (i.e., hand-drawn sketches), but also word processing software, presentation software, as well as vector and raster image editors in order to come up with multiple, “rough” ideas during the design exploration phase\(^2\). Based

---

\(^1\)Self-efficacy is defined as “beliefs in one’s capabilities to organize and execute the courses of action required to produce given attainments” [30].

\(^2\)One particular sub-type of website designers, that of website GUI designers, primarily utilized sketching during the
on their findings, the authors suggested the following desirable features of novel computational tools aiding in design: (1) use of an informal, sketch-like user interface, (2) support for multiple representations, (3) focus on early design phases, (4) integration with other tools, and (5) management of history and variations. In [368], Terry and Mynatt studied expert users of raster image editors, and found that such users regularly develop alternatives (or “variations”) in their work. In a quantitative study involving 48 participants [374], Tohidi et al use paper HCI prototyping and establish that developing and testing many interfaces is better than developing and testing just one, in the sense of being able to obtain ratings of HCI designs that are less prone to inflation, as well as obtain useful (i.e., critical, calling attention to flaws and errors) comments that help the designers to identify issues and shortcomings associated with a particular HCI design. The authors found that having more design alternatives did not lead to more useful suggestions for HCI design improvement; however, the number of superficial suggestions decreased³.

4.1.6 Expert Opinion

As another form of obtaining evidence for the claim that designers do (or should) work with multiple design alternatives, in this section I quote expert opinion, i.e., personal beliefs affirming that one should indeed work with multiple design alternatives. Expert knowledge and opinion is considered to be a valid way to justify beliefs; for instance, C. S. Peirce [283] claimed that referencing an authority on knowledge is one of the well-grounded ways to “fixate one’s beliefs”.

For instance, Linus Pauling, the famous American quantum chemist and biochemist, stated that “if you want to have good ideas you must have many ideas. Most of them will be wrong, and what you have to learn is which ones to throw away”[82]. Pauling’s statement thus implies that not only it is important to have multiple ideas, but one also has to know how to evaluate them for their worth, and then choose the best ideas. To dissect the statement above further, Pauling apparently implies the cognitive activities of generate, evaluate, and choose pertaining to a set of “ideas”.

Next, according to J. C. Jones, the author of the influential “Design Methods” book [179], three types of expert work activities exist: divergence (generation of alternatives in order to find out limits of the problem), transformation (re-combination of partial solutions), and convergence (fine-tuning the design solution); the activity of divergence then supports the claim for multiple design alternatives. Cross [87, p. 21] states that “consideration of alternative solution concepts might save time and effort in the long run”. When considering multiple partial solutions as “partial design alternatives”, Gordon Murray, an expert designer, apparently confirms the Jones’ transformation activity (of recombining partial solutions), by describing his design process in terms of “bits” as

³Therefore, having multiple design options appears to help alleviate somewhat one of the shortcomings of prototyping, that of superficial comments, as mentioned in Chapter 2, Section 2.6.2.1 on page 27.
4.1. EVIDENCE FOR CONCURRENCY IN DESIGN

well as in terms of selection ("throwing bits away") [87, p. 52]:

“I do a quick sketch of the whole idea, and then if there’s one bit that looks good, instead of rubbing other bits out, I’d put that bit to one side; I’d do it again and expand on the good bit, and drop out the bad bit, and keep doing it, doing it; and end up with all these sketches, and eventually you end up throwing ninety percent of these away.”

In the foreword to [328], J. Foley states that in the process of user interface design, one should “consider alternatives”. The authors further state [328, p. 47] that, when trying to understand the problem space of designing an interactive product, that “…spending time enumerating and reflecting upon ideas during the early stages of the design process enables more options and possibilities to be considered”. Further, “generating alternatives is a key principle in most design disciplines, and one that should be encouraged in interaction design” [328, p. 417].

In [151], Hartmann et al enumerate four benefits of developing multiple design prototypes: (1) a better understanding of the design space, (2) answering multiple “what-if” questions, thus assisting with organizational decision making, (3) better communication of design requirements, and (4) a better understanding of design trade-offs, thus enabling critical feedback.

In [327], Sedlmair et al discuss how to proceed with design studies i.e., with research projects concerned with the design of information visualization computational tools, and claim that “in the design stage it helps to broaden the consideration space of possible solutions, and to select good solutions over bad ones”, and that “a common pitfall is to consider too few solutions and to prematurely commit to a selected solution”. Researchers should strive to have “a broad consideration space of possible solutions” (i.e., the set of valid design solutions), in order to later be able to filter down this considerations space further, and come up with “a narrow proposal space”, which should then be showcased to domain experts, in the form of mockups, or prototypes, to finally select the best design proposal which will then be implemented as a high-fidelity prototype or as a release-quality application. This model, again, appears to closely mirror the Jones’s diverge-transform-converge model of designing [179], as well as Toyota Corp’s concurrent, progressively “set-narrowing” approach to car design [396, 348]. The authors conclude that a small consideration space might lead to a higher probability of “only considering OK solutions, and missing a good one”.

In another study, also concerned with the design of information visualization tools, van Wijk [388] claimed that design alternatives can easily be generated, but that “it is much harder to decide which choice is optimal”, and that the developers of new visualization methods “should be aware of alternative solutions and carefully study their advantages and limitations”. Furthermore, in the basic design cycle a researcher should “first, set up requirements; second, invent a number of solutions; and, finally, match the solutions against the requirements and select the best one”.
In conclusion to this brief synopsis of expert opinion, it appears that there is an overall agreement on the importance of generating, and working with, multiple design alternatives. Further research would certainly uncover more, similar statements, in support of alternatives-enabled work.

### 4.1.7 Theoretical Justification for Multiple Design Alternatives

How can the designers’ need to see, compare, and recombine many alternatives be explained using theoretical accounts of design?

Design problems are incomplete, ambiguous, and lack definitive formulations [299, 300, 307, 338, 146]. Due to this lack of a clear, unambiguous problem definition, a pragmatic approach to exploring the design problem space is to generate multiple design solutions, thus enabling the designer to acquire a better appraisal of what is needed, desired, and sought after in an adequate design. For instance, Cross [85] claims that designers use design solution proposals “as a way of exploring and defining problem-and-solution together”. Likewise, Hillier et al [159] suggest that designers often “conjecture” a solution by proposing a design, in order to learn more about the design space. One might thus safely assume that the more design solutions a designer generates and evaluates, the better the designer’s understanding of the design problem at hand becomes. Jones [179] further confirms that the activity of “divergence”, within his three-stage “divergence-transformation-convergence” model, is the process of generation of alternatives in order to “find out limits of the problem”.

Next, since there is no objective and clear criteria [307, 146, 231] to evaluate design solutions, any given single solution will be judged as either “good” or “bad” [307] by the designer, or whether it “satisfices” [264] the problem. Having an entire collection of design alternatives enables the designer to obtain a better idea of the fitness or “goodness” of any particular alternative, as compared to having a single, isolated design solution, due to being able to arrange such alternatives into an ordered set, according to some criterion or a set of criteria.

Another common task in design is the recombination of partial solutions, or transformation [179] of design solutions. Having multiple such (partial or whole) design alternatives in the field of view facilitates such a recombination, by eliminating the relatively slow cognitive process of memory retrieval and storage to and from short- and long-term memories, and replacing them with the faster visual scan process [409], thus combating the “limited transfer problem” [103]. This way, the retrieve-recombine-store process typical of single-state system use, gets replaced by the faster scan-recombine process (Figure 4.7).

---

4This raises the interesting question of diminishing returns as the number of design alternatives increases. Does the understanding of a design problem decrease marginally after some number n of design alternatives have been generated? In addition, how does one go about generating such alternatives, in order to obtain a representative sample of the underlying design space?
4.1. EVIDENCE FOR CONCURRENcy IN DESIGN

Figure 4.7: Having multiple design alternatives in the field of view facilitates their comparison and recombination. The retrieve-recombine-store process (left), where the designer retrieves alternatives one by one and is forced to remember the details of previous alternatives in the retrieve-store sequence, gets replaced by the faster scan-recombine process (right) in which the designer can readily pinpoint, select, and evaluate individual parts, and recombine them into a new alternative state.

Finally, I should mention the (as per Figure 4.7 evident) process of visual comparison which “can reveal common structure and combine partial structures and thus promote transfer” [128], and which helps users to direct their attention to “key relations” [129] among the constituent elements of design states. In addition, vision is the sense with the highest bandwidth of all: we “acquire more information through vision than through all of the other senses combined” [398, Ch. 1]. Visual comparison thus reveals the components of information: patterns, repetition, change, and surprise [381].

4.1.8 Ambiguity in Design

Finally, ambiguity and uncertainty are considered by some authors to be an integral part of design, thus naturally leading to, or implying, multiple design alternatives. For instance, Cross [86, p. 20] claims that ambiguity is “essential to design process, allowing participants the freedom to manoeuvre independently within object worlds and providing room for the recasting of meaning in the negotiations with others”. Each such recasting could then independently lead to a new design alternative. J. N. Hooker [162] expressed the notion that any given design, due to its ambiguity, in general encompasses a whole class of designs instead of just one specific design: “a characteristic and remarkable trait of design is that it deals with incompletely described objects”. Furthermore, M. Black [40] lists three types of indeterminacy:

(i) Ambiguity, when a choice between several or many alternatives has to be made,
(ii) Vagueness, when an entity (object, process, . . . ) must be specified for later elaboration, and
(iii) Generality, when a precise description can actually lead to many different elements.
According to the Black’s classification above, ambiguity thus already implies a set of pre-existing alternatives, and emphasizes the importance of choosing among such alternatives. Vagueness, on the other hand, permits for multiple design alternatives by implying different “elaboration paths” which then might lead to different design alternatives. Generality, in turn, implies a phenomenon of certain generative nature, because a “precise description” implies many possible valid elements, or designs, matching such a description.

4.2 Fixation on a Single Design Solution

In an apparent paradox, it seems that two contradicting bodies of evidence, the first confirming the claim that designers indeed do work with entire sets of alternatives (as demonstrated and argued for in the previous section), and the second one claiming that designers instead “fixate” on a single design solution, exist. In this section, § 4.2, I provide an overview of the evidence supporting the view that designers fixate on a single design instead. In the next section, § 4.3, I conjecture what might be the possible reasons for this apparent discrepancy, and provide an attempt to reconcile the two (apparent) schools of thought.

According to this second school of thought, then, some designers actually “fixate” on a single “core” design solution at the outset, and then develop (tinker with, tweak, or patch) this solution until it becomes a satisfactory one. Terms similar to “core solution” include the concepts of “primary generator” [92], “kernel idea” [182], “central concept”, “early solution conjecture”, “primary position”, “guiding theme”, “position-driven design”, “early fixation”, “design fixation”, or “solution fixation” [170], “premature commitment”, and “early crystallisation” [392].

As I already mentioned, designers are in general solution-focused, instead of problem-focused [85, 108]. However, Cross [85] warns that this solution-oriented behaviour might lead to an “fixation effect”:

“The solution-focused nature of designer behavior appears to be appropriate behavior for responding to ill-defined problems. Such problems can perhaps never be converted to well-defined problems, and so designers quite reasonably adopt the more realistic strategy of finding a satisfactory solution, rather than expecting to be able to generate an optimum solution to a well-defined problem. However, this solution-focused behavior also seems to have potential drawbacks. One such drawback might be the fixation effect induced by existing solutions.”

Therefore, somewhat counter-intuitively, the designers’ very tendency to produce multiple design solution proposals instead of focusing on the analysis of the problem, can also lead to the effect of designers fixating on one, or just a few such design solutions.
4.3 Discussion

Empirical studies corroborating or observing the “design solution fixation” behaviour include for instance a study by Atman et al [26] in which some of the participants expressed the opinion that designers are not supposed to develop alternatives, while yet other participants in the same study indeed did develop multiple design variants (to be exact, in this particular study, around eleven alternatives on the average). In another example, in a series of experiments undertaken by Purcell et al [295, 297, 296], the authors determined that senior students of mechanical engineering are much more prone to solution fixation as compared to senior students of industrial design. In the realm of expert opinion, Álvaro Siza Vieira, a Portuguese expert architectural designer and the winner of the 1992 Pritzker prize, prefers to “elaborate an idea” when going about designing a building, thus apparently indicating a preference for solution fixation [120, p. 72]:

“Preparing the plan [of the building] is not a process of collecting a certain type of background material; instead the information is merely the means of correcting and elaborating a certain idea—an idea that occurs upon one’s first glimpse of the area.”

In another description of design fixation involving architectural designers, P. G. Rowe [313] observes that initial design ideas exert “dominant influence” on design problem solving, causing a designer to frequently backtrack to such early ideas i.e., “to retrench to what seemed as a more advantageous position”. N. Cross [87, p. 21] comments that “the designer adopts a blinkered approach, overly focused on a particular solution concept”. Cross further interviews two expert designers in the fields of automotive design and product design, and describes how such designers come up with innovative, radical design solutions under tight temporal and financial constraints. In the first case study involving a famous racing car designer, Gordon Murray, Cross describes how the designer devised the hydro-pneumatic suspension, using terms such as “sudden illumination”, or “a basic creative idea”. Thus, a “clear, generative design concept” is devised, sometimes through a sudden, radical insight, to be refined further in later stages of design. The implied claim, of course, is that designers as a rule come up with just a single such “basic creative idea”, instead of multiple ones.

4.3 Discussion

Is there a way to recombine the ideas from § 4.1 and § 4.2 into a complex whole? How can we reconcile the two apparently contradictory bodies of evidence concerning the multiplicity of design alternatives?

\footnote{N. Cross then proceeds to criticize this approach, where the designer “doggedly” presses on, by stating that “a more considered and reflective approach, and consideration of alternative solution concepts, might save time and effort in the long run”, thus contributing to expert opinion listed in § 4.1.6.}
Probably, no reconciliation is in fact necessary because, for instance, even in a single study [26] different participants expressed different opinions, either pro or contra multiple alternatives. Thus, whether a designer in fact develops or does not develop multiple design alternatives in parallel might depend on his personal style, work preferences, the domain of expertise, and many other potential factors. The following list provides a number of such factors which might or might not influence whether a designer engages in alternatives-rich work:

1. **Fixating on the “guiding theme” only.** Some designers might indeed fixate on the “guiding theme” only, and then develop multiple design alternatives in later stages of design [392, p. 201]. Such a guiding “theme”, then, should not be understood as a “design alternative”, but simply as the unifying, general *motif or idea* providing the overall framework to embed design alternatives in. For instance, the two aforementioned expert designers from two different fields of design (Álvaro Siza Vieira, and Gordon Murray) both talk about an “idea”, or “a basic idea”, which should most probably be understood as the “unifying theme” that lends consistency to all the subsequently developed design revisions, including multiple design alternatives *stricto sensu*.

2. **Personal preference.** Simply, some designers prefer to work on and tweak a single design solution, while others might prefer to develop a number of design solutions in parallel. In this case, a personal preference cuts across other concerns and aspects of design.

3. **Inconsistency of declared and pursued intent.** Interestingly, some designers might state that they will indeed pursue multiple design alternatives, yet fail to do so afterwards [392, p. 201]. Possible reasons for this behaviour include undue pressure and/or expectations in an empirical study, poor experimental design, or the nature of the design problem at hand.

4. **“Open-ended” design projects.** “Open-ended” or “creative” design situations, such as those encountered in architectural design [11, 10, 208] might lead to a higher multiplicity in designing. In this case, designers develop multiple design alternatives because they are solving non-routine, “creative” problems, i.e., the first of the following three problem types:

   (i) **Exploratory/evaluative, creative, “black box” design problems.** When solving these design problems, the designer explores different solutions, evaluates them, and repeats the cycle [59, 60]. Since the designer creates a completely new artifact, he or she thus lacks initial knowledge in order to successfully solve the problem [107]. Such problems are also called “black box” because cognition behind such methods is unavailable to introspection [179]. In conclusion, this design problem type lends itself naturally to high concurrency, because multiple design alternatives allow the designer to better explore and define the problem [159, 85].

   (ii) **Inventive, variant, “glass box” design problems.** Here the designer revises an existing design, thus is already in possession of the required knowledge, but still needs to adjust
existing components [107]. This type of design problems are called “glass box” problems because they are fully explainable [179]. Since designer starts with a pre-existing design, this problem type might lead directly to design fixation behaviour.

(iii) *Constraints and requirements-based, routine, “search” design problems.* The designer solves a well-known set of constraints and requirements [59, 60], is in the possession of most required knowledge [107], and deals with the situation by managing search patterns [179]. In other words, these are routine, “mundane” and everyday design problems. This problem type might lead to design fixation because such problems are well-constrained, thus significantly limiting the space of feasible or adequate design solutions.

(5) *Team projects.* Designers working in teams, as opposed to working solo, might prefer [347] developing multiple design solutions, instead of focusing on just one solution.

These are just some factors which might influence whether a designer (or a team of designers) will recur to work processes involving multiple design alternatives. Of course, there might be additional ones. However, according to the points listed above, one may safely conclude that evidence exists for both (1) the design solution “fixation” behaviour, and (2) for designing involving several or many design alternatives at once.

### 4.3.1 A Way to Conceptualize Design Multiplicity

Considering the discussion above, how can we best conceptualize and think about the phenomenon of multiplicity or concurrency in design? One way would be to assume a simple model of designing where designers, in general, work with $m$ alternatives at a time. Any specific instance of “design fixation” can then simply be regarded as the special case of $m = 1$.

![Design multiplicity graphs for two (hypothetical) designers](image)

**Figure 4.8:** Design multiplicity graphs for two (hypothetical) designers, are in general different for the same design project.

According to this model, multiplicity $m$ fluctuates during a design project; at any given point in time, a designer might work with just one design solution; at another point, the designer might now
work with perhaps five design alternatives, either for the same or a different design requirement. I depict such a fluctuation of design multiplicity in Figure 4.8, for two hypothetical designers “Emily” and “John”, when tasked to solve a specific design problem consisting of twenty episodes of designing.

In this hypothetical example, John, as compared to Emily, exhibits a stronger tendency for “design fixation” behaviour. Out of 20 episodes of designing, John spent 9 episodes working with a single design solution, compared to Emily’s 5. Therefore, the notion of “working with multiple design alternatives in parallel” would be a fluid one. In general, a designer would shift his mode of working opportunistically, and consider either just one, two, or more design alternatives at a time.

Considering the evidence provided in this chapter, the final implication is that an effective pCAD tool (and indeed all computational tools, if we consider designing as the general cognitive activity supported by such tools⁶), should support working with $m$ design alternatives, or design solutions. Within this framework, then, the currently prevalent single-state systems can be best regarded as supporting the special case ($m = 1$) of general, alternatives-enabled design work. By induction, single-state systems might be considered as belonging to the sub-class of the more general class of alternatives-enabled computational tools.

### 4.4 Chapter Summary

In this Chapter, in order to build and present the case for alternatives-enabled work, I described a body of evidence comprising (1) illustrative examples from various fields of design (architectural, product, and industrial design), (2) empirical studies confirming the use of multiple alternatives, (3) instances of expert opinion, (4) a theoretical justification for alternatives, and (5) a brief overview of the phenomenon of design ambiguity that might play a role in the context of multiple design alternatives. I further discussed possible scenarios in which designers might or might not engage in alternatives-rich work, and proposed a simple model of design multiplicity in which the current number $m$ of alternatives depends on a variety of factors, and whose value changes dynamically in the course of designing.

In conclusion, the evidence available confirms the existence of design alternatives, and I can speculate and state a stronger claim that the ability to support alternatives-enabled work should probably be embraced and adapted by pCAD and other types of computational tools, in order to increase the effectiveness of their users.

---

⁶See the general definition of designing given in § 2.1.1.
Chapter 5

Related Work

As I have already mentioned elsewhere (e.g., in Chapter 3, § 3.3), the review of related work is to be considered as one of the “research artifacts” produced throughout the work described in this dissertation, the one which I have been continuously updating over the course of this research, as well as utilized it as a research input to all the other major phases of research. Accordingly, I now turn my attention to past work related to computational support for managing, and working with, multiple design alternatives in parallel. Differently from previous chapters, this one is heavily computing-centric, providing a compendium of past approaches, ideas, and techniques that might constitute fitting and useful “design ingredients” of effective alternatives-enabled computational tools and systems, divided into the following sections:

• Interactive visualization in general (§ 5.1). I start with a description of a large body of work related to HCI design and interaction design in general, and which has in most cases (but not all) been generated in the historic context of single-state tools and systems. I place a special emphasis on existing taxonomies of interaction tasks and techniques, as a way to enumerate potential approaches in a well-structured, systematic manner.

• Multi-state interactive visualization (§ 5.2). The core of this chapter, this section describes past work related to interacting with multiple computational objects in parallel in various domains of application, such as parallel exploration of n-dimensional parameter spaces, parallel searches of the World-Wide Web, set-based exploration of raster images, parallel text editing, and others.

• Design history (§ 5.3). Various ways to interact with design history have been reported in the literature and implemented in assorted research and commercial systems, including viewing, navigating, undo/redo, editing, filtering, annotating, and searching past (historical) design states.
5.1 Interactive Visualization in General

Computational tools are “cognitive artifacts” that display or operate upon sets of information, in order to serve a representational function and facilitate human cognition (cf. Section 2.7 in Chapter 2, as well as [271] for detailed discussion). The use of computational tools hence implies the two principal processes of visualizing (i.e., representing, rendering) a set of information, and interacting with said sets of information.

Considering these two fundamental functions (representing and interacting) related to interactive systems, I adopt the view that they are both inextricably linked to each other, and that they work in a synergistic fashion in order to “scaffold the human knowledge construction process” [285], in particular the “design knowledge” described in section 3.2.2.1. I will thus, in the remainder of this dissertation, use the all-encompassing term “interactive visualization” to denote this tight coupling between interaction and visualization, as well as to refer to the large body of past work that is in any way or sense related to either visualization, human-computer interaction (HCI), and akin areas, and which, again, should be regarded as constituent sub-disciplines of the integrated discipline of interaction design [328]. Another fairly recent term that describes this integrated body of knowledge, and has lately seen much use, is “interaction science” [73, 285].

Turning our attention back to the concept of cognitive artifacts of which computational tools are but one subset, a well-designed computational tool can transform [271] an existing “user + task \( T + \) old tool” total system to a more efficient “user + new task \( T + \) new tool” system, where \( T \) stands for task(s) made possible through the use of the new tool. Yet, as I have already mentioned in Chapter 3, Section 3.3, since this thesis deals with an innovative or exploratory design of tools supporting sets of pCAD alternatives, by definition one does not, and cannot, know precisely which tasks \( T \) should be supported by such novel tools. In addition, even if one were in possession of an exhaustive list of all the tasks that a tool should support, how should the interaction designer-researcher select and decide on suitable interaction visualization techniques for fulfilling said tasks?

As another perspective of the same problem, and using Simon’s terms [336, 341], the goal of this research is to find design solutions (i.e., multi-state tools) within large “solution spaces” which, in turn, are induced by the corresponding task environment whose elements, tasks, can be understood as hierarchies of sub-tasks. Within this work, those tasks can be understood as tasks for working with multiple design alternatives; since those tasks are by and large currently unknown, this in turn implies that the induced solution space(s) are currently unknown as well. Since we know, however, that solution spaces are, as a rule, vast, in this work I will recur to “examining promising regions of the entire space and simply ignoring the rest” [341]. I will find these “promising” regions of IxD solutions by utilizing various heuristics; in the first place, by reviewing pertinent work (in this chapter), and later one by adopting other approaches such as
prototyping, formative studies, and evaluation studies.

Thus, at the outset of any innovative interaction design research, the researcher-designer deals with at least two known unknowns: tasks, and interactive visualization system capabilities supporting said tasks. A good place to start, therefore, is to consider sufficiently complete classifications of tasks and techniques that any reasonably complete, general computational tool should support. Such classifications are conveniently provided in the form of taxonomies, typologies, and categorizations of interaction primitives.

5.1.1 Taxonomies of Interaction Tasks and Techniques

A number of different taxonomies, typologies, and approaches to conceptualize and systematize interaction tasks, actions, or primitives have been reported in the literature [333, 331, 418, 69, 16, 286, 135, 311]. To start with an early, well-known and influential classification, in the field of information visualization, Shneiderman [333, p. 539] puts forward his widely known “interaction mantra” of “overview first, zoom and filter, then details on demand” acting over seven basic data types (1D, 2D, 3D, n-D, temporal, tree, and network), as well as seven fundamental tasks manipulating such data (Figure 5.1).

<table>
<thead>
<tr>
<th>(1) Overview</th>
<th>of a collection.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2) Zoom</td>
<td>into and out of a collection, or an item.</td>
</tr>
<tr>
<td>(3) Filter out</td>
<td>(i.e., suppress) items that are irrelevant or uninteresting from the view.</td>
</tr>
<tr>
<td>(4) Provide details on demand</td>
<td>for a collection, or an item.</td>
</tr>
<tr>
<td>(5) View relationships</td>
<td>among a number of items.</td>
</tr>
<tr>
<td>(6) Provide history of actions, so as to support undo, redo, progressive refinement.</td>
<td></td>
</tr>
<tr>
<td>(7) Extract subcollections</td>
<td>from a larger, containing collection, e.g., by modifying query parameters.</td>
</tr>
</tbody>
</table>

Figure 5.1: Seven fundamental interaction tasks by Shneiderman and Plaisant [333, p. 539].

Following this progressive sequence of overview → zoom into → filter out → work with details and relationships of interaction tasks, it is helpful to reinterpret the sequence in terms of the main constructs used in this work, i.e., going from the top level (design project level) of the tool, and then drilling down to collections, alternatives, and parts of alternatives\(^1\). In addition, this sequence can provide helpful guidance on the implementation (i.e., software architecture) of alternatives-enabled tools as well.

\(^{1}\) I depict this schematically in Chapter 10, Figure 9.24.
WARE [398, CH. 10] OBSERVES THAT IN REALISTIC SCENARIOS IT IS FAR MORE LIKELY THAT THE USER WILL EXECUTE SCHNEIDERMAN’S SEVEN FUNDAMENTAL TASKS IN A MORE OPPORTUNISTIC FASHION, I.E., NOT NECESSARILY IN THE ORDER GIVEN ABOVE. FOR INSTANCE, THE USER MIGHT ZOOM INTO A REGION, THEN ZOOM OUT, THEN ZOOM INTO ANOTHER REGION, VIEW DETAILS AND RELATIONSHIPS, BACKTRACK TO ADDITIONAL FILTERING, ETC., THEREFORE PERFORMING ANY OF THE SEVEN TASKS IN AN OPPORTUNISTIC AND PRAGMATIC FASHION.

NEXT, AMAR ET AL [16] PRESENT A TAXONOMY OF TEN LOW-LEVEL VISUALIZATION TASKS, CAPTURED BY ANALYZING 200 SAMPLE QUESTIONS FROM STUDENTS ON HOW THEY WOULD ANALYZE FIVE DIFFERENT DATA SETS (FIGURE 5.2).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Compute Derived Value</td>
<td>7. Characterize Distribution</td>
<td></td>
</tr>
<tr>
<td>4. Find Extremum</td>
<td>8. Find Anomalies</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5.2:** Taxonomy of interaction tasks by Amar et al [16].

In the field of visual analytics, Gotz and Zhou [135] describe an approach to tracking insight provenance (i.e., recording of the rationale of the process behind insight). They also suggest a catalogue of twenty common actions (or “units” of visual analytics activity), and a taxonomy of actions comprised of three major intent classes (Figure 5.3): exploration, insight, and meta actions.

1. **Exploration** actions, which are further divided into:
   
   (a) **Data Exploration** actions: Filter, Inspect, Query, Restore.
   (b) **Visual Exploration** actions: Brush, Change-Metaphor, Change-Range (subsumes Zoom and Pan), Merge, Sort, and Split.

2. **Insight** actions, which are further divided into:
   
   (a) **Visual Insight** actions: Annotate and Bookmark.
   (b) **Knowledge Insight** actions: Create, Modify, and Remove.

3. **Meta** actions: Delete, Edit, Revisit, Redo, and Undo.

**Figure 5.3:** Taxonomy of interaction tasks by Gotz and Zhou [135].

For the set of design guidelines for specifying alternatives-enabled pCAD tools, and which I
present in Chapter 10, I decided to lean heavily upon the following two recent yet well-received articles dealing with taxonomies of interaction primitives, the first one by Matthew Brehmer and Tamara Munzner [49], and another one by Robert E. Roth [311]. Due to the import and overall significance of these two papers, I decided to devote separate sections § 5.1.1.2 and § 5.1.1.1 to these two classifications, as follows.

5.1.1.1 Interaction Typology by Brehmer and Munzner (2013)

Brehmer and Munzner [49] present a more general, multi-level typology of abstract visualization tasks in which said tasks are categorized in terms of their

1. High-level reasons or intents (the “why?” category),

2. The methods used to carry out said tasks (“how?”), and

3. What are the inputs and outputs to said tasks (“what?”).

Figure 5.4 below summarizes the “why?” category, and Figure 5.5 the “how?” category.

(1) **High-level intents**: **Consume, Produce**. The intent of Consume can further be one of the following:

   (a) **Present**. Includes story-telling, communicating information.

   (b) **Discover**. Generation and verification of scientific hypotheses.

   (c) **Enjoy**. Casual viewing of the visualization.

(2) **Mid-level intent**: **Search** methods for finding elements of interest, which include **Lookup**, **Locate**, **Browse**, and **Explore**.

<table>
<thead>
<tr>
<th>Target known:</th>
<th>Location known:</th>
<th>Target unknown:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lookup</td>
<td>Browse</td>
</tr>
<tr>
<td>Location known:</td>
<td>Locate</td>
<td>Explore</td>
</tr>
</tbody>
</table>

(3) **Low-level intent**: **Query**. Querying of search results, includes **Identify**, **Compare**, and **Summarize**.

**Figure 5.4**: “Why will this task be performed?”, according to the typology by Brehmer and Munzner [49].
(1) **Encode.** Refers to how a certain set of data is presented visually on computer display. For example, a data set might be visualized as a group of lines and shapes, or as a data grid of alphanumeric values.

(2) **Manipulate.** Includes the tasks of Select, Navigate (e.g., overview + detail, zooming, focus + context (distortion), and cue-based techniques), Arrange, Change (i.e., modify the visualization), Filter, and Aggregate/Segregate (e.g., changing the view from “Daily” to “Monthly” and vice versa).

(3) **Introduce.** Includes the tasks of Annotate, Import, Derive, and Record (most frequently associated with the Produce intent described in the “Why?” category, for saving visualization elements such as e.g., screenshots and settings).

**Figure 5.5:** “How will this task be performed?”, according to the typology by Brehmer and Munzner [49].

Equivalently, in the field of CAD or computational design, in the “why?” category one might adapt high-level intents e.g., Converse-reflectively, Enjoy, Present, and Produce. The intent Converse-reflectively addresses the activity of solving “messy” design problems, within the Schön’s notion of “reflective conversation” [324] with the materials of the situation (i.e., while interacting with a computational design tool, supporting certain design representations). The intent of Enjoy, likewise, would be to allow the user to casually view a collection of design alternatives, just like a collection of works of arts is available for viewing in a real, brick-and-mortar art gallery. High-level goals Present and Produce would retain their original meaning, as well as mid-level and low-level goals or intents, due to their universal applicability to any reasonably complete computational tool or system.

The interaction methods (the “how?” category) shown in Figure 5.5 apply directly to CAD computational tools as well due to the universal nature of said methods, with the necessary qualification that the methods suitable for alternatives-enabled pCAD tools will unavoidably have their own distinctive characteristics, and which I will describe in full in Chapter 10.

As for inputs and outputs (the “what?” category), a number of tasks can be chained into an “interaction process”; the output from a preceding task can be fed as an input into the succeeding task. According to Brehmer and Munzner, inputs and outputs for abstract visualization tools include statistical values, graph values, time series data, and various interaction (GUI, visualization) objects such as widgets, diagrams, and screenshots. Within the context of this thesis, major inputs and outputs include interaction objects, design states, collections of design states, design history (the global DAG), and individual items within design alternatives.
5.1. INTERACTIVE VISUALIZATION IN GENERAL

5.1.1.2 Interaction Taxonomy by R. E. Roth (2013)

In a recent paper [311], Robert E. Roth presents an empirically-derived taxonomy of interaction primitives in the domain of cartographic interaction, but which can also be regarded as a significant contribution to the interaction science in general\(^2\). The author builds his taxonomy on the basis of Norman’s seven-step interaction model [273, 271]\(^3\), the model that I use heavily in this work, and considers the following four dimensions: goals, objectives, operators, and operands in this taxonomy, whereby a task can be expressed formulaically as

\[
\text{Task} = \text{Goal} + \text{Objective} + \text{Operator} + \text{Operand}.
\]

(1) **Goal.** Refers to the first step ("set goal") in the Norman’s seven-step model of interaction, and represents the highest-level “cognitive involvement” in Roth’s taxonomy. Goals can be, by increasing order of cognitive complexity:

(a) **Procure.** Retrieve information about some geographic phenomena.

(b) **Predict.** Forecast what might occur in the future based on current conditions.

(c) **Prescribe.** Suggest what should occur in the future based on current conditions and future predictions.

In computational design, high-level goals might include, and in concordance with the discussion in § 5.1.1.1, Converse-reflectively, Enjoy, Present, and Produce.

(2) **Objective.** Can be Identify, Compare, Rank, Associate, and Delineate. Refers to the second step ("form intention for action") in the Norman’s model. An objective can be regarded as a bridge spanning the goal (see above), and the operator (see below). Since there are 3 operand types and 5 objective types, this results in the total of $3 \times 5 = 15$ different objective+operand combinations.

(a) **Identify.** Refers to searching for, and finding, map features:

i. **In a space.** e.g., find a house in an aerial image of an area.

   • **In pCAD\(^4\): find a design alternative within collection(s); find items within alternative(s).**

ii. **By attributes in a space.** e.g., find hazardous materials in a building on fire.

---

\(^2\)I also note here a certain similarity of 2D vector drawings to 2D map rendering, which contributes to the relevance of the article to this thesis’ research.

\(^3\)See also the schematic depiction of Norman’s seven-step interaction model in Fig. 2.9 on page 31.

\(^4\)For each illustrative cartographic example given by Roth, I contribute equivalent pCAD examples (using the cursive typeface), for each objective+operand combination.
• pCAD: Find all items whose geometry resembles a rectangle, or whose border colour equals #84AAFF, within one or multiple alternatives. Find collections whose similarity index lies in the range $[3.78, 7.55]$, using some similarity metric.

iii. In time, for a space. e.g., how many hotels were there in this city 50 years ago?
• pCAD: How many items $a$ were contained within this design alternative a year ago, or twenty design revisions ago?

(b) Compare. Refers to determining similarities and differences between two map features:

i. In a space. Compare crime rate and foreclosure distributions, for an area.
• pCAD: Compare two (or several) design alternatives within a collection, or multiple collections. Compare all the selected items within multiple alternatives.

ii. By attributes in a space. Juxtapose two different policies on a map.
• pCAD: Compare e.g., respective rates of change for two or several design alternatives, while either juxtaposing or superposing said alternatives.

iii. In time, for a space. Compare the current and a past spatial extent, for a forest.
• pCAD: Compare how has an item developed in time; for instance, the revisions of an item taken at the following times: 1, 2, 3, 4, 5, 10, and 15 days ago. Or, the same for a design alternative.

(c) Rank. Order or relative position of three or more map features.

i. In a space. e.g., what are the nearest/farthest grocery stores to my location.
• pCAD: What are the most similar or dissimilar design alternatives within a collection, for a selected alternative, using a similarity metric?

ii. By attributes in a space. e.g., what restaurants have the best ratings in an area.
• pCAD: What design alternatives have the best ratings (i.e., personal preference scores) in a collection, or multiple collections?

iii. In time, for a space. e.g., have there been any traffic incidents in an area.
• pCAD: Has a particular item ever been used in this design alternative?

(d) Associate. Refers to characterizing the relationships among map features.

i. In a space. How is this store connected to the main highway?
• pCAD: Is this item $a$ linked to another item $b$?

ii. By attributes in a space. Are crime rate and foreclosure distributions correlated spatially?
• pCAD: For two or several design alternatives, are the items that are located at the same geometric position in each respective alternative, also related to each other?
iii. *In time, for a space.* Has the geographic extent of a harmful chemical been decreasing?

- **pCAD:** *Has the geometric extent (i.e., the bounding box; alternatively, the relative importance, degree of in/out linking with regards to constraints, etc.) of an item been decreasing or increasing within an alternative?*

(e) **Delineate.** Refers to dividing and organizing map features into some logical structure.

i. *In a space.* What are the high-risk clusters of morbidity in an area?

- **pCAD:** *Group design alternatives into proximity-based clusters, using some personal preference or criteria.*

ii. *By attributes in a space.* Find clusters of similar attribute values, e.g., average income.

- **pCAD:** *Find alternatives with similar attribute values, using some similarity metric.*

iii. *In time, for a space.* Find out historic extremes (“spikes”) of crime rates in an area.

- **pCAD:** *Find out which alternative(s) underwent most modifications in the past two months.*

(3) **Operator.** Refers to the third step (“map intention into action”) in the Norman’s model. In other words, the user identified an interactive system feature (“operator”) that helps him achieve his objective. Operators refer to system functions i.e., interactive functionality for manipulating a visualization, and are also commonly also known as “interaction techniques”, “tasks”, “operations”, “manipulations”, or “behaviours”.

(a) *Work operators (Re-express, Arrange, Sequence, Re-symbolize, Overlay, Re-project, Pan, Zoom, Filter, Search, Retrieve, and Calculate), and*

(b) *Enabling operators (Import, Export, Save, Edit, and Annotate).*

Again, just like in the typology by Brehmer and Munzner (the “how?” category), the “operators” by Roth would apply directly to CAD computational tools as well, due to the universal nature of said operators. However, the same qualification applies as well: operators suitable for alternatives-enabled pCAD tools will unavoidably have their distinctive characteristics, which I detail in Chapter 10.

(4) **Operand.** Specifies the recipient of the operator, i.e., the visualization state acted on. In Roth’s approach, these can include *Space-Alone* (e.g., a house on a map), *Attribute-in-Space* (e.g., all buildings with more than 30,000 sq. feet), and *Space-in-Time* (e.g., buildings built and demolished in this area on the map for the last 150 years).
Equivalently, in multi-state pCAD, one might imagine operands Geometric-Scope (i.e., the spatial extent of an alternative, or an item), Property-Value-in-Geometric-Scope (e.g., border colour, or geometry, of an item), Geometric-Scope-in-Time (e.g., design history of an item, or an alternative; for instance, how the geometry of an item changed throughout time), In-Constraints (the set of constraints through which an item influences other items), or Collection-of-Alternatives (i.e., an operand to be fed into collection-specific operators).

As a conclusion to this section, Section 5.1, interaction design researchers have over the years attempted to enumerate, describe, and systematize interaction tasks and operations, resulting in a number of corresponding taxonomies, typologies, and categorizations. Some of these provide a template as well as a part of inspiration for coming up with my own taxonomy of interaction primitives for alternatives-enabled pCAD tools, which I will in turn use to articulate and frame the set of design guidelines for said tools.

5.2 Multi-State Interactive Visualization

A number of approaches, tools, and techniques to interacting with multiple computational objects at once have been reported in the literature. To denote such simultaneous interaction with multiple objects, one might also use designations “multi-state interaction”, “multiplexed interaction”, “plural interaction”, or “subjunctive interaction”. In other words, interaction that involves more than one design state, or artifact, at a time, organized into following categories:

(1) **Parallel visualization** (§ 5.2.1). Reported approaches to visualizing multiple computational objects at once.

(2) **Subjunctive interfaces** (§ 5.2.2). Reported approaches to interacting with many computational objects at once, reported by A. Lunzer and his collaborators, with particular emphasis on the mixed-initiative exploration of $n$-dimensional parameter spaces.

(3) **Set-based interaction** (§ 5.2.3). Reported approaches to set-based interaction with multiple raster images, by M. Terry and his collaborators.

(4) **Parallel text editing** (§ 5.2.4). Reported approaches to parallel text editing.

(5) **Other approaches** (§ 5.2.5). Other reported approaches to working with multiple objects in parallel and which cannot fit into any of the categories above.

(6) **Commercial tools** (§ 5.2.6). Finally, an overview of current state-of-art in commercial tools and systems that provide at least some support for multiple design alternatives.

---

5Lunzer et al in fact use the adjective “multiplexed” in at least one of their publications [222].
5.2. MULTI-STATE INTERACTIVE VISUALIZATION

5.2.1 Parallel Visualization

Within the general body of work related to ways and approaches to visualize multiple computational objects at once, “small multiples” [380, 381], or scaled-down images, facilitate comparison of multiple (similar) objects, i.e., objects with similar structures repeated across all such objects. I note here that multiples can be “small” or not. However, when presented with multiple said images, their stability of structure allows the user to “focus on change of information rather than changes in graphical composition” [380, p. 28, 29]. As another approach to displaying multiple computational objects, Roberts [309] describes multiplicity in visualization, where data are shown in separate windows. The authors list the following five reasons for using multiple views of data: 1) correct dissemination, 2) exploration, 3) alternative viewpoints, 4) realization comparison, and 5) collaboration; in (multi-state) computational design, the advantages listed might be re-interpreted as follows:

1. Multiple views or representations of a pCAD model might lead to correct dissemination in the sense that the user will, by being able to view and reflect upon multiple different representations, gain a more complete and correct understanding of the model(s);

2. Multiple views might aid the exploration and examination of the model(s), through mechanisms such as brushing and linking of separate view, selecting multiple items within multiple views, and viewing of design history;

3. Multiple views might provide different interpretations of the same model(s) to different users, thus enabling alternative viewpoints which could thus in turn lead to additional insights;

4. Multiple aids might aid with comparing the same design state, or multiple design states, and

5. In collaborative settings, multiple views showing the same visualization to each collaborators can facilitate collaboration.

In a more recent paper [310] by the same author, Roberts reviews the state-of-art in the field of coordinated and multiple views.

Herring et al [156] confirm the crucial importance of being able to see multiple examples in the context of web, graphic, and product design, and recommend that computational tools provide support for retrieval, storage, search, and dissemination of multiple such examples.

In yet another approach to viewing multiple objects, Busking et al [56] study the evolution of multiple shapes (i.e., 2D or 3D surfaces represented as “statistical shape models” [SSMs]) within “shape spaces”, as well as means to explore and visualize said shapes, and present an approach characterized by strongly-linked multiple views.
As shown in Figure 5.6, within each shape space, a shape is represented by a point; once a point is selected, a shape is visualized within the object space view. In their approach, the evolution of shapes can be visualized using (1) side-by-side displays, (2) overlaid contours, and (3) as a stack of temporally ordered shapes. As the reader will notice in subsequent chapters and sections, these three presentation modalities appear to be some of the most frequently reported and used ways to display multiple objects at once.

### 5.2.2 Subjunctive Interfaces (A. Lunzer)

In another significant body of work related to alternatives-enabled computational tools (with special emphasis on interaction), that of subjunctive interfaces, Lunzer, Hornbæk, and their colleagues present a number of approaches to interacting with many computational objects at once, with particular emphasis on the mixed-initiative exploration of \( n \)-dimensional parameter spaces defining such objects. Due to the importance and overall relevance to this work, I devote it a separate section.

To start with the original motivation for subjunctive interfaces, D. R. Hofstadter’s presented the concept of Subjunc-TV[161], a kind of television where the viewer is allowed to select among several parallel versions of the same TV show, and which are created upon viewer’s suggestions on how the narrative should be influenced and developed. According to Hofstadter, such a television could satisfy typical human curiosity related to “what-if” situations, or how a situation might have
developed, thus leading to its name\textsuperscript{6}. Therefore, although the viewer can see just one channel at a time, multiple possible scenarios of how a show might play out become possible, using such a television system.

Motivated by Subjunc-TV, and its ability to support multiple “realities”, in an early research report [218] as well as in his PhD thesis [219] Aran Lunzer explores computer-assisted searches in large parameter spaces, which he termed \textit{computational reconnaissance}. In his approach, the user selects several options (or parameters), which can then assume a number of values leading to a Cartesian product set of those options; the author calls such a set of options \textit{parallel coordinates}. Such an approach provides a middle ground between fully automated searches in parametric spaces (i.e., constraint satisfaction problem solving, where the user is obliged to manually build a network of such constraints), and manual “dynamic queries” [7, 334, 6, 330] where the user does not have to build a network of constraints defining feasible and “acceptable” objects, but must on the other hand interactively manipulate multiple sliders or GUI controls (one such slider or control per parameter) in order to find such feasible or acceptable objects.

As an example of a tool built on the concept of computational reconnaissance, Lunzer presents a proof-of-concept prototype built on top of \LaTeX, called \textit{InteracTeX}, where the user can vary three input variables (font, style, and paragraph skip) resulting in three output measures (total pages produced, last page length, and whether a list was placed completely on a single page), thus helping the user decide on an acceptable set of input parameters. Each data point in this space is then represented by a set of values for each of the input variables and output measures. Once the user specifies a number $N$ of such data points, \textit{InteracTeX} runs automatically $N$ times as well, thus producing $N$ reconnaissance reports. The user can review each of those $N$ reports, and decide which sets of parameters are the acceptable ones, or most desirable ones.

In a subsequent work-in-progress publication [220], Lunzer again deals with spaces in which multiple dimensions can vary at the same time, and goes on to suggest “subjunctive” interfaces based on the following set of four design principles.

\begin{itemize}
\item[(1)] \textit{Specify multiple values for a parameter}. The user has the freedom to specify multiple different values for each parameter defining a state. For instance, a slider can assume multiple such values, as distinguished from conventional sliders which can assume one value only.
\item[(2)] \textit{View many states at once}. The user can view multiple states (defined by all the combinations of the parameters and their values) at once. For this, Lunzer suggests semi-transparent displays whereby multiple states can be rendered on top of each other (i.e., superposed) in a single view.
\item[(3)] \textit{Compare many states at once}. The user can compare multiple states at once, by visually
\end{itemize}

\textsuperscript{6}Subjunctive is an adjective that describes states that are contingent or possible.
(4) **Select preferred and discard unwanted states.** Finally, the user can select preferred states, and discard unwanted ones.

He goes on to illustrate such principles by showcasing two interface mockups, one based on a 3D authoring environment, and another one related to text editing.

In a continued elaboration [221] of subjunctive interfaces, Lunzer claims that while dynamic queries and *constraint satisfaction solving* allow for only very narrow progress in the “result space”, subjunctive interfaces should allow for a broader exploration of such spaces. Therefore, subjunctive interfaces should be much more suitable for *exploratory tasks*, or *exploratory design*, because in many cases the user does not know in advance what would the end result be, or look like⁷. According to Lunzer, while utilizing the *cognitive dimensions* framework [139, 138], current single-state systems fare poorly along the following dimensions:

1. **Visibility.** Due to having to work with a single data point at all times, visibility of single-state systems is poor.

2. **Viscosity.** Switching data points in single-state systems involves a lot of effort, therefore viscosity is high.

3. **Juxtaposability.** Again, due to having to work with just a single data point, it is not easy to view and evaluate multiple data points at once. Therefore, juxtaposability is poor in single-state systems.

4. **Premature commitment avoidance.** Low visibility, high viscosity, and low juxtaposability lead to the detrimental effect of user frequently committing to what might be a subpar data point. Therefore, single-state systems fare poorly along the “premature commitment” dimension as well.

Again, dynamic queries [7, 334, 6, 330] help alleviate the issues listed above, and improve on all four dimensions. However, dynamic queries inherently deal with static data; they are unsuitable for *steering* the direction of design exploration towards interesting regions of the design space. Lunzer [221] thus again argues for and champions subjunctive interfaces, which afford the following four main benefits:

1. **Specifying multiple values for a parameter.** To avoid premature commitment, Lunzer argues that the user should be able to specify multiple values for a parameter, and work with such values in parallel i.e., simultaneously.

---

⁷M. Terry calls such tasks *open-ended tasks* [369, 370]. Other designations of such tasks include *creative*, *non-linear*, and *experimental*. 
5.2. MULTI-STATE INTERACTIVE VISUALIZATION

(2) **Steering multiple paths in parameter space.** Instead of always exploring just a single path in the parameter space (such as with dynamic queries), the user should be able to pursue multiple such paths at once.

(3) **Replaying the interaction path.** Frequently, each interaction step leads to the creation and opening of a new view obfuscating the view related to the previous interaction step. Lunzer however argues that this leads to high viscosity, and that the user of a subjunctive interface should be able to interact with multiple such (linked) views, thus allowing him to easily modify a previous interaction step and immediately observe the effect of such a modification.

(4) **Recombining parts of solutions.** The user should be able to view multiple solutions at once, and recombine parts of such solution into a new solution. To illustrate this point, Lunzer presents a novel medical image tool named 3DVIEWNIX, demonstrating how partial solutions can be recombined together into a global solution.

Likewise, in a subsequent publication [226], Lunzer and Hornbæk claim that conventional interfaces for interactive exploration of information burden the user through (1) excessive number of interface actions, (2) forcing the user to remember earlier results (thus frequently pressuring the user into writing down such results on a piece of paper), and (3) forcing the user to invest significant effort when trying to explore different combinations of multiple parameters. They claim that these shortcomings can be reduced through the use of subjunctive interfaces, and present two such interfaces for browsing census data used in the paper by Hochheiser and Shneiderman [160]: menu-based and slider-based. In those interfaces, both the input widgets (sliders, menus) and the output widgets (for instance, text labels) are **multiplexed** (see also [222] for a discussion on multiplexed widgets) i.e., can assume or render multiple values at once. The authors then argue that in some scenarios subjunctive interfaces will lead to lower click counts when completing a task, compared with conventional interfaces, thus resulting in more effective work.

In the position paper [223], Lunzer reflects on his earlier work and states that, in retrospect, the feedback and acceptance by users was at best mixed due to (a) extra effort required by the users, and (b) distraction from the core activity. To him, this signalled the need to better integrate alternatives-handling mechanisms into conventional, single-state systems. To this end, Lunzer proposes “ingredient-based computing”, a type of subjunctive interfaces based on data propagation graphs, and in which alternatives are generated automatically instead of by user request, thus providing “mixed-initiative” creation of alternatives. This new approach results in a series of new papers, as follows.

In an usability study [227] of the census data browser featuring a subjunctive interface (and first described in [226]), participants experienced higher satisfaction, decreased use of making notes on paper, and lower cognitive workload, while performing fewer user interface actions (i.e., mouse
clicks), with task completion times reduced up to 27% as compared to conventional, menu-based interfaces.

In [224], Lunzer describes subjunctive interfaces for exploring online resources (such as e.g., flight reservation enquiries), and reiterates that such interfaces offer inadequate support for (1) specifying ranges of (and not only single) search parameters, (2) comparing results, and (3) disambiguating the context of search parameters. Subjunctive interfaces improve on such tasks, by offering (1) multiple scenarios in parallel, (2) juxtaposition of multiple retrievals enabled by such scenarios, and (3) disambiguating the context of retrievals by enabling the user to directly view and compare results. To illustrate these benefits, Lunzer describes five applications: the browser for census data, C3W [123, 124] tool for clipping and cloning snippets of online resources, TopicaBrush tool for comparing linked data sets, HIBench tool offering multiple searches against multi-attribute databases, and ScheduleBlender calendar tool for alternative event scheduling. In [171, 172] Jantke, Lunzer and Fushima explore the application of subjunctive interface to exploratory e-learning, and present two examples: one for exploratory learning of decision trees, and another one for learning about foraging behaviour of ants.

In the continuation of work employing dataflow networks, Fujima, Lunzer et al present the C3W prototype tool [123, 124] for “subjunctive” WWW searching, i.e., searching done in parallel. Likewise, in [224], Lunzer describes subjunctive interfaces for exploring online resources in parallel. Extending this work, in [228, 229] Lunzer and Hornbæk present RecipeSheet, whereby various online resources are linked and processed through multiple, alternative processing pipelines inserted into a spreadsheet-like tool. Just like previous subjunctive interfaces, the user can set up multiple scenarios, view such scenarios in a juxtaposition, and modify multiple such scenarios in parallel. Scenarios are modeled as “recipes”, or as dataflow-based acyclic directed networks of nodes receiving inputs (or “ingredients”) and producing outputs (or “results”). For example, by varying font type, font size, background colour, and text colour, the authors were able to produce three different Web page designs in parallel; by employing juxtaposition, the users are then able to simultaneously compare all three designs.

In the extensive journal paper [230] Lunzer and Hornbæk review past work related to subjunctive interfaces done in the past ten years. In addition to reports concerning subjunctive interface for census data browsing [226, 227, 224], for exploratory e-learning [171, 172] as well as the RecipeSheet tool [228, 229], the authors also describe a subjunctive interface for an text editor, as well as list three key design principles for such interfaces: (1) multiple, simultaneous scenarios, (2) juxtaposition of such scenarios, and (3) parallel modification of said scenarios.

Finally, in more recent work [225, 351], Lunzer et al present an application of subjunctive interfaces and RecipeSheets in the field of medicine (specifically, cancer treatment), correspondingly named OncoRecipeSheet. The tool is able to process and display multiple patient-specific cancer treatment simulations, governed by a relatively large number of parameters.
5.2.3 **Set-Based Interaction (M. Terry)**

Another important body of pertinent work, that completed by Michael Terry and his collaborators, is concerned with providing computational support for *set-based interaction* with a number of computational objects (in this case, raster images) in parallel.

In the first paper [369], Terry and Mynatt explored how experts go about completing “open-ended” (i.e., creative, non-linear, experimental) tasks, and proposed *Side Views*, a GUI design approach to providing on-demand previews of how a command would affect a paragraph of text, or a raster image, for the purposes of: (a) quickly performing “what-if” experiments, (b) previewing not just one but a whole set of alternatives using so-called “parameter spectrums”, (c) comparing many such alternatives at once, by keeping several review windows open, (d) creating alternative visualizations through those same windows, and (e) “serendipitously” chancing upon new, interesting alternatives through accidental discovery. Such preview windows or views are dynamic, support interaction (for example, by resizing it or through a slider control), and can be made to persist on screen (i.e., making the preview stay on screen, instead of disappearing after a couple of seconds, thus effectively creating a new alternative). Furthermore, if the active document (among several or many) is changed by the user, side views are updated immediately as well, thus providing a preview of the new document. Finally, previews can be chained together by dragging and dropping one preview onto another, thus creating a new, composite preview based on multiple visual effects or filters.

In [368], Terry and Mynatt explore three case studies of the creative process in the domain of raster image editing: image toning in newspaper image control; GUI design for interactive multimedia content; and pen drawing colouring in fine art. Based on the data extracted from said case studies, the authors proposed a set of guidelines for specifying effective alternatives-enabled HCI designs. Concisely, the guidelines pertain to supporting:

(i) **Near-term experimentation** i.e., efforts intended to discover and instantiate the next action or operation while using the tool. Traditionally, experimentation has been supported by the user trying out and undoing multiple commands, or tweaking a command’s parameters. To support near-term experimentation, a tool should:

(a) Provide better support for previews of commands.

(b) For parameter tweaking, provide a set of views into the space of possibilities.

(c) Offer a dedicated space (Schön’s virtual worlds [323] in which to test hypotheses) in which to perform near-term experiments, without modifying the document.

(ii) **Varying designs** i.e., creating variations which can be considered to be “design alternatives”.

With conventional systems, users usually create copies of the document and save them under
a different file name, use layers, or place objects side-by-side within a image editor. For this, a tool should:

(a) Support intentional or serendipitous generation of variations.
(b) Treat all parts of the document as containers which permit multiple alternatives to a part.
(c) Record all user’s actions, make the history of such actions permanent, and offer tools to navigate and edit such histories.

(iii) Evaluations of designs i.e., examination of current progress, or the effect and consequences of previous user actions. For this activity, a tool should:

(a) Support alternative representations of documents.
(b) Support multiple views into documents.
(c) Support persistent previews of multiple commands.

To illustrate the application of such design guidelines, the authors then described three different HCI designs: Side Views and Parameter Spectrums (reported in [369]), as well as Design Horizon, a workspace supporting multiple document versions at once.

In [370], Terry et al present a novel approach to simultaneously developing multiple design solutions in the area of raster image editing, called Parallel Paths, with a concrete implementation called Parallel Pies which renders a document in a single view, and allowing the user to view multiple alternatives at once in a kind of “limited juxtaposition” (instead of whole alternatives, just parts of them are shown in a slice of the pie). In this approach, the user can (1) create new image variations before, during, after the execution of a command, (2) visualize multiple image alternatives in one single view, and (3) modify the current image variations either separately (one-by-one), or all of them at once. In the remainder of the paper, authors review the two prior usability studies of their census data browser reported in [223], as well as describe a third usability study of a modified version of the same browser.

Finally, in his PhD thesis [367], Terry evaluated the impact of Side View and Parallel Pies on problem solving by conducting three experimental studies (two controlled, and one think-aloud study). Experimental results related to Parallel Pies indicate both negative effects (use of Parallel Pies can lead to lower-quality results) and positive effects (users explore problems more broadly). Experimental results related to Side Views indicate either no effects (i.e., no change in user performance) or positive effects (easier discovery of unexpected/unplanned-for but still viable solutions).

5.2.4 Parallel Text Editing

Parallel text editing provides additional, important references to the problem of interacting with multiple objects in parallel. For instance, Miller and Myers [252] present a new technique, named
“simultaneous editing”, for parallel editing of multiple parts of plain text documents. The user first defines a set of so-called records, either manually (i.e., by selecting passages of text), by a pattern, or a combination of the two. After such a set has been defined, all the subsequent operations (for instance, selection, typing, cutting and pasting, deleting) work in parallel, on all the records within the set. In the evaluation study, it was found that the participants find simultaneous editing easy, trust the technique to do the right thing, and would likely use it for their own text editing tasks.

In a similar fashion, in [376] Toomim et al present Codelink, a prototype editor for managing duplicated (or “cloned”) source code utilizing the “linked editing” technique. Such linked editing can be thought of as an “abstraction mechanism” for end users [375]. Most software projects contain instances of duplicated code; for instance, the authors list as 21–29% of all the code in the Java SDK to be duplicated, as well as 15–25% in the Linux kernel, 19% in the X Window System, and up to 60% in internally developed projects. To cope with duplicated code, programmers usually utilize various abstractions, such as functions, methods, classes, and macros; however, such abstractions come at a cost, including being hard to create, adding overhead, being hard to change or understand, and being impossible to express. Thus, using the Codelink prototype editor, the user can keep track of, visualize, and edit instances of duplicated code within a software project, without having to recur to aforementioned abstractions. Many such instances can then be modified as a single unit, in parallel. Similarly to Miller and Myers’ approach [252], the user (1) first selects two parts deemed to be duplicates, and then (2) clicks on the “Link Selections” button. At this moment, Codelink runs a difference-detection routine, thus highlighting all the differences in the selected parts, thus allowing the user to easily view and understand said differences. If the user now (3) moves the cursor to a certain location in the first part, another cursor (“ghost cursor”) appears in the corresponding location in the second, linked part. At this moment each operation (typing, deleting, pasting, etc.) works in parallel, on all instances of duplicated code. If the user now, however, (4) unchecks the “Link Selections” button, the ghost cursor disappears, and the user can edit just one instance of duplicated code. In the evaluation study of Codelink, thirteen participants rated it higher than the abstraction method, in all of the following five metrics: maintainability, understandability, changeability, editing speed, and editing effort.

As another example, in [151] Hartmann et al described parallel source code editing which, differently from Toomim’s approach [376] which focuses on detecting snippets distributed throughout a code project, presupposes that all code alternatives have substantial overlap, and thus uses incremental synchronization and tracking.

As for commercially available packages, the Sublime Text\textsuperscript{8} text editor allows the user to multi-select and edit different parts of a text file at the same time, by having multiple cursors within the document (Figure 5.7). Thus, once the cursors have been inserted into the document, all

\textsuperscript{8}http://www.sublimetext.com
the subsequent keyboard actions are performed in parallel, for all cursors. For example, Delete deletes a character to the left of each cursor. Similarly, pressing a cursor key (Up, Down, Left, or Right) moves all the cursors accordingly, in the same direction; command “paste” inserts the object currently in the clipboard at all cursors’ present positions, and so on.

![Figure 5.7: In the coding editor Sublime Text, multiple cursors (in this case, three) allow the user to edit different parts of the textual file at the same time. Left: three cursors inserted into the document. Right: one character deleted to the left of each cursor, after the user presses the Backspace or Delete key.]

To conclude, then, in parallel text editing, it seems that most approaches are based on (1) first defining a set of such alternatives (in the form of multiple selections, multiple “records”, or multiple paragraphs of text), and then (2) implementing editing operations so that they run on multiple such alternatives, instead of on just one as is the case with conventional, single-state text editors.

5.2.5 Other Reported Approaches to Multi-Interaction

Besides subjunctive interfaces, set-based interaction, and parallel text editing, other important and influential approaches to interacting with multiple objects have been proposed in the literature.

Marks et al [239] introduce Design Galleries, an approach to the problem of parameter tweaking involving 1) batch-computing a large number of computationally-expensive graphics, each based on a vector of input parameters, 2) summarizing subjective qualities of each of these graphics as an (objective) output vector, and 3) presenting these graphics through some automatic arrangement method. The so-called dispersion method picks input vectors so that output vectors (and thus output graphics) are perceptually well-distributed. Output vectors have distance metric which is used to summarize subjectively relevant qualities of output graphics, and then arrange said graphics into “design galleries”. The user can then pan and zoom in and out of such a gallery, with the thumbnails being arranged dynamically around the central area of the gallery showing smaller thumbnails of the generated graphics. The method presented was applied to three different cases: image rendering using raytracing, volume rendering, and motion control. In a related paper [240], Marks et al describe and showcase several browsers (i.e., browsing modalities) for Design Galleries, both 2D and 3D. The 3D version is in some aspects more usable than the 2D version (for example, providing some insights that the 2D version cannot provide).

In [169] Jankun et al present a spreadsheet-like interface for exploring and displaying multidimensional parameter spaces and their associated visualizations. The visualizations themselves
5.2. MULTI-STATE INTERACTIVE VISUALIZATION

(i.e., output graphics) are organized in a two-dimensional tabular form. Axes (X, Y) show a series of values for the two chosen parameters at a time, while the rest of the parameters take on their default values. Data exploration then effectively becomes the activity of steering this 2D grid through the n-dimensional parameter space.

In [33], Bavoil et al present VisTrails, a scientific visualization system featuring formally-specified visualization “trails” which in turn generate visualization “pipelines” based on their own, separate execution instances, as well as multiple views with coordinated cameras.

In the already mentioned paper by Hartmann et al [151], the authors describe editing and execution of multiple program variations in parallel. The prototype editor (Juxtapose) is able to both (1) support variations of source code(s), as well as (2) the parallel execution of runtime codes corresponding to these source codes. For varying parameter values (a process they call “tuning”), the users utilize an external, hardware-based MIDI board with sliders. All the variables that support such tuning are marked in the source code through custom tags such as “//@RANGE 0..100”, resulting in the automatic generation of associated slider GUI widgets during runtime; variables that should be ignored by tuning are tagged with “//@IGNORE”. In the summative evaluation study, it was found that the participants readily used linked editing, and that tuning allows for faster and more complete parameter exploration.

In his masters thesis [329] Sheikholeslami presents an approach to exploring multiple alternative solutions in the domain of parametric modeling, called hysterical space, based on the recombination of parameter values implied in the interaction history thus naturally inducing a Cartesian product of said values. In a recent paper Zaman et al [416] present GEM-NI, a system for working with alternatives in generative design, with support for parallel editing, branching, merging, Cartesian products, and history recall. Different from the systems developed for this thesis, however, the interaction is predominantly graph-based i.e., indirect, as distinguished from the direct manipulation interaction techniques showcased with the prototypes developed in this work, CAMBRIA 1 and 2. Finally, in a recent paper [274] O’Donovan et al present DesignScape, a system for interactive suggestions of document layouts. The authors use non-linear inverse optimization (NIO) to find out the parameter space of a set of document layout examples, construct a lower-dimensional parameter sub-space of this larger space, apply principal component analysis (PCA) to this sub-space, and finally collect random samples in the new basis, thus now being able to generate layout resembling the ones in the initial set of document layouts.

5.2.6 Commercial Tools and Systems

As for tools and systems developed by business corporations and sold on the market for general use, or developed by design firms for their internal use, the support for multiple design alternatives is predominantly still rooted in the single-state interaction paradigm, and therefore sporadic,
improvised, and incomplete.

For instance, in the widely used Microsoft Office suite, the user can preview multiple effects at a time, applied to an object (e.g., a figure, or a table) within the document. Figure 5.8 shows multiple “artistic effects” when applied to the raster image of an abstract design alternative, and multiple “visual styles” that can be applied to a table. Yet, the underlying data model is still firmly rooted in the single-state paradigm—the user at all times works with a single document only.

CAD industry has begun introducing some support for alternatives in their tools. For example, within the Autodesk’s Showcase tool (Figure 5.9), users can configure 3D models imported from other packages such as AutoCAD, Inventor, and Revit, and then view and compare said models in parallel with different visualization settings such as materials, lightning, environmental effects, and so on.

As another example, Configuration Manager [407, p. 276] is a separate module within the SolidWorks package by Dassault Systèmes, which allows users to create, select, and view multiple configurations of parts and assemblies in a document (Figure 5.10). The underlying data model is still based on the single-state paradigm, however; varying parameters still takes place within a single document. Users can create configurations manually, or by managing them in a Microsoft Excel worksheet.

![Figure 5.8: Previews in MS Office Word 365.](image)

![Figure 5.9: Viewing two visualizations in parallel in Autodesk ShowCase. Image credit [106].](image)
5.2. **MULTI-STATE INTERACTIVE VISUALIZATION**

![Image](Figure 5.10: Three different “configurations” of a SolidWorks model. Image credit [350].)

Autodesk’s *Revit*[^9], a parametric Building Information Modeling (BIM) tool offers a functionality called “design options”, allowing the user to define different options for different parts of the current CAD model. Basically, each option defines a set of different components for a specific part of the main model. However, no parallel editing of multiple options is available; all option sets are relative to the currently-active, “primary” option set and at any given time, the user works with only one array of option sets. Again, the underlying data model is still based on the single-state paradigm, and all editing takes place within a single document.

SolidWorks *Simulation*[^10] is a collection of simulation tools from Dassault Systèmes SolidWorks Corp., which “makes evaluating [the user’s] design alternatives quick and easy to help [the user] decide which one works better”. The interface resembles a spreadsheet, where each column represents a different “what-if” scenario. The tools puts emphasis on testing the performance of the product (for example, movements, loads, stresses and forces) and making sure that the product will function correctly in the real world. While the tool offers GUI controls (e.g., sliders) to explore the design space of parametrized models (each parameter set is saved as a so-called “scenario”), the predominant paradigm of the tools is still single-state: only one scenario is active at any given time.

In another package by Dassault Systèmes, CATIA’s *Product Engineering Optimizer*[^11] “allows users to explore additional design alternatives and accurately optimize designs utilizing two complementary tools: *Design Of Experiments (DOE) & Design by Goal*”. DOE enables the user to explore design alternatives through “virtual experiments” for determining interactions between parameters, while Design by Goal helps at finding the optimal set of parameters when multi-discipline and multi-goal specifications are taken into consideration. However, the tool is strongly focused on the optimization of mathematical functions for structural design, through the use of algorithms such as simulated annealing algorithm, gradient algorithm, and others, and as such is still grounded in the single-state paradigm.

Another tool worth mentioning is the *3DVIA Home* tool (Figure 5.11) developed by Dassault

[^9]: http://www.autodesk.com/products/autodesk-revit-family/overview
Systèmes\textsuperscript{12}. With this cloud-based tool for architectural design, the end user can first select an architectural style, sketch the floor layout, and the tool will propose a number of feasible design solutions (in this example, a kitchen) automatically. The user can then further edit the proposed design solution(s), and finally create and navigate a highly realistic 3D rendering of the model. Once satisfied, the user submits the design to a design expert who then accesses the model in the cloud, leverages his or her design expertise, and creates the final design.

Choosing different layouts

![Choosing different layouts]

**Figure 5.11:** The 3DVIA commercial tool by Dassault Systèmes.

However, although the user of 3DVIA Home is able to choose among several auto-generated floor layout options, the user still works with a single design model at all times, thus very much staying within the single-state paradigm, with the exception of being able to view thumbnails of, and choose different floor layouts.

![More layouts]

**Figure 5.12:** Alternatives in video gaming. In this example, the user picked three different options for the part of the avatar's armour protecting torso. Images credit [386].

As a representative example of the use of alternatives in another application domain, that of video gaming, in Microsoft’s *Halo 4* game gamers can choose among multiple options for different parts of the avatar’s armour (Figure 5.12). In this specific example, gamers can choose different helmets, as well as parts of armour protecting torso and shoulders of the avatar, which can thus result in a large number of different armour configurations.

\textsuperscript{12}http://www.3ds.com/products-services/3dvia/space-planning-software/home/
5.3 Histories and Undo/Redo

As per the definition of “collections of alternatives” given in Chapter 7, Section 7.2.8 (see also the construct definition in the domain vocabulary, Chapter 6, Section 6.1), a collection is simply a subset of design states discovered, or reached, by the user at some point in the past. All design states contained within a collection may or may not be related through the relationship of lineage, or derivation; but in any case, the set of design alternative can always be considered as an acyclic directed graph, with a number of connected components (which are in general greater than one). From there, it follows that any collection of design alternatives naturally induces a sub-graph of
the global design history DAG, and that the question of interactive visualization of design history is therefore of significance within the context of this work. This is, of course, in addition to many references (such as e.g., citeshneiderman1996eyes, shneiderman2006creativity) confirming the value of being able to view the interaction history, for any kind of computational work.

Of numerous references, Heer et al [152] provide a fairly complete description of the design space of history tools. They detail (a) how to store and structure history data (states, actions, or both), (b) how to visualize history data (e.g., linear, branching), and (c) how to interact with history (navigation, editing, search, and export). The authors then give a description of the design and implementation of a specific graphical history tool for Tableau, a multi-dimensional database visualization system, characterized by:

(a) The use of a linear strip display for history,

(b) A hybrid state & action history model,

(c) States recorded as "VizQL" statements,

(d) Classification of each action into one of the following five categories: \textit{shelf} (add, remove, replace), \textit{data} (bin, derive field), \textit{analysis} (filter, sort), \textit{worksheet} (add, delete, duplicate), and \textit{format} (resize, style) commands,

(e) Grouping of history items into composites, and

(f) Support for merged histories.

According to the authors, interacting with design histories includes, in the most general sense [152], viewing such histories, as well as handling such histories.

- \textbf{Viewing histories}. Viewing a design history implies operations concerned with laying out, presenting, arranging, and rendering, states and state transitions; for instance, a state can be rendered as a simple text description, or a thumbnail. Furthermore, such renderings can be spatially arranged into linear timelines [188], or two dimensional structures such as set-theoretical graphs.

- \textbf{Handling histories}. Handling histories, on the other hand, includes the following actions:
  
  - Navigating a history (e.g., undo and redo, timeline slider, browsing graphs, etc.),
  
  - Editing histories (deleting past states, modifying past actions),
  
  - Annotating states and transitions (e.g., text notes, or audio snippets),
  
  - Searching and filtering states and transitions, and
  
  - Exporting and importing histories, or parts thereof.
Three major approaches to saving a history of commands exist: actions only, data states (design states) only, or a hybrid actions+data states.

(a) *Actions only.* Action-only design histories can be augmented by (a) intermediary states saved at action intervals (say, every 100 actions) in order to speed up re-application of past actions, and (b) by aggregating a number of separate actions into higher-level *hierarchical commands* [261] which might then potentially better describe the design rationale behind those actions. Conventional tools predominantly save their command history in LIFO-like data structures, in which re-application of a command in many cases leads to a loss of a sequence of past actions.

(b) *Data states only.* This approach is commonly adapted by software versioning systems, also called revision control systems [15], which store snapshots of data objects in their databases. The drawback of saving data states only is that the semantics or intent of actions is in most cases lost, which in turn means that design intent is lost too.

(c) *Hybrid actions+data.* The complete approach, of course, would save the complete DAG i.e., both (1) design states, and (2) transitions among states. This approach can however lead to excessive time and space (storage) demands.

Two additional concepts worth mentioning are *linear* and *non-linear* histories; a design history can be linear, when implemented as LIFO stacks of actions, or non-linear (in the sense of supporting branching), when the history of commands is saved as either data states, or in the hybrid actions+data form. Either of these can be applied to pCAD tools.

Other influential approaches to interacting with histories include e.g., a technique by Kurlander and Feiner [202] to visualizing and working with histories of drawing operations, through a prototype image editor named *Chimera*. The two most important ideas include (a) a strip containing before-after pairs of thumbnails, thus providing a visual indication of what has been changed in the image, and (b) using such a strip as a way to implement redo and undo. In addition, in *Chimera* the user can modify past steps “in-place”, i.e., without needing to retrace all the undo steps leading to a previous state. One can easily foresee the application of this approach in pCAD, showing “before” and “after” states for every entry in the undo stack, for each alternative.

Klemmer *et al* [188] present an approach to collecting and navigating alternatives in web design, as well as a way to interact with histories, without forcing the user to cope with a cognitively-demanding and frequently rather large derivation tree, through *history stub-branching*. In addition, in the evaluation study, the participants (professional web designers) suggested features for simultaneous comparison and merging of history states. Su *et al* [358, 359] examine operation history within the context of 2D vector graphics, and presented as interactive storyboards, thus improving interaction by visualizing the history in spatial context, instead of as a linear list of past operations.
That way, the authors achieve the so-called “non-sequential” undo which is minimally destructive, in the sense of not cancelling actions on the top of the undo stack. Grossman et al [143] present Chronicle, a new system built upon an open-source image editor for tracking document revision histories based on before-after pairs of document and GUI states, an elaborate timeline subsystem, and the ability to play back videos of interaction (i.e., document modification sequences) in order to preserve the same for learning purposes.

More generally, design history can be visualized as “usage patterns” [158, 176]; linear timelines with local and global layers [109]; semantic zoom on nodes [28, 174]; 2D matrices of clustered nodes [239, 144, 335]; nodes linked into trees [99, 174]; and nodes linked into graphs [28, 408]. Sub-graphs representing parts of histories can be copied and inserted at a different points in the global design history [144]; likewise, one can foresee an analogous approach in multi-state pCAD, where the user manipulates the design history graph directly, thus being able to re-use operator sequences at different point of entry in the global design history. Another term for this functionality includes “path reuse”, as one of the “opportunities” to amplify designer’s capabilities using a computational tool [406].

5.4 Chapter Summary

This chapter contains a review of related work that is most pertinent to the overall objective of this thesis, with strong emphasis on the topics dealing with interaction and visualization. To this end, I focused on the following general themes: (1) existing taxonomies of interaction tasks and techniques, that provide a comprehensive enumeration of interaction primitives (and which will come in handy when proposing design guidelines in Chapter 10), (2) existing approaches to interacting with multiple computational objects in parallel (e.g., subjunctive interfaces, and set-based interaction), and (3) interacting with design history.

Due to space limitations and the subject matter of this thesis’ work (i.e., interaction design), I necessarily focused on related work pertaining to interactivity in the context of alternatives. However, a more complete presentation might include material e.g., on versioning, revision control and merging [15], computational comparison and differencing [275, 322, 133, 352], and other research pertaining in most cases to data structures underlying design histories, as well as ways to store, compare, and transform design states.
Chapter 6

Phase A: Pre-Design

Previous chapters introduced the theoretical framework i.e., the conceptual structure underlying this work (Chapter 2), the adopted research methodology or research approach (Chapter 3), as well as described the relevant prior work (Chapter 5). This phase of research, or pre-design phase (Phase A), can be considered to be the first phase of research in the strict or even chronological sense\(^1\), i.e., marking the initiation of research whose overall aim is to broadly explore the research problem addressed by this thesis. My two immediate goals in this phase were: (a) to better understand, delimit, and conceptualize the space of feasible solutions (i.e., the space of interaction designs that effectively solve the problem of working with multiple design alternatives), and (b) to obtain a broad, comprehensive sample of such solutions.

As for the first goal, in order to generate a relatively large number of different solution proposals so as to better understand the problem, I temporarily suspended my judgement so as not to evaluate any such solution proposal in order to generate as many potentially feasible solutions as is possible, thus affording multiple entrance points in the associated interaction design space. Following Osborn [276], by avoiding premature evaluation a designer can avoid hindering the creativity, and brainstorm more easily. In the same vein, Dow et al [103] state that "immediate feedback sets the focus on refinement, whereas postponing a critique until after creating multiple designs encourages more divergence". In this phase, therefore, I refrained from immediate feedback and evaluation of HCI designs, in order to expand my consideration as much as possible. Accordingly, I devised a good number of different approaches to visualizing data, and to interacting with said data. To that end, I first gathered a number of possible approaches through literature review, collaboration with my academic supervisors, reflected on the problem, and then developed a number of design solution proposals in the form of low-fidelity sketches, wire-frame GUI mockups, and low-level (yet still working i.e., interactive) graphical user interfaces.

\(^1\)I remind the user that Related Work has been updated continuously, throughout the duration of this research.
6.1 Domain Vocabulary

As mentioned in the chapter on the overall research approach adopted for this thesis (see Chapter 3), I consider the domain vocabulary to be one of the major, continuously-updated “research artifacts” produced over the course of any design-based research. (See Glossary in Appendix 12.5 on page 414.) Therefore, over the duration of the research, I kept returning to this section, updating entries as needed, based on the feedback from other parts of the research.

As a concise summary of the dictionary, Figure 6.1 shows a diagram of the main concepts present in the contextual model. Moving in the bottom-up direction, one can have small and large iterations, however only a sufficiently different iteration can also be considered to be a design alternative [367]. In addition, variation or variant can be considered to be synonymous with alternative; accordingly, each design alternative, solution variant, design solution variant, or design solution variation can be considered to be a type of design solution, which in turn is a kind of representation.

6.2 Pre-Design Activities

Pre-design activities include coming up with the initial domain vocabulary, defining the initial list of design requirements, identifying possible approaches from the literature, sketching the initial tentative GUIs, and developing throwaway interactive prototypes.

To start with, what are the design requirements on the tools supporting alternatives-enabled work? I begin with the following set of intentionally very broad, initial (or “seed”) requirements for alternatives-enabled pCAD systems (Figure 6.2). According to this list, an alternatives-enabled tool should offer the following fundamental, general capability: interact with multiple design alternatives in parallel. This key, central capability can be further divided into three sub-capabilities: view, edit, and manage sets of design alternatives.
6.2. PRE-DESIGN ACTIVITIES

1. Interact with multiple design alternatives in parallel.
   1.1 View many alternatives at once.
   1.2 Edit many alternatives in parallel.
   1.3 Manage collections of alternatives.

Figure 6.2: Initial (seed) design requirements on alternatives-enabled tools.

Therefore, the set of initial requirements was small and simple enough: the user should interact with multiple design alternatives in some way (requirement 1). This overarching goal was further decomposed into three constituent yet still high-level sub-requirements of: 1.1) the user should be able to view, 1.2) edit, and 1.3) manage (collections of) multiple alternatives in parallel. As the designing proceeds within the framework of research-through-design (RtD)\(^2\), this list of design requirements will be developed, refined, and elaborated further.

6.2.1 Cataloguing Candidate Approaches Through Literature Review

This section enumerates a number of promising and potentially effective techniques and approaches that have been reported in the literature.

6.2.1.1 Interaction Paradigms and Interaction Styles

According to two principal classifications [1][333, p. 193] of interaction paradigms, and considering the research problem stated in Chapter 1, the paradigm adopted for solving this thesis’ research problem is the graphical user interface (GUI) paradigm, based on the direct manipulation interaction style, which is characterized by “the feeling of involvement directly with a world of objects” [165], thus significantly reducing the gulf and execution and gulf of evaluation\(^3\).

6.2.1.2 Interaction Techniques

Through the review of related work (Chapter 5), in an early effort to satisfy the set of initial design requirements (Figure 6.2), I obtained a preliminary list of feasible, promising approaches to visualize, edit, and interact with, pertinent data.

\(^2\)See Chapter 3, Section 3.2.
\(^3\)See Chapter 2, § 2.7 discussing cognitive artifacts and gulfs of execution and evaluation.
(1) **View many alternatives at once.** Inspiration for displaying many alternatives in parallel can be found e.g., in the “small multiples” approach by Tufte [380, 381], where scaled-down images of alternatives facilitate their comparison, Roberts’ [309] use of multiple views in visualization, Busking’s [56] visualization of shapes using three different modes (juxtaposition, superposition, and timeline). Lunzer provides several ideas in the form of visualizing the Cartesian product of sets of parameter values [218] (where each set is represented by a column of values), multiple states rendered on top of each other within a single view [220], or within perspectively-distorted planes [223], clusters of alphanumerical data displayed within individual, multiplexed [222, 226] GUI widgets [223, 172, 229] using juxtaposition, parallel search results [223], as well as multiplexed GUI widgets that display 2D or raster image data either in juxtaposition [220, 226, 172, 225] or superposition [225]. Terry displays multiple raster images either as a set of juxtaposed previews [369], or as separate slices within a pie [370]. Marks et al [239] display multiple alternatives either as a 3 × 8 grid of (juxtaposed) thumbnails, or as a scattered set of smaller (automatically arranged and clickable) thumbnails in the central area of the view, framed by somewhat larger thumbnails linked by a guideline to the corresponding smaller thumbnail in the central area. Bavoil et al [33] display alternatives using multiple views with coordinated cameras. Hartmann et al [151] utilize either a grid of (juxtaposed) alternatives, or the “tabbed view” where each alternative is displayed within its own tab view. Finally, as a variant of juxtaposition, Jankun et al [169] present multiple alternatives as separate cells within the 2D grid. Considering the wide acceptance of these approaches in the general HCI design community, they represent good candidates for initial consideration in alternatives-enabled pCAD tools.

![Figure 6.3: Multiple alternatives arranged as a sequence in a rectilinear grid, with two different expression types for alternatives (thumbnails and glyphs).](image)

As an illustrative example, Figure 6.3 shows two possible ways to arrange alternatives into a grid sequence: a grid of thumbnails, and a grid of glyphs.

(2) **Edit many alternatives in parallel.** Beyond visualizing multiple objects in parallel as well as exploring parametric spaces in parallel, Toomim et al [376] describe simultaneous editing of plain text (i.e., source code) based on detecting snippets of code distributed throughout a code
project. Hartmann et al [151] on the other hand present parallel source code editing based on incremental synchronization and tracking. At least one commercial source code editor (Sublime Text) allows its users to insert multiple, synchronized cursors at different locations within a text file, allowing multiplexed input actions (for example, pressing Delete key will delete a character to the left of all such cursors).

(3) **Manage collections of alternatives.** This includes ideas for generating, and managing sets of collections of alternatives. Terry [369, 370] presents a way to interactively generate sets of alternatives using the so-called “parameter spectrum” approach, and which progressively and selectively filters and produces a range of modified alternatives (or previews thereof). Marks et al [239] provide a way to generate a collection of alternatives by batch-computing a large number of renderings. The spreadsheet-like interface by Jankun et al [169] allows for a generation of a set of alternatives defined by the two sets of values, one set per parameter. Hartmann et al [151] provide both an approach to parallel source code editing based on synchronization, and an approach to varying variables’ values by using either GUI slider controls or a hardware MIDI board with sliders.

![Early-stage interaction sketches.](image)

**Figure 6.4:** Early-stage interaction sketches. In (a), parallel editing of analogous parts for two different yet related designs: a bicycle design, and a wheelbarrow design. In (b), a number of different bicycle design alternatives, as well as ways to pass analogous bicycle parts from one design to another.

### 6.2.2 GUI Sketches and Mockups

Again, my first method for obtaining a broad, comprehensive sample of potential IxD solutions, as described in § 6.2.1, was to catalogue and then winnow candidate approaches through literature review. However, other means to **brainstorm** or **ideate** are possible. Figure 6.4 shows two examples of ideation having been done by manually sketching possible approaches using markers on

---

4 http://www.sublimetext.com
a whiteboard. Using this technique, the overall goal was to determine basic mechanisms through which, by combining different parts of certain artefacts (bicycles, wheelbarrows, ...), one could come up with a multitude of different yet related designs.

Figures 6.5 and 6.6 show, on the other hand, a storyboard developed with the help [409] of Brad Holtz, president & CEO of Cyon Research and two other experts from The Congress on the Future of Engineering Software\(^5\), an CAD/AEC event held in year 2013. I first drew, in collaboration with my academic supervisors, a number of different bicycle designs, scanned them, and then manipulated bicycle parts by editing corresponding parts of raster images. The storyboard was then submitted to the aforementioned CAD experts, modified according to their input, and resubmitted. This iterative process was repeated several times, over the course of several weeks, until the experts at COFES were satisfied with the illustrated interaction mechanisms.

In addition to hand-drawn GUI sketches, I also used a commercial tool\(^6\) for creating, and interacting with, wireframe mock-ups of possible graphical user interfaces. This and similar tools allow for rapid creation and positioning of GUI elements (windows, panes, list boxes, labels, links, text controls, images, and so on) on the working form. Interaction can be simulated by defining “hot-spots” on a form, which when clicked on, results in the opening & displaying of another form, with different placement of its GUI elements. For example, clicking on the 'Edit' command (either the link or the menu) in form (a) on Fig. 6.7, leads to the displaying of the form (b), thus simulating the interaction which would eventually take place in a concrete implementation.

### 6.2.3 Throwaway Interactive Prototypes

Using rapid prototype development techniques [245], I also developed a fair number (around one hundred) of early graphical user interfaces (Fig. 6.8), featuring promising visualization and interaction capabilities. Each version of these early prototypes was building on previous ones, with increasing number of refined and new features. For instance, a version might offer parallel highlighting on mouse over events; the subsequent version might also implement parallel select on mouse-down events, and so on.

---

\(^5\)www.cofes.com

\(^6\)Balsamiq, http://balsamiq.com/
6.2. PRE-DESIGN ACTIVITIES

Figure 6.5: Early-stage storyboard (hand sketching + raster image editing), frames 1–24.
Figure 6.6: (Continued from Fig. 6.5.) Early-stage storyboard (hand sketching + raster image editing), frames 25–36.
Figure 6.7: Early wireframe mockups of graphical user interfaces. (a) Explorer view (latest versions), (b) Explorer view (version history), (c) Parallel editor (juxtaposition mode), and (d) Parallel editor (superposition mode).
Figure 6.8: A sample of rapidly-developed, low-fidelity prototypes. (Listed in the chronological order of development).
While all lower-fidelity prototypes did help me to learn more about, and better explore and understand the research problem, I will now describe in some detail the following four prototypes, and which significantly influenced further research and development.

6.2.3.1 Prototype #1: “2 × 2 Parametric Designer”

The throwaway prototype shown in Figure 6.8 turned out to be the most influential design in all the subsequent design and development of later prototypes, because it featured a number of capabilities that have shown to be potentially effective for their application in alternatives-enabled tools, as described below.

![Diagram of 2 × 2 Parametric Designer]

**Figure 6.9:** An early, throwaway prototype nick-named “2 × 2 parametric designer”, featuring: (1) multiple views (one per design alternative), which can be active or inactive (as indicated by the toggle button in top left corner), (2) items linked through unique identifiers, (3) brushing (mouse-over) of all the linked instances (as shown in the case of the curve, highlighted in bright red), and (4) selection of multiple graphical items in parallel (the curve selected in all three active views, indicated by dashed line contour).

This prototype featured a fixed 2×2 grid of four “parametric 2D editors” in which the user could perform the following tasks:

(a) **Linking and unlinking alternatives.** Link and unlink individual alternatives, by clicking on the large red “toggle” button in the upper left corner of each alternative.
(b) **Parallel drawing.** Draw basic graphic elements: closed and open Bézier curves, as well as straight lines. Drawing could be done in parallel, for all linked alternatives. For instance, drawing a line in one editor would automatically draw the same line in the remaining three editors. Unlinked alternatives did not receive any drawing actions.

(c) **Brushing and linking.** An implementation of brushing and linking of individual items, on mouse-over actions. Thus, if the mouse pointer passed over an item in any of the four alternatives, all the corresponding items in all the other alternatives (linked or unlinked) were rendered in red, for visual emphasis. This helped the user to establish correspondence between related (or “linked”) items.

(d) **Parallel selection.** Select command was parallelized. The action of selecting an item in the currently active alternative led to the action of selecting all the corresponding items in all the other linked alternatives. (Thus, the item was not selected in unlinked alternatives.)

(e) **Parallel editing.** Parallel editing, in the form of dragging the control points defining a line or a curve, or in the form of dragging the whole item (by clicking on its body). The action of dragging a control point in one active alternative led to the simultaneous action of moving the corresponding control point in all the linked alternatives, by the same amount of pixels. Thus, as a line or curve (defined by this control point) was being changed, all the corresponding lines and curves in the remaining three editors were being changed simultaneously as well.

### 6.2.3.2 Prototype #2: “2D Font Designer”

This prototype merits its own sub-section due to the implementation of several interesting ideas which also influenced further development and research. For instance, it clarified the concept of *collections* of alternatives, the concept of design history, merging (or “unification”) of selected designs, and (random) perturbation of the current (main) design.
6.2. PRE-DESIGN ACTIVITIES

In the upper part of the application, the user could automatically generate a number of alternatives. Each action generates an user-defined number of randomly re-sampled versions of the current ("master") design shown in the central-left part of the application (large letter “S”). Selecting any of the alternatives in the Alternatives or History panel, would make that alternative the active one.

Furthermore, editing the active model (i.e., performing any edit action, such as displacing a control point) resulted in its immediate placement into the “History” panel (shown at the bottom left corner). Within the History panel, the user could additionally select multiple designs, which would then automatically result in the immediate creation (based on simple averaging of coordinates of the four control points) of the “merged” model shown in the “Unification” panel (bottom right corner).

Additionally, the central-right part of the application contained both a basic “property editor”, with individual values for each parameter, and the parameterization graph (the “Parameterization” tab, center right, not shown in the figure).

6.2.3.3 Prototype #3: “Subjunctive 3D Points”

I developed another throwaway prototype, shown in Figure 6.11, to experiment with how one might interact with 3D alternatives. In this prototype, the left part (the “Subjunctive model” panel) of the
The application was reserved for the visualization of “3D subjunctive points”, i.e., “plural” or “subjunctive” 3D points which could then “collapse” into any of the concrete points shown within their corresponding bounding boxes in the “Collapsed model views” panel to the right of the window.

**Figure 6.11:** Interactive prototype for 3D modelling (“Subjunctive 3D Points”).

As shown in Figure 6.11, the master model has in this case collapsed into four “concrete” models (shown in the upper right part of the application, and arranged into a $2 \times 2$ grid). All 3D views (main view, and four collapsed-model views) had synchronized cameras, i.e., the act of rotating and translating 3D models would preserve the vantage points for all five models. Optionally, bottom right panels showed additional information about the main subjunctive model.

### 6.2.3.4 Prototype #4: “Taming Combinatorial Explosion”

In another prototype, I explored the issue of how to manage Cartesian products of parameter values and the ensuing combinatorial explosion of all possible permutations; the screenshot is shown in Figure 6.12.
6.2. PRE-DESIGN ACTIVITIES

6.2.4 Discussion

From the activity of cataloguing promising approaches in the literature, creating GUI sketches and mockups, as well as developing lower-fidelity interactive prototypes as a means of generating multiple (potential and promising) HCI designs for alternatives-enabled pCAD tools, I synthesize the following important principles that merit further investigation in the subsequent phases of research:

(1) **Having multiple alternatives on screen.** This subsumes and details the realization that multiple views (one view per one alternative, i.e., drawing) should be present in the tool. Limits to human cognition are well-known: people can only remember and keep active a very small
number of memory states. The universal work-around for this limitation is off-loading tasks to cognitive artifacts and external representations. Thus, in design exploration, designers need to be able to see multiple design alternatives at once, in order to compare and combine them. Furthermore, a view should be “active” (“linked”), or “passive” (“unlinked”), capturing the notion of how an alternative is responding to changes in other linked alternatives.

(2) Maintaining correspondences (links) between graphical items. In general, \( N \) alternatives are displayed on screen. If a graphical item (say, a curve) exists in one alternatives, then that same curve might (but does not have to) exist in other alternatives. Therefore, a graphical item could have multiple instances in multiple drawings. An alternatives-enabled pCAD tools must have the means to track correspondences between graphical items, for example by associating a unique identifier to each of such instances. To illustrate, if a curve has been added to say six active alternatives, that same curve would then have the same unique ID (say, 2378) in all six alternatives. This technique of associating unique IDs is commonly used in matching, automatic comparison, and differencing approaches (see e.g., [15, 352]).

(3) Brushing (mouse-over) of linked items. Again, if multiple instances of a graphical item exist in multiple alternatives (drawings), when the user moves the mouse pointer over one such instance, all the corresponding instances in other alternatives get highlighted in some way, for instance by being rendered using a bright red colour for visual emphasis. This helps the user to match graphical items in multiple alternatives, and thus better understand their structural differences and similarities.

(4) Simultaneous selection of items in linked alternatives. Multiple instances of an items might be associated with the same unique identifier, in different alternatives. If that is the case, then we say that such items are “linked” or “associated”. If multiple such instances of a graphical item exist in multiple alternatives (drawings), when the user selects one such instance, all the corresponding instances in other alternatives should be selected as well (if so desired by the user). This technique is a necessary prerequisite for the implementation of parallel editing techniques.

(5) Simultaneous modification of items. Once the user selects multiple item instances in multiple alternatives, such instances can be modified in parallel. For instance, if the user edits (i.e., modifies line thickness, colour, ...) a graphical item in one alternative, then all the corresponding items follow with the same change.

The low-fidelity prototype shown in Figure 6.8 illustrates all the aforementioned principles well: (a) four design alternative on screen, arranged in a 2x2 grid, with three active and one inactive (as indicated by the button in top left corner); (b) unique integer identifiers, linking items on screen,
shown; (c) one particular item (a curve) hovered-over with the mouse pointer, thus highlighted in bright red; and (d) all instances of that curve selected in all the active alternatives. In addition, if a control point is dragged in one alternative, then all the corresponding control points in all the other alternatives are being moved by the same amount, thus modifying the item in question (the curve) in parallel as well.

### 6.3 Updated Design Requirements

Based on my pre-design findings and activities, I will now amend design requirements as follows (bold italic for new requirements, and strikethrough for removed requirements), by claiming that an alternatives-enabled pCAD system should offer its user the ability to:

1. Interact with multiple design alternatives at once.
   1. View many alternatives at once:
      1. Lay out alternatives in different ways:
         1. *Lay out alternatives in different ways:*
         1. *Into groups previously created manually by the user.*
         2. *Into groups created computationally by applying various distance metrics.*
         3. Layered (superposed) on top of each other.
         4. In a grid.
         5. In a data table, with parameters listed by column.

2. Edit many alternatives at once, by performing operations on one or more items in parallel, in multiple alternatives:
   1. Select.
   2. Insert.
   3. Modify.
      1. Standard operations: translate, scale, rotate, change visual attributes (colour, pen width, fill colour, etc.)
   4. Delete.

3. Manage collections of alternatives:
   1. Copy alternatives from one collection into another.
Thus, in concordance with the research-through-design (RtD) approach adopted for this thesis, the set of design requirements underwent a series of iterative changes through pre-design activities, leading from its very basic, “seed” form shown in Figure 6.2 to its further elaboration, detailing, and maturation, as shown in Figure 6.13.

6.4 Chapter Summary

This chapter described my process of generating as many potentially feasible solutions (and their parts) as is possible, in order to better understand the space of interaction designs. In order to do that, I suspended my judgement and postponed critique which are typical for designing in the strict sense (hence the name of this phase: "pre-design").

To obtain such a list of feasible solutions, I: (1) catalogued existing approaches from the relevant literature, (2) sketched and mocked-up preliminary GUIs, as well as (3) developed a number of throwaway interactive prototypes.

The experience gained through these activities has allowed me to expand and detail the initial set of design requirements (Figure 6.13), as well as to develop initial understanding of what might constitute effective approaches and techniques to alternatives-enabled work. This experience and understanding now enables me to proceed with conceptual design (see the next chapter, Chapter 7), to be followed by the design, implementation and evaluation of high-fidelity prototypes CAMBRIA 1 and CAMBRIA 2 (Chapters 8 and 9).
Chapter 7

Phase B: Conceptual Design

Having acquired a basic understanding of the interaction design problem at hand, I proceed with the development of a conceptual model [328, p. 51], or conceptual design [268] for the system or tool being designed\(^1\). Such a model represents a very high-level, abstract representation of the concepts (constructs, entities) present in the solution, as well as their relationships, and should [177] present:

1. An idealized view of how the system works, with a hope that the end users will internalize the model,
2. A basic (coarse-grained) ontology of the concepts involved in the system, and
3. Ways to accomplish users' tasks that the system is supposed to support.

A conceptual model is not to be confused with the description of the user interface itself, or with the data model used by the tool; a conceptual model operates at a higher level, and directs the subsequent development of both the user interface, and the data model. A conceptual model should at the very least include the description of the following [177][328, p. 51]:

1. Metaphors (or analogies) used,
2. Concepts,
3. Relationships among concepts, and
4. Mapping of concepts to tasks.

\(^1\)I should note here that the activity of design does not in general proceed in this sequential order (from design problem, to design solution). Rather, both the design problem and any of its solutions can be considered an ensemble that is being developed in parallel, or in any opportunistic order. See Section 2.6.3 on page 28.
As for rationale for using metaphors when designing interactive systems, they help reduce the large underlying space of interaction designs, and express the GUI and its affordances in terms of some familiar environments [263, 65]. Two historically important and well-known instances of metaphors are the business office metaphor which greatly influenced all modern operating systems, or the ledger metaphor which influenced modern spreadsheet applications. A good metaphor is one that is “apt, matches the system well, and is easy to understand” [398, Ch. 10]. The benefits of using metaphors extend not only to system designers but to users as well, because they can reuse prior knowledge from a familiar domain, and apply it to a new domain (that is, the domain addressed by the system) while using and learning the tool [178, 205, 263].

Several metaphors might be used concurrently for a system design [328, p. 60]. Additionally, some systems follow a global metaphor, which is supported by a number of auxiliary metaphors [66, 98, 263]. For instance, the global metaphor for most operating systems, as has already been mentioned above, is that of a business office, which is then supported by a number of auxiliary metaphors such as desktops, files, folders, clipboards, garbage bins, and menus [263].

As for concepts, they can be defined as high-level entities making up a conceptual model [328, p. 51][177]. Every concept has a set of attributes and a set of methods (or actions) attached to it. Relationships among concepts account for how concepts relate to each other. Finally, mappings translate concepts into tasks, which in turn are the elements of the task-domain that the system being designed is supposed to support.

### 7.1 Metaphors Chosen

I made the following design decisions on what metaphors could be effective when specifying or building alternatives-enabled pCAD computational tools:

1. **Design gallery** (global metaphor),
2. **Tracing paper** (auxiliary metaphor), and
3. **Evolutionary tree** (auxiliary metaphor).

#### 7.1.1 Metaphor #1: Design Gallery

The primary metaphor is that of a gallery (or a collection) of cognate design artifacts, allowing one to view, compare, and evaluate many such artifacts at once. In real-world practices, a gallery may also take on the form of a shelf stacked with physical models or maquettes (such as those found in architectural studios), or multiple sketches hand-drawn over multiple pages in a drawing sketchbook. Figure 7.1 shows a real gallery (Vitra Design Museum) with a number of building models set up for public display.
7.1. METAPHORS CHOSEN

Figure 7.1: Metaphor #1: Design gallery. Reproduced by permission © Vitra Design Museum. Photo: Bettina Matthiesen [242].

Having multiple designs within the field of view eliminates the need to retrieve and re-store design states from and to long-term memory. In other words, having multiple visible design states takes advantage of the human visual system (see Section 4.1.7), by alleviating the “limited transfer problem” [103], and promoting visual comparison thus revealing key relations [129] and common structure [128]. Finally, art galleries are a popular and familiar concept, thus allowing reuse of knowledge, expectations, and typical activities such as browsing and casual enjoyment. I should also mention here the widely known Design Galleries paper by Marks et al [239], which presents an approach to summarizing subjectively relevant qualities of graphics, and then arranging said graphics into interactive “design galleries”.

7.1.2 Metaphor #2: Sketching and Tracing Paper

This metaphor adapts sketchbooks and tracing paper to view multiple designs superposed on top of each other in a two-dimensional plane, following a certain z-order. Designers have traditionally used the tracing paper to easily view and copy multiple drawings (or parts thereof) at the same time. For example, in order to create a new variant of a drawing, the designer would place a tracing paper onto an existing design, simply copy (re-trace) the lines that should remain unchanged, and then draw anew only the parts of the design that require modification.
7.1.3 Metaphor #3: Evolutionary Tree

The third metaphor, that of the lineage tree or evolutionary tree, also sometimes called phylogenetic tree or tree of life which captures early vs. late stages in the development of a species or multiple species (Figure 7.3).

As for rationale for evolutionary trees, designs evolve in time; they are continuously being refined, branched, and recombinated by the designer. Evolutionary trees capture this notion well. This must be qualified by the fact that designs, or partial solutions contained within, can also be recombinated, merged, or unified. For instance, Jones [179] talks about the process of transformation, or the design process of recombinating partial solutions. In revision control, merge refers to the process of recombinating two different files, or versions thereof, by determining differences and similarities (i.e., identical sections), and then producing a third, merged file. To capture this notion, the metaphor of evolutionary trees has to be amended by the concept of “recombinating designs”, which in effect results with a more general directed acyclic graph (see Figure 7.12 on page 141), as compared to the tree shown in Figure 7.3.

Continuing with the metaphor of evolutionary biology, this corresponds to the process of hybridization in which two animals or plants from two different taxonomic groups cross-breed, thus resulting in an offspring with recombinated properties. If organisms are sufficiently similar, hybrids can be viable. And vice versa, the farther organisms are in their taxonomic classification, the less likely hybridization is. In design, this can be interpreted as the feasibility of designs being inversely proportional to the distance in their taxonomic classification.
7.2 Main Concepts and Their Relationships

The high-level concepts comprising the conceptual model that I developed, as well as their relationships, include:

1. **Variables**,
2. **Items** (simple and composite),
3. **Constraints**,
4. **Design models** (i.e., set of items and associated constraints), and
5. **Design history**.

Figure 7.3: An early evolutionary tree created by E. Hӓckel in 1866 [147].
7.2.1 Variables

Variables are the second-lowest-level building blocks of the conceptual model, used to build more complex (i.e., composite) entities called items (see Section 7.2.2). Variables have facets, or the lowest-level elements describing various aspects of variables. Facets for a variable $v$ might include for instance $v$.name, $v$.type, $v$.value, and others, whereby

- $v$.name: name of the variable $v$. For example, $v$.name = "width".
- $v$.type: can be number, boolean, string, reference (i.e., pointer), or an user-defined type. For example, $v$.type = "number".
- $v$.value (or simply $v$): a quantity of type $v$.type which can be read and/or written to. For example, $v$.value = "6.263".
- other facets, specific to a variable.

7.2.2 Items

One level up from variables on the abstraction scale are items. Basically, items are sets or agglomerations of variables. In this work, one of the requirements is that all items must have some visual expression i.e., can be rendered as a visual (i.e., with spatial extents) object on the screen, such as a line, circle, Bézier curve, or text. There are two main types of items: simple items, and composite items. Figure 7.4 depicts an abstracted rendering of an item, delineated by a dashed rectangle representing the item’s axis-aligned minimum bounding box (BB).

![Figure 7.4: An abstracted visualization of an item.](image)

Each item type exposes a set of variables called properties, such as: title, position (relative to the item’s BB upper left corner), width and height (relative to the item’s BB), penWidth, penColor, fillColor, fillTransparency, and others. In terms of access, an item’s property can be a (1) read-only, (2) write-only, or (3) read-and-write property.

Simple items. Lowest-level items, i.e., those that cannot be decomposed into even simpler ones, are called simple items. Different types of simple items exist. Figure 7.5 depicts some of
7.2. MAIN CONCEPTS AND THEIR RELATIONSHIPS

them, such as lines, curves, geometric shapes (e.g., a rectangle, triangle, circle, and so on), and text.

Figure 7.5: Simple item types, along with their axis-aligned minimum bounding boxes (BB): line, curve, rectangle, text.

Composite items. Composite items (Figure 7.6) are those created by grouping together other items (simple or composite).

Figure 7.6: Items can be composite i.e., modular. In this example, item 3 comprises three sub-items 3.1, 3.2, and 3.3, which can be either simple or composite.

Subjunctive items. In computational matching, relationships among objects can be maintained by assigning unique part identifiers to equivalent parts across multiple scopes. This technique of associating unique IDs is commonly used in matching, automatic comparison, and differencing approaches (see e.g., [15, 352]). Likewise, in this thesis, I decided to adopt unique identifiers as the means to track correspondences between graphical items. To illustrate, if a line has been added to say four active alternatives, that same line would then be given the same unique ID (say, 2378) in all four alternatives. This simple mechanism then later one enables various “parallel” operations, such as brushing, linking, and parallel editing operations.

To clarify the terminology somewhat, a subjunctive item is an item (simple or composite) that may or may not have corresponding, matching items in other alternatives. Thus all items, simple or composite, are by definition subjunctive as well, only that they may have 1, 2, or more instances, in multiple alternatives. If the user recurs to parallel operations, and if various instances of an item exist in other active alternatives, then the tool should use (i.e., search) for unique identifiers in all active alternatives, thus affecting all instances of the item, depending on what parallel operations has been chosen by the user.
7.2.3 Constraints

The behaviour (i.e., shape, layout, etc.) of items are being governed by constraints. Intuitively, constraints are the means to somehow relate one item to another, or even a set of items to another set of items, by referencing properties of those items. Constraints have been found to be successful for the layout of graphical elements, for instance in user interface design [260, p. 13].

### 7.2.3.1 One-Way Constraints

One-way (or unidirectional) constraints allow items to influence other items in one direction only. That is, the flow of control goes one item to another, but not vice versa (Figure 7.7).

![Figure 7.7: One-way (unidirectional) constraints are indicated by one arrowhead.](image)

Formally, if we suppose that \( y \) is a property of item 2 in Figure 7.7, and a set of variables \( \{x_1, \ldots, x_n\} \) are properties of item 1, we can write

\[
y \leftarrow f(x_1, \ldots, x_n)
\]

Both forms (right-to-left flow, or left-to-right flow) are equivalent. Here \( f(\cdot) \) indicates any means for computing the value of \( y \), such as for example mathematical expressions, functions, or general-purpose computational methods.

**One-Way Dataflow Constraints.** One important subclass of one-way constraints are the so-called *one-way dataflow constraints*. The modifier *dataflow* indicates that computation takes place at interactive rates, i.e., synchronously as the user interacts with the model. For this constraint type, whenever one of the input variables \( \{x_1, \ldots, x_n\} \) changes, the function \( f \) is automatically recomputed on-the-fly (i.e., during the interaction with the model) and its result immediately stored in \( y \). Computational models utilizing one-way dataflow constraints only are usually represented as *directed acyclic graphs* (or DAGs), or graphs with no directed cycles (Fig. 7.8). Such graphs contain no directed cycles because they would, in principle, lead to infinite looping during the constraint evaluation process.
7.2. MAIN CONCEPTS AND THEIR RELATIONSHIPS

Figure 7.8: A directed acyclic graph (DAG) is a directed graph with no directed cycles. In this example, adding an one-way dataflow constraint from item 2 to item 6 would create a cycle (infinite loop) \( 2 \rightarrow 6 \rightarrow 4 \rightarrow 2 \rightarrow 6 \rightarrow \ldots \) and this graph would therefore cease to be a DAG.

Computational models based on DAGs enable simple, compact and declarative descriptions of design states, state transitions based on differences, and efficient state update algorithms [407]. One potential shortcoming of models based on DAGs is the high threshold to use [260], due to users having to pay attention to keeping the constraints (i.e., the induced directed graph) free of cycles at all times. However, benefits outweigh shortcomings of models based on DAGs, and due to the efficiency of state update and data propagation algorithms as well as the relative simplicity of implementing such models, I decided to adopt the same for developing the interactive prototypes described in this work.

7.2.3.2 Two-Way Constraints

For the sake of completeness, two-way (or bidirectional) constraints are a generalization of one-way constraints, in the sense that the computation can flow in both directions, from one set of variables to another set of variables, and vice versa.

Figure 7.9: Bidirectional constraints are indicated either by two arrows, or no arrow.

Formally, if we suppose again that \( y \) is a property of item 2 in Figure 7.9, and a set of variables \( \{x_1, \ldots, x_n\} \) are properties of item 1, we can write

\[
y = f(x_1, \ldots, x_n)
\]

Thus, just like in the case of one-way constraints, if any of the input variables \( x_1, \ldots, x_n \)
changes, function $f$ is recomputed and its result stored in $y$. However, for two-way constraints, the opposite direction is also possible, i.e., if $y$ changes, the values of variables $x_1, \ldots, x_n$ are being recomputed as well. I will not be pursuing two-way constraints in this work, but I mention them here for completeness, and as a possibility for future work.

### 7.2.3.3 Constraints Constrain Items

Finally, in this bottom-up approach to constructing the concepts, constraints weave a web of controls and dependencies among items which govern the shape, appearance, and other aspects of all the elements in the design model. Figure 7.10 depicts a number of items being ruled by a number of constraints, of both the one-way and two-way kind.

![Diagram of items and constraints](image)

**Figure 7.10:** The behaviour and geometric configuration of items is governed by constraints (here depicted as red dotted lines).

### 7.2.3.4 Constraint Solver

The expression *constraint solving* (also: *constraint satisfaction*) denotes the computational activity of bringing a set of constrained variables to a consistent state i.e., a state where no constraints are being violated. In other words, where all the constraints are *satisfied*.

In principle, one single variable modification results (or might result) in a cascade of modifications to variables elsewhere, due to the interdependency of variables induced by constraints. This cascade of re-computations of variables present in a model is performed by *constraint solvers*, or computational modules for evaluating constraints and recomputing the values stored in model’s variables. The role of constraint solvers, therefore, is to assign values to a model’s variables so that constraints stay satisfied. Many different constraint solver types exist; a complete treatment of constraint satisfaction solvers is beyond the scope for this work, but for excellent accounts in this area of research you might refer for example to [201, 379, 95].
7.2. MAIN CONCEPTS AND THEIR RELATIONSHIPS

7.2.4 Design Model = Items + Constraints

What is usually called a ‘design model’ or simply a ‘model’, then, is just a term used to designate the set of top-level items that the user is currently working with, along with all the associated constraints. Depending on the domain under consideration, a ‘model’ might also be variously called as ‘computational model’, ‘geometric model’, ‘document’, ‘file’, and so on.

![Figure 7.11: An abstraction of a design model consisting of five items.](image)

Figure 7.11 depicts a model comprised of five items and seven constraints. Items 1, 2, 4, and 5 are simple items, while item 3 is a composite item comprised of sub-items 3.1, 3.2, and 3.3. Out of seven constraints, four are one-way constraints (one arrowhead), while the remaining three are two-way constraints (two or no arrowheads).

7.2.5 Design Model Instantiation

A model instantiation is some assignment of concrete values to all the variables contained in a model. Theoretically speaking, a model can be instantiated in innumerably (uncountably) many ways, which as a general rule holds even for very simple models. For example, a model consisting of just one circle can be instantiated innumerably many times just by varying the circle’s radius.

7.2.6 Design Model Modification

I distinguish between two qualitatively different types of model modifications: (i) discrete modification, and (ii) continuous modification.

(i) Discrete (structural) model modification. Such a modification implies a structural change to the model i.e., that either an item or a constraint has been added or removed from the model. Therefore, the focus here is on change affecting the organizing structure (items,
and constraints) of the model, instead of on what values are being assigned to the model’s variables.

(ii) Continuous (variational) model modification. A continuous (variational) modification implies that the values of one or more variables contained in the model have been changed (i.e., varied), while still using and retaining the structure of the model, at all times.

In other words, ‘continuous model modification’ implies that the model undergoes a number of model instantiations i.e., variables’ assignments, while still preserving the same organizing structure (items, and constraints governing items) at all times.

A Side Note On Continuous Modifications Triggering Discrete Modifications. In a sufficiently powerful model formalisms, one can easily see the possibility of a continuous variable modification causing discrete structural modifications\(^2\). For instance:

```java
if ( rectangle.position.x >= 30 )
    delete triangle[2]
```

In the example above, moving a rectangle will, at a point, cause the deletion of a triangle from the model. (A similar routine might also add new items to the model.) However, it should be noted that for any value of \(x < 30\), for an infinitesimally small increment \(\epsilon > 0\) it will still hold \(x + \epsilon < 30\), thus leaving us within the confines of a continuous model modification. Thus while more powerful model formalisms might lead to a hybrid model where continuous and discrete modifications are inter-mixed (or triggering each other), for sufficiently small changes in variable values, we can still safely assume two modification types only.

Of course, in floating point arithmetics, where real numbers are compressed into a finite number of bits, there are clear limits on the minimum \(\epsilon\), and for a certain value of \(x\), the sum \(x + \epsilon\) will always exceed the value of 30. For instance, the smallest\(^3\) possible positive value for \(\epsilon\), using double precision, is \(2^{-1074}\). In this case, one might assume that for a variable \(x\), the neighborhood \(\langle x - \alpha, x + \beta \rangle\) for a specific value contained in \(x\) is sufficiently wide as to still stay away from a discontinuity i.e., within the limits of continuous model modification. Otherwise, one has to deal with at least two structurally different models.

\(^2\)I first described this two-way classification of modifications of parametric models in my PhD comprehensive exam [192].

\(^3\)See the IEEE 754-1985 standard for details.
Figure 7.12: An abstract design model's history can be represented as an acyclic directed graph, whose nodes (i.e., design states) are further distinguished by developmental branches, merge events, and revision numbers.
7.2.7 Design History of a Model

Since there are two types of modification (discrete and continuous), I equivalently distinguish among two types of history: one due to discrete modifications (the ‘discrete (structural) history’), and another one due to continuous modifications (the ‘continuous (variational) history’).

(i) Discrete (structural) design model history. As a model is being modified through time, it traces a trajectory through time (Figure 7.12). I call this trajectory the model history. Each and every model modification creates, in a way, a new “version” of the model, which we might interchangeably also call a model revision. In principle, every modification of any of the model’s items’ properties or model’s constrains (no matter how small), leads to the augmentation of the model’s history.

(ii) Continuous (variational) design model history. Likewise, for any given arbitrary but fixed model, one may focus only on the sequence of its instantiations, while assuming that the model (i.e., its structure) remains immutable at all times.

I should note here that the variational history of a design model is, in a way, “subordinate” to the structural history of a design model, because model instantiations are relative to a specific, structurally constant design model. Therefore, the variational history “attaches” itself to one and only one such structurally constant model; once the structure of the model changes, one could assume a new “variational history” relative to that new, structurally changed model.

I also note that most current computational tools save both the structural changes and the variational changes into a single data structure (in many cases, a LIFO stack). However, aforementioned considerations might constitute a suitable design rationale for placing both types of changes into their own data structures, thus possibly achieving beneficial separation of concerns.

7.2.8 Set of Alternatives for a Model

Finally, I define the concept of a set of alternatives for a model, as some subset of the set of all the revisions contained in the model’s history. For instance (Figure 7.13), nodes 1, 2, 3, and 4 belong to the set B of alternatives. Naturally, these same nodes induce a sub-graph of the complete, global history of the model, by inheriting arcs (i.e., design state transitions) $1 \rightarrow 2$, $2 \rightarrow 3$, and $2 \rightarrow 4$.

In addition, as illustrated in Figure 7.13 where some subset overlap, an alternative can in principle belong to more than one set or collection. Choosing whether an alternative belongs to a single set, or to multiple sets in a concrete implementation, is basically a design decision: a designer might decide either to support disjunctive or non-disjunctive sets of alternatives.

---

4Or a variant, see the equivalency of these two terms on page 112.
Figure 7.13: Three sets A, B, C of alternatives, for a design model. For instance, set A contains alternatives 0, 1, 2, 6, 7, and 9.

7.3 Chapter Summary

In this Chapter I introduced the conceptual model (or equivalently, conceptual system design) underlying the alternatives-enabled systems being developed in this work. For this, I first described metaphors being used to develop alternatives-enabled systems (design galleries, tracing paper, and lineage trees). Finally, I described the main concepts (and their relationships): variables, items (simple, composite, and subjunctive), constraints, models, model history, and set of alternatives for a model. The benefits of a conceptual model, in general, include (1) provision of guidance for system designers, (2) reducing the size of the space of feasible interaction designs, and (3) reuse of knowledge from familiar domains.
Chapter 8

Phase C: CAMBRIA 1 Prototype

Based on the outcomes of pre-design activities of comprehending, delimiting, and conceptualizing the space of feasible HCI designs (Chapter 6), as well as on the outcomes of the activities of conceptual designing (interaction metaphors, concepts, and relations described in Chapter 7), I proceed with the design, implementation, and evaluation of higher-fidelity interactive prototypes, collectively named “CAMBRIA¹”. The screenshot of the first iteration (CAMBRIA 1) is shown in Figure 8.1.

Figure 8.1: Screenshot of CAMBRIA 1 prototype, with its three main constituent parts: Collections view (left), Alternatives view (middle), and the Parallel Editing view (right).

¹The choice of this particular name was inspired by the Cambrian Era (lasting from 541 to 485 million years ago) and the corresponding phenomenon of “Cambrian Explosion” or a massive diversification of existing organisms, from simple one-cell to complex, multi-cell ones. In CAMBRIA, equivalently, the user should be able to easily achieve a diversification and “richness” of multiple designs.
8.1 Design of CAMBRIA 1

As shown in Figure 8.1, the interface of CAMBRIA 1 consists of the following three main parts [193]:

(a) Collections view, where the user can create, select the current collection to work with, and manage all the collections present in the design project.

(b) Alternatives view, showing thumbnails of all the alternatives in the currently selected collection, and where the user can select multiple alternatives.

(c) Parallel Editing view, showing six selected alternatives being edited in the Juxtaposition Parallel Editor (top) and Superposition Parallel Editor (bottom), where the user can edit selected alternatives in parallel.

8.1.1 Collections View: Managing Collections of Alternatives

The left-most view, or Collections view (Figure 8.2), is devoted to managing the collections of alternatives contained in a design project. In this particular example, the project contains two collections of alternatives named “C-0” and “C-1”.

Within this view, the user can create a new empty collection (by selecting New Collection), select an existing collection, as well as Duplicate or Delete a collection. Operation duplicate duplicates all the alternatives contained in the originating collection, and places such duplicates in the new collection automatically named “<name>-copy”, where “<name>” is the name of the originating collection. In addition, a collection can be renamed by double-clicking on its name or thumbnail, and typing in a new name.

As for expressing collections, in CAMBRIA 1 they can be either represented by a thumbnail image generated by the superposition of all the alternatives currently contained in that collection (see Figure 8.2), or by a row of alphanumeric values arranged into columns. If represented as thumbnails, the user can increase and decrease the size of thumbnails by selecting the corresponding “make larger” and “make smaller” commands.

As for arranging collections i.e., their expressions, two views are available: (1) the “Thumbnails” arrangement where thumbnails of collections are arranged into a rectilinear grid (shown on Figure 8.2).
8.2, in this case as a $1 \times 2$ grid, however general $m \times n$ grids are possible by moving the vertical delimiter), or (2) “Data Grid” arrangement where alphanumeric data for each collection is arranged into columns and rows. Header fields in the Data Grid arrangement are selectable, allowing the user to easily rank collections by different criteria, e.g., by the count of alternatives contained, or by the date on which the collection was created or modified.

The user can navigate all the available collections by using conventional scrollbars, or by pressing corresponding system keys such as Page Up/Down, and cursor keys. The act of selecting a collection causes its thumbnail to get highlighted; in addition, the thumbnails of all the design alternatives contained within the collection are automatically displayed in the middle view, the Alternatives view, to be described in the next section.

**8.1.2 Alternatives View: Selecting and Managing Alternatives**

Figure 8.3, or Figure 8.1(b), shows the view in the middle of the tool, which gets populated automatically, as soon as the user selects a collection in the Collections view.

![Figure 8.3: “Alternatives” view in CAMBRIA 1. In the mode shown (“manual arrangement” mode), the user can select multiple alternatives (i.e., thumbnails thereof) at once, zoom in and out of them, and arrange them into proximity-based clusters.](image)

Within this view the user can select and manage (but not edit, see Section 8.1.3 describing parallel editing) separate alternatives contained within the currently selected collection. I classify the interaction features of the Alternatives view into several distinct categories, as follows.

1. **Selecting alternatives.** Within this view, the user can select one or multiple design alternatives, which is indicated by a blue border around them.
Creating and deleting alternatives. The user can create a new (blank) alternative which is then immediately placed into the current collection, and given a default name “A-x”, where x represents the running number (count) of alternatives in the system. The user can rename the alternative as needed. Furthermore, the user can duplicate all the selected alternatives, by choosing the corresponding option in the pop-up contextual menu for the selection; all duplicated alternatives are exact copies of the originating alternatives, with names suffixed by “-copy”. Finally, the user can delete a selection of alternatives.

Expressing alternatives. Within this view, an alternative can be represented either as a thumbnail image, or by a row of alphanumeric values describing the alternative, arranged into columns.

Arranging alternatives. The user can arrange for viewing (Figure 8.4) all the alternatives within the current collection in the following three ways:

(a) “Thumbnails” mode which lays out thumbnails into a rectilinear grid, thus allowing the user to easily rank alternatives, and quickly view them. In addition, the user can make thumbnails larger or smaller, and manage the tradeoff between faster access (smaller thumbnails) and better overview (larger thumbnails).

(b) “Data Grid” mode where all alternatives in the collections are arranged into columns and rows of (sortable) alphanumeric data, thus again enabling the user to sort alternatives by clicking on corresponding table headers, and

(c) “Arrangement” view where the user can arrange thumbnails of all the alternatives in the current collection manually, thus creating his or her own meaning. For instance, the user might cluster several alternatives into “promising” or “bad” designs, thus effectively being able to create sub-collections within a collection.

Navigating alternatives. To navigate all the alternatives in the current collection, the user can employ scrolling (i.e., use scrollbars, or press corresponding system keys such as Page
Up/Down or cursor keys) when using the data grid or the grid of thumbnails. In the manual arrangement mode, in addition to panning using scrollbars, the user can also zoom into and out of the canvas containing all the thumbnails.

### 8.1.3 Parallel Editing View: Modifying Selected Design Alternatives

As the user selects one or multiple alternatives in the Alternatives view, said alternatives are immediately transferred and made available in the Parallel Editing area shown in Figure 8.1(c). For convenience I also show this view in Figure 8.5 below, and its two modes of presentation or arrangement: (1) selected alternatives arranged side-by-side (juxtaposition), and/or (2) layered on top of each other (superposition).

![Figure 8.5](image)

**Figure 8.5:** Two ways of presenting alternatives in the parallel editing area of CAMBRIA 1: (a) juxtaposition, and (b) superposition.

Note that (as shown in Figure 8.1) either or both presentation modes (juxtaposition and superposition) can be active on the screen simultaneously. The Parallel Editing view is distinguished by the following interaction capabilities:

1. **Expressing alternatives.** Within this view, an alternative is represented as a full vector-based drawing or rendering, and not as a raster image as is the case with Alternatives view. The user can thus zoom in and out of the alternative without any loss of visual quality.
(2) **Arranging alternatives.** The alternatives can be presented (displayed, visualized) in two ways within the parallel editing area:

(a) Juxtaposition, in which selected alternatives are arranged in a side-by-side manner, as shown in Figure 8.5(a), and

(b) Superposition, which stacks (overlays) selected alternatives on top of each other, as shown in Figure 8.5(b).

Both editors are fully linked to each other, so a change to any alternative made in any of the editors is automatically reflected in the other editor.

(3) **Making alternatives ‘linked’ or ‘unlinked’**. Make alternatives ‘linked’ or ‘unlinked’, by checking or unchecking the checkbox GUI control on top of each alternative in the Juxtaposition Parallel Editor (Figure 8.6). A linked alternative is taking part in all the subsequent editing operations; an unlinked alternative is on the other hand ‘locked’ i.e., not receiving any editing operations, with the exception of brushing (mouse-over) in order to facilitate visual navigation among alternatives.

![Figure 8.6: Linking and de-linking of alternatives in CAMBRIA 1 occurs by toggling the checkbox at the top of the view. Here, both A-3 and A-4 are linked.](image)

Another example of linkage is illustrated in Figure 8.7 below, where the alternative A-3 has been de-linked and thus painted with a gray background, for easier visual identification.

(4) **Brushing of linked graphical items.** Refers to the action of moving the mouse pointer over an item in any of the edited alternatives will lead to the action of automatic highlighting all the corresponding items in all the other alternatives, whether linked or unlinked; I felt that brushing should be independent from linking/unlinking of alternatives, since it is basically concerned with understanding of alternatives, instead of with editing them.

(5) **Perturbing selected items in a random fashion.** Refers to the action of slightly perturbing the positions and sizes of selected items in a random fashion, by first selecting items (either in one alternative, or multiple alternatives), and then clicking on the “Throw Dice” button on the toolbar (see upper left part of Figure 8.5). The design rationale behind this capability is to potentially encounter interesting alternatives in a serendipitous manner.
As an example (Figure 8.5), the user moved the mouse pointer over a specific part of the bicycle (the “down tube”), and as the result all six down tubes have been rendered in red for visual emphasis.

(6) **Parallel selection.** Similarly to linking and brushing, and since the identifiers preserve items’ identities in different drawings, corresponding items can be selected in parallel, within multiple drawings.

![Parallel selection in CAMBRIA 1](image)

**Figure 8.7:** Parallel selection in CAMBRIA 1.

For instance (Figure 8.7), selecting a curve in one linked alternative will automatically select all linked curves in all linked alternatives.

(7) **Pass variable.** As a form of the standard copy-paste mechanisms, only generalized to an arbitrary number of potential receiving design models, if an item exists in one model, it can be “passed” to another (one, or multiple) models.
Figure 8.8: “Passing variable” mechanism in CAMBRIA 1.

Figure 8.8 illustrates this mechanism, starting from a total of four alternative (two linked, two de-linked) with the following sequence of states: (1) choosing to insert a curve (menu option), (2) two linked curves inserted, (3) parallel selection of two curves, (4) another alternative linked (the bottom-left one), (5) choosing the pass variable option (in this implementation, named as “replicate item”, (6) the curve passed into the third alternative. Of course, the same for an arbitrary number of linked design alternatives; if the curve does not exist in a linked alternative, it would be passed into all such alternatives.

(8) Pass variable value. Items can be “linked” to other items in CAMBRIA 1. Through editing, said items will inevitably diverge in terms of the values of their properties: shape, border colour, fill colour, position, size, and so on, despite their being linked (for instance, the user could activate one alternative only, and then modify it). The “pass variable value”, then, represents the mechanism which propagates the values of a subset of an item’s properties to all the linked items in all the currently linked alternatives. For instance, if the originating item’s border colour is green, then “pass variable value” for border colour will pass the green colour to all the linked items, so they too will assume border colour green.
Figure 8.9: “Passing variable value” mechanism in CAMBRIA 1.

Figure 8.9 illustrates the “pass value” mechanism, starting from a total of four alternatives (three linked, one de-linked) with the following sequence of states: (1) initial state, with three linked curves in three linked alternatives, (2) two alternatives have been de-linked (A-1 and A-2), (3) the curve now being edited (shape change) in A-0 only, (4) alternatives A-1 and A-2 being linked back, (5) choosing the pass variable value option (in this implementation, only “Shape” option available), (6) the value of geometric shape passed into all the linked items from the originating item, in all linked design alternatives.

(9) Parallel editing. Each alternative (i.e., view thereof) has a simple “tool bar” at its top featuring three buttons i.e., three available commands (“make line thinner”, “make line thicker”, and “change line colour”), that can be applied to a selection of items. The first two commands are applied immediately to the selection of items. The third command brings up a colour picker widget; upon choosing a colour, all selected items assume the chosen colour.

Figure 8.10: In CAMBRIA 1, each design alternative features a simple “editing toolbar” at its top.

In parallel editing, invoking any of the three available commands to a selection of items in
one design alternative, is applied to all the linked items in all the linked alternatives as well. Figure 8.11 illustrates the parallel editing technique, with the following sequence of states: (1) a control point selected in parallel (indicated by the red colour), (2) that same control point dragged in the view affects the geometric shapes of all three curves changed in parallel, and (3) line thickness and colour of curves being edited in parallel.

**Figure 8.11:** Parallel editing in CAMBRIA 1.

### 8.2 Evaluating CAMBRIA 1: Expert Review

Six experts were recruited for the formative evaluation of CAMBRIA 1, with the main purpose of obtaining qualitative, high-level feedback on the prototype. The experts were encouraged to “think aloud” as they explored and interacted with the prototype. Among those experts, two were SFU faculty, with significant expertise in HCI. Of the remaining four, two were experienced CAD product designers from a major USA CAD vendor, and the other two experienced computational design experts from a major (international) architectural firm based in Vancouver, BC. The latter four experts thus had significant expertise in architectural design, as well as significant expertise and experience in working with parametric CAD tools such as Bentley GenerativeComponents, Rhino Grasshopper, Dassault Systèmes SolidWorks, and Autodesk Revit. (The prototype was being developed with this target audience in mind.) The experts’ ages were in the 30–50 years range, and all were male. For brevity, in further text, when denoting an expert, I will use the descriptors P1, . . . , P6 (Table 8.1):

<table>
<thead>
<tr>
<th>Participant:</th>
<th>Main Area of Expertise:</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>HCI</td>
</tr>
<tr>
<td>P2</td>
<td>HCI</td>
</tr>
<tr>
<td>P3</td>
<td>CAD/HCI</td>
</tr>
<tr>
<td>P4</td>
<td>CAD/HCI</td>
</tr>
<tr>
<td>P5</td>
<td>CAD/HCI</td>
</tr>
<tr>
<td>P6</td>
<td>CAD/HCI</td>
</tr>
</tbody>
</table>

**Table 8.1:** Experts who participated in the study.
In the study, we were interested in finding whether the experts perceive that such a system would provide value to them, and why. Furthermore, we were interested in learning more about how can such an interface support the experts’ design tasks, as well as how the interface can be improved. The study itself was conducted in four parts:

1. *Introduction to design alternatives.* Each participant was introduced to the notion of design alternatives, and shown several examples from real-world design practice.

2. *Prototype walk-through.* Each participant was familiarized with the features of the prototype, and his voice and actions on the screen have been video recorded.

3. *The participant performs tasks.* Each participant was asked to perform tasks with the tool, primarily to design a simple house, which is representative of the target audience for the tool. In addition, each participant was also given the freedom to explore the tool freely, and perform any other (open-ended) task or to draw/design any other object that comes to their mind. Each participant’s voice and actions on the screen have been video recorded, as he was working with the tool.

4. *Post-interview.* Short interview was conducted on participant’s overall impressions about the prototype. The participant’s voice was video recorded. Questions asked were as follows:

   (a) What are the most interesting features of this prototype?
   (b) What are the least interesting features of this prototype?
   (c) Envision this in a much more developed and polished version. How would you work with it in your practice?
   (d) What barriers does it present?
   (e) How would you change it?
   (f) Do you have anything else to add?

The duration of interviews ranged from one hour to two hours. In the data analysis phase, I “coded” interview transcripts into a number recurrent themes, topics, and ideas, using NVivo 10, by conducting a total of three analysis “passes” over each of the six interviews. In the first pass, I transcribed all the video recordings; in other words, I wrote out the participants’ spoken statements as well as their actions executed with the prototype. This pass (transcription), beside expressing interviews in another (textual) form, has also allowed me to determine all the recurring “categories” or “themes” that appear in them. In the second pass, I tagged all the statements (that is, all the feedback given by participants) by (a) the type of functionality provided by each of the three main parts of CAMBRIA 1, as well as (b) recurring themes discovered in the first pass.

---

2 A commercial qualitative data analysis computer software package by QSR International.
8.2. EVALUATING CAMBRIA 1: EXPERT REVIEW

Figure 8.12: Coding of an interview transcript.

Fig. 8.12 shows a screenshot of the transcript of the interview given by a participant, being coded into a number of different categories. I note here that coding categories overlap to a degree, therefore some statements fell into more than one category. Finally, as the final (third) pass, I performed the last check by again going over all the transcripts, verifying that I did not overlook or leave out any important recurring theme, and thus achieving the completeness of analysis.

8.2.1 Feedback by Theoretical Concepts

In this section I will describe the participants’ feedback on the prototype, classified by the following theoretical concepts (categories) discovered during the transcripts’ analysis:

1. **Alternatives, and Their Structure.** Refers specifically to alternatives, and their intrinsic nature.

2. **Duplicating alternatives.** Refers to creating new alternatives from existing ones.

3. **Collections** of alternatives. Indicates several alternatives (or collections of alternatives) being grouped together and considered as a whole.

4. **Sequential editing of single alternatives.** Refers to episodes of solo-edit-solo-edit-... sequences involving just one alternative at a time, within a collection.

5. **Parallel editing of multiple alternatives.** Refers to the act of modifying an alternative, and letting all the corresponding items in other, active alternatives be edited as well, in parallel and in real time.
6. *Pass and link variables, and their values.* Designates the act of copying or moving an item from one alternative, into one or more additional alternatives, as well as the act of propagating one or many property values from the originating item (contained with an alternative) to one or more receiving alternatives.

7. *Viewing and comparing many alternatives at once.* Refers to instances of interaction where the participants viewed a number of alternatives, in order to compare them and (possibly) to combine them, or parts thereof.

8. *3D.* Feedback related to the prototype being implemented for 3D modeling as well.

9. *Collaboration.* Feedback related to potential collaboration features which might be useful in CAMBRIA or similar tools.

### 8.2.1.1 Alternatives, and Their Structure

The participants’ comments revealed their perception of the internal structure of the design alternatives related to their modularity, and strength of (de)coupling of constituent design components. According to participant P1, different artifact design types (GUIs, graphic designs, code, prose, etc.) have fundamentally different inner structure, therefore the implications for extending CAMBRIA to fields beyond 2D vector graphics editing and CAD would accordingly be different. For example, this participant suggested that the code is “tightly woven” and so modifying parts of it is always a “hellacious” amount of work [P1 59:45], as distinguished from graphical work (e.g., photo editing, 2D design, etc.) where the coupling among constituent components is presumably weaker.

For participant P2, and somewhat in accordance with P1, CAMBRIA would probably lend itself well to design solutions which require, in his own words, “small iterations”, “refinement”, and “tweaking” only [P2 50:22][P2 55:04]. Participant P3 on the other hand referred to alternatives as “assemblies” [P3 51:27] that he would like to reuse in other projects, which implies that he sees designs as groups of parts that fit together to form a self-contained unit. Accordingly, this participant would have liked to be able to develop alternatives in parallel for parts of designs as well [P3 1:42:04][P3 1:44:55][P3 1:46:16], and not only alternatives for whole (global) designs.

Participant P4 was of the opinion that creating alternatives from the scratch is difficult, and that they most probably come into existence naturally, almost as if “by accident”, as one proceeds with their work and “leaves them behind” in the trail of work [P4 1:05:37].

When describing how designers in his firm typically go about initially planning a building, participant P5 described this process in terms of its constituent parts: space, shape, walking path, patient room, corridor, washroom, window, bed [P5 25:22]. Furthermore, this participant was of
the opinion that a user should be able to “zoom into a detail”, and “work at various levels of detail” [P5 51:26], implying the ability to modify specific parts of the model as well.

Finally, participant P6 mentioned the capability to manage parts (e.g., “wall”, “roof”) of models as well [P6 1:28:18], and furthermore used the term “components” as he was describing the drawing [P6 1:29:12]. This implies a certain way of thinking for this participant, for whom designs are modular and therefore able to be decomposed into sub-assemblies and components.

### 8.2.1.2 Duplicating Alternatives

Four participants explicitly mentioned, or had invariably expected, or would have liked to have had, the capability to duplicate alternatives easily and quickly both in the Alternatives widget, and the Parallel Editing widget [P1 45:06] [P1 1:18:09] [P2 26:14] [P2 27:57] [P2 43:49] [P3 27:18] [P3 1:16:04] [P3 1:16:59] [P6 52:30].

Additionally, participant P1 suggested a separate “rapid-fire” editing mode in which every single modification of an item leads to the automatic spawning of a new alternative [P1 45:06], together with the appropriate animated “moving away” or “sliding away” [P1 1:18:09] of existing alternatives.

### 8.2.1.3 Collections of Alternatives

The participants were of the opinion that the utility of Collections view (Fig. 8.13) is fairly limited [P1 1:19:59] [P3 1:32:26] [P4 1:23:40] [P6 1:19:20], and that:

(a) Its functionality should either be incorporated into the Alternatives view [P2 33:27] [P2 34:56] [P3 1:32:26] [P6 1:19:20], or

(b) That it should be removed altogether from the prototype [P1 1:21:03] [P1 1:23:55] [P2 34:56] [P4 1:25:49].

In addition, two participants, when initially faced with the prototype, were under the false impression that the Collections view actually manages multiple different design projects, instead of collections [P1 1:20:03] [P1 1:20:27] [P6 1:01:15] [P6 1:19:20].

As another consistent finding related to collections of alternatives, the participants found that being able to group (i.e., arrange, cluster) thumbnails of alternatives manually in the Arrangement view, thus being able to create proximity-based “collections” there, represents a very useful feature of the prototype [P1 1:13:57] [P2 56:30] [P2 1:02:30] [P3 49:38] [P3 1:27:37] [P4 1:23:40].
Furthermore, the proximity-based in the Arrangement view should be demarcated somehow, perhaps by a rectangular frame (i.e., a “box”) with an editable title [P6 57:23]. One participant expressed a desire for such collections to be infinitely nestable [P3 50:50] [P3 53:28], while yet another participant was of the opinion that such collections should be shallow i.e., one-level-deep only [P6 58:10].

Moving Alternatives Among Collections. At various points during the interviews the participants tried to move (i.e., drag and drop) alternatives from one collection into another, in various ways. For example, one participant attempted to drag alternatives from the Arrangement view into a collections thumbnail in the Collections view [P2 23:31]. Another participant explicitly mentioned the need for easy and flexible moving of alternatives among collections [P3 50:38]. Participant P6 additionally stated that moving alternatives between collections would be useful, and furthermore suggested that user-created one-level-deep clusters inside the Arrangement view would make moving alternatives a) simple to implement, and b) to use [P6 1:02:21].

8.2.1.4 Sequential Single Alternative Editing

The fact that three participants embarked on episodes of solo-edit-solo-edit-... sequences involving just one alternative at a time, within a collection, is an indication that this might be a new pattern of use when using CAMBRIA and similar tools.

For this pattern of use, the participants would deactivate all alternatives in a collection (except one), and then edit that one while at the same time comparing them visually with the rest. Then they would activate and edit another alternative [P1 46:50] [P6 46:15] [P5 27:33].

Closely related to this finding, another participant would have liked to be able to maximize an alternative in the juxtaposition parallel editor (i.e., stretch the editor over the entire screen), do some work on the alternative, and then revert back to the grid-based layout [P2 29:31], as shown in Figure 8.5(a).

[P2 29:31] I mean take this whole window, and make it big. Just select the window, with a rubber band, zoom into it. I don’t want to see four copies of it, I just want to see A-4.

8.2.1.5 Editing of Many Alternatives Simultaneously

Five out of six participants stated, in one way or another, that parallel (simultaneous) editing of multiple alternatives is a a rather interesting feature of the prototype [P1 1:12:30] [P3 1:11:01] [P3 1:36:28] [P3 1:37:38] [P5 27:33] [P6 1:15:59].
8.2. EVALUATING CAMBRIA 1: EXPERT REVIEW

About most interesting features, and without any prioritization, I think what's definitely interesting is to be able to design alternatives side by side and have a relationship between those alternatives.

One participant, however, expressed a certain level of scepticism about parallel editing, stating that it will probably lead to a “destabilization” of IDs (i.e., identities) of linked items, and that it requires hand-eye coordination that he has never practiced thus potentially causing “vertigo.” Furthermore, even if he were to use parallel editing, this particular participant was convinced that he would predominantly use the superposition parallel editor for this purpose, and that it remembered him of tracing paper as commonly used in architectural design.

Another participant suggested parallel editing in the context of collaboration, using some type of a large “conference table” where designers pass changes around, and comment (i.e., annotate) each design move by audio-recording orally stated rationale for the design move.

A participant expressed the desire to be able to draw a line simultaneously in several alternatives, by drawing lines “that are essentially the same but just have different locations.” In other words, starting and ending points for such lines would be determined relative to the items in each particular alternative (for example, to control points contained in each alternative), instead of assuming identical (absolute) values.

Finally, three experts stated that users should be able to edit in parallel in the Arrangement view too, and not only be able to navigate and arrange thumbnails of alternatives. So in a sense, the Arrangement view should integrate the functionality of the parallel editors, by offering equivalent in-place capabilities.

8.2.1.6 Moving Alternatives Among Collections

At various points during the interviews the participants tried to move (i.e., drag and drop) alternatives from one collection into another, in various ways. For example, one participant attempted to drag alternatives from the Arrangement view into a collections thumbnail in the Collections widget. Another participant explicitly mentioned the need for easy and flexible moving of alternatives among collections. Another participant found that moving alternatives between collections would be useful, and further suggested that groups which are only one level deep inside the Arrangement view would make moving alternatives 1) simple to implement, and 2) simple to use.
8.2.1.7 “Pass” and “Link” Interaction Features

Participants found pass and link mechanisms “interesting” [P1 1:23:30] [P2 50:22] [P3 1:11:01] [P4 28:23] [P5 35:46] [P5 36:32]. Additionally, one participant stated that it has to be easy to “link” and “delink” items constituting alternatives [P1 49:07]. One participant, when imagining a CAMBRIA-like tool in other domains of applications, expressed his conviction that maintaining variable linkages properly in the context of text-based alternatives (such as when writing documents, or source code) might be really hard, because prose and code are “tightly woven” thus probably not amenable to being easily maintained internally (i.e., within the tool) in consistent (i.e., synchronized) state at all times; that is, the linkage would sooner or later break down in said domains of application [P1 52:23][P1 53:34]. This statement was echoed by another participant who claimed that parallel editing will probably lead to a “destabilization” of linkages between items [P4 28:23][P4 31:01].

[P4 28:23] No, it’s just a general comment, it’s very interesting how you maintain the identity of an object across multiple alternatives, especially when some of the operations can destabilize the IDs.

Another participant was worried that the ease of passing items and their values would affect the idea generation process in a negative fashion, and cause the user to come up with “small iterations” only [P2 50:22][P2 55:04].

Both the participant P5 and P6 expressed their desire to be able to freely associate (i.e., link) items of different types, so that the user can pass properties among arbitrary, user-defined “source” and “sink” items [P5 30:11] [P5 35:07] [P5 35:46] [P5 36:32] [P5 39:25] [P6 26:52] [P6 27:11] [P6 27:30] [P6 1:37:09].

[P6 26:52] Can you pass a property between different elements? So for example, you select that line, and you want to pass that property to other lines in the canvas?

8.2.1.8 See and Compare Many Alternatives at Once

Five participants stated that the ability to see and compare many alternatives at once is probably the most interesting and useful feature of the prototype [P1 47:01] [P1 48:16] [P1 1:12:30] [P2 52:31] [P3 1:15:21] [P3 1:27:37] [P5 39:25] [P6 1:10:01].

Participant P1 further confirmed that he would like to see all of them, when dealing with alternatives in graphical design [P1 47:01]. Not so, however, when dealing with text-based alternatives (prose and code writing), where the participant would prefer to see only one version of the document at a time [P1 48:16].

Interestingly, differently from P1, participant P2 would, like to compare multiple alternatives (and visualize their differences) of “academic papers” [P2 1:02:30].
8.2. EVALUATING CAMBRIA 1: EXPERT REVIEW

[P2 1:02:30] So I think, I can imagine this is a paper, and I can see the paper, and this is another version of the paper, I could pull them up side by side, or same way I can just drag them on top of each other, and if I align them nicely, it would show me what is different between the two of them, turn it red or something.

Furthermore, this participant stressed, on several occasions, the importance of somehow bringing out the differences among multiple alternatives [P2 53:37]. Finally, to confirm and emphasize the importance of visualizing differences across multiple alternatives, the participant suggested a feature akin to the “spot the differences” game for kids [P2 1:00:26].

[P2 1:00:26] And to see the differences, like those kids games, “the five things are not the same in these two pictures”. Or “spot the differences”.

Participant P3 would have liked to be able to visualize multiple designs per parameter i.e., arrange alternatives in a rectangular grid where axes X and Y represent different values for two different parameters [P3 0:59:16] [P3 1:01:47] [P3 1:03:55] [P3 1:06:05]. To see and compare multiple alternatives at once, this participant also expressed the desire to have a linear “chain” of sequentially derived alternatives, whereby the user can go back to the initial sketch and then see how changes in that sketch affected sketches that had developed downstream, thus comparing all the sketches in the chain [P3 1:11:01]. This, in turn, indicates a desire for being able to modify the design history of the model. This participant also suggested that, when viewing multiple alternatives, it would be useful to have additional information displays (“dashboards”) showing alphanumeric information about each alternative (and not only their graphic rendering) [P3 1:39:17] [P3 1:40:04]. Finally, the participant expressed the desire to be able to see alternatives of parts (and not only whole designs) as well [P3 1:42:04].

Upon duplicating a few alternatives, participant P6 stated that he sees the juxtaposition parallel editor as a type of design history as well, where each edited alternative shown in the grid represents “a snapshot of a particular evolvement of the design”. The juxtaposition editor thus allows him to simultaneously see, edit and compare multiple alternatives, thus eschewing the need to execute rapid and repeated do-undo-redo cycles as is the case with conventional, single-state systems [P6 1:14:22].

8.2.1.9 3D

Three participants expressed the desire to see the prototype being implemented for 3D modeling as well [P3 1:11:01] [P3 1:36:28] [P4 1:31:30] [P5 55:32]. Within this context, participant P3 worries that the interface could become much more complex, and lead to issues such as “object occlusion”, and “visual clutter” [P3 41:53]. Participant P4 expressed the idea of having some type
of “clash detection” (i.e., collision detection) when working with multiple 3D alternatives, as a way to deal with the inherent complexity and visual clutter [P4 1:26:35].

8.2.1.10 Collaboration

Three experts noted that they wish to see the prototype being used in the context of collaboration as well [P3 1:35:08] [P4 1:19:37] [P5 42:01]. Participant P4 referred to the push-pull model of collaboration [P4 1:19:37]. Participant P5 suggested parallel editing in the context of collaboration, using some type of a large “conference table” where designers pass changes to each other, as they stand around the table [P5 42:01].

8.2.2 Feedback: Post-Interview Questions

After the end of experimental sessions, the participants were asked five post-interview questions, as follows.

8.2.2.1 What Are The Most Interesting Features of This Prototype?

What immediately stands out from the interviews, is that Arrangement view was deemed to be interesting and useful with no exception, i.e., by all six participants [P1 1:13:57] [P2 56:30] [P3 1:11:01] [P4 1:23:40] [P5 39:25] [P6 1:15:59], [P6 1:17:07]. This was followed immediately by being able to see and compare many ideas and designs [P2 52:31] [P3 1:11:01] [P3 1:11:01] [P4 1:23:40] [P5 39:25] [P6 1:15:59][P6 1:17:07].

Other features that were found to be interesting include parallel editing (four participants) [P1 1:12:30] [P3 1:11:01] [P4 1:23:40] [P5 39:25] [P6 1:06:30][P6 1:10:01][P6 1:15:59][P6 1:17:07], as well as the “pass items” mechanism: [P2 50:22] [P5 39:25].

<table>
<thead>
<tr>
<th>Feature:</th>
<th>Votes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrangement view</td>
<td>6</td>
</tr>
<tr>
<td>See and compare many ideas</td>
<td>5</td>
</tr>
<tr>
<td>Parallel editing</td>
<td>4</td>
</tr>
<tr>
<td>“Pass items” mechanism</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 8.2: Most interesting features of CAMBRIA 1.

Looking at Table 8.2, feedback demonstrates that experts prefer to view, edit, and compare multiple design alternatives.
8.2.2.2 What Are The Least Interesting Features of This Prototype?

The Collections view/widget was found to be the least interesting feature by four participants [P1 1:20:03] [P3 1:32:26] [P4 1:25:49] [P6 1:19:20]; in addition, all views should be integrated into one view [P2 55:50]. Some feedback illustrating the disapproval with the Collections view:

[P1 1:20:03] The Collections widget. (...) Basically, I would have everything (in the application). When I have multiple projects (then yes, I would like to have a Collections widget), but within a project I would have exactly one collection.

[P1 1:20:27] (Dealing with many collections is) too much work. I rarely want, for things like this, I rarely want folders and folders and folders. But, within a given project, I’ll put all the alternatives (into just one collection).

[P6 1:19:20] (...) To me, this (Collections widget) and thumbnails (icon list view) would be least interesting. Arrangement view I think needs some modification so that I can actually take the functionality of this (Collections view) into Arrangement (view). And this also simplifies the whole view if you do that.

8.2.2.3 Envision This in a Much More Developed and Polished Version. How Would You Work With It in Your Practice?

Two participants would use a system such as CAMBRIA for collaboration [P3 1:35:08] [P5 42:01]. Further, two participants would like to use CAMBRIA for 3D modelling [P3 1:36:28] [P4 1:25:49]. Other declared uses include graphical work (e.g., photo editing) [P1 1:22:35], and refinement of GUI sketches [P2 58:12].

<table>
<thead>
<tr>
<th>Envisioned use</th>
<th>Votes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaboration</td>
<td>2</td>
</tr>
<tr>
<td>3D modelling</td>
<td>2</td>
</tr>
<tr>
<td>Graphical work</td>
<td>1</td>
</tr>
<tr>
<td>Refinement of GUI sketches</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 8.3: Envisioned uses for a CAMBRIA-like system.

Looking at Table 8.3, implementing collaborative features as well as generalizing CAMBRIA’s features to 3D modelling would likely represent interesting areas of future work, and promising directions for further expansion of CAMBRIA.

8.2.2.4 What Barriers Does It Present?

This category produced varying feedback; taken as a whole, different participants expressed different barriers to the effective use of CAMBRIA. If one were to find a common thread, however,
then the inability to work with parts represents one major barrier to effective use:

- The inability to compare parts of alternatives [P2 58:12], [P2 1:00:26].
- Not being able to break up designs into their constituent parts, and then explore alternatives only for those parts [P3 1:42:04], [P3 1:44:55], [P3 1:46:16].
- Not being able to work at different “levels” in the model [P5 51:26].

Other stated barriers include: the inability to compare whole alternatives [P2 58:12], [P2 1:00:26]; not being able to group things hierarchically i.e., organize them into nestable collections [P3 1:42:04]; the learning curve involved in using the prototype [P4 1:30:00]; the lack of interoperability with established CAD packages [P5 46:06]; and the complexity inherent to the tool [P6 1:30:53]. One participant (an CAD/HCI expert with many years of experience with using and deploying CAD systems in a large design firm) also stated that with this complexity, comes the corresponding computational cost [P6 1:32:12], and which could lead to slow processing in a polished, release-quality version of the tool [P6 1:32:56], [P6 1:34:32]. Emblematic quotes:

[P6 1:30:53] Complexity. Oh yeah. Complexity related to multiple designs, and multiple things that are changing in that one particular design. One thing is too see the proof of concept on a set of three lines, and actually coming and working through it, and say, OK, you are snapping through this point but it might lose this snap point on another one because you change things so much that it lost the connection or something like that. And if you are trying to connect with three hundred different objects and each of them is associated with another thing, and... I don’t know exactly if it’s going to work correctly or not. It might be too complex to actually achieve. So it’s complexity in both ways: the number of items that you are trying to control, and visualize at the same time. And also, the multiple alternatives that you are visualizing.

[P6 1:32:12] Yeah, so even the model would not be able to handle that complexity because it’s calculating 300 different items at the same time. For me, in GC for example, sometimes it’s actually hard with the computer power that I have to, because of all the different components, it’s thousands of things that they are calculating and visualizing at the same time, to actually move an element here because, this line actually triggers 3000 different components that are calculating the sun’s angle, and they are changing different orientations of that device. So it’s really hard on the computer power to do it on one alternative.

When asked to expand further on his opinion regarding the computational cost of handling multiple models simultaneously:
Yeah, definitely. Because I already know that it could be slow with only one alternative. While the idea is pretty good to actually be able to see two and three and four, and be able to, let’s move this one here, you are actually seeing it move in the thumbnail. Maybe there is an option to only see the thumbnails changing when you select it. Doesn’t have to update all the windows all the time. When you have four different options, and when you think about the thousands little pieces that you are updating on the computer, how to calculate and update them, in one alternative, then you will find this by four, so it’s quarter of the power. This is something that worries me and is challenging about this system. Especially the graphics, like a lot of times there are ways in different 3D programs where you would turn off the graphical update, so you do one thing, work through it, and then turn it on to see how it works graphically. Like switching to wireframe graphics and similar. All these different things and tricks that you would do to sort of make your model move faster, because now you are opening up the canvas to, like, multiply, multiply, all these different things, it might just crash on you.

It’s slow. Like, no matter how powerful the machine is, you might get into doing some computational drill that you are basically just updating thousands and thousands of lines, it takes time to update. You can always sort of maximize your use, like push it to the edge, in a way? It’s easy to do that, especially if you don’t know what you are doing. If you don’t know the implications of how the computer in the background works, with the certain elements that you are trying to update, you might just run into things that would really really slow down your process.

Looking at the barriers described above, one may infer the following: (1) the ability to work with alternatives at the level of parts and sub-parts should probably be looked into; (2) multi-state tools, while powerful, might be inherently relatively complex to use; and (3) the complexity of dealing with multiple (or even “many”) design states could lead to excessive computational costs, both in terms of time as well as space.

Possible ways to ameliorate or even remove the aforementioned issues include: (1) implementing functionality supporting modularity of states; (2) exploring ways to limit the inherent complexity of multi-state tools while still preserving their power; and (3) implementing efficient computational multi-state architectures, possibly by adopting parallel computing approaches.

### 8.2.2.5 How Would You Change It?

Four participants expressed the desire to integrate separate views into a single view.
of views, is thus by the most stated one. The second most desired change is the ability to use a CAMBRIA-like system for 3D modelling [P4 1:31:30][P5 55:32], which echoes the findings from Section 8.2.2.3 (envisioned use).

Other desired changes include: powerful comparison capabilities based on layers (i.e., superposition) [P2 1:01:03], additional drawing shapes and undo [P1 1:23:55], being able to select alternatives across collections [P3 1:51:04], adding an “object browser” for subjunctive items [P4 1:31:30], being able to change relative sizes of thumbnails in the Arrangement view in order to somehow indicate their importance [P6 1:36:23], and being able to link and de-link items belonging to different alternatives [P6 1:37:09].

<table>
<thead>
<tr>
<th>Suggested changes</th>
<th>Votes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrating three separate views into one</td>
<td>4</td>
</tr>
<tr>
<td>3D modelling</td>
<td>2</td>
</tr>
<tr>
<td>Comparison based on superposition</td>
<td>1</td>
</tr>
<tr>
<td>Additional drawing shapes and undo</td>
<td>1</td>
</tr>
<tr>
<td>Selecting alternatives across collections</td>
<td>1</td>
</tr>
<tr>
<td>“Subjunctive item browser”</td>
<td>1</td>
</tr>
<tr>
<td>Manually adjusting thumbnail sizes</td>
<td>1</td>
</tr>
<tr>
<td>Linking items of different types</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 8.4: Changes suggested for CAMBRIA 1.

Looking at Table 8.4, it is clear that the participants would overwhelmingly prefer a simpler, less cluttered interface, and that they have a relatively strong desire to try out CAMBRIA’s features in a 3D modelling environment. Other suggested changes, albeit on a one-by-one basis, also provide promising avenues of potential future research.

8.2.3 Other Feedback

This section describes participants’ feedback according to following categories: (1) confusing features, and (2) intriguing feedback.

8.2.3.1 Confusing, Non-Intuitive and Unsatisfactory Features

Again, most of the participants’ confusion and dissatisfaction revolves around the overly complex, three-part interface of CAMBRIA 1 (Collections, Alternatives, and Parallel Editing views). For instance, not being able to move alternatives around in the juxtaposed parallel editor [P1 1:13:57] implies the desire to be able to move alternatives freely around on the canvas. Being “forced” to use the Collections view [P1 1:20:27], confusion about double-clicking thumbnails in the Collections view [P2 23:16][P2 23:31][P3 24:04] , confusion due to the “change view” button in the Collections view [P2 23:31], and having a separate editing area for alternatives (instead of being able to edit them directly in the Arrangement view) also confused some participants [P2 43:50].
Other complications related to this confusing separation of views include not being able to move alternatives in the juxtaposed parallel editor, as well as not being able to control or predict their relative position in the grid \[P3 32:11\] \[P3 33:19\] \[P3 33:42\]; likewise, for having to right-click-and-drag in the juxtaposed parallel editor to pan, and then left-click-and-drag in the Arrangement view \[P3 1:55:33\]. As a further confirmation of issues surrounding the Collections view, it reminded another participant of a simple “slide sorter” in a presentation program, and which felt like being “more a management convenience than really a creative part” \[P4 1:23:40\]. Finally, not being able to remove or minimize alternatives in the juxtaposition parallel editor \[P6 55:04\], as well as the impression that the Collections widget’s purpose is managing projects, not collections \[P6 1:01:15\], and a clearly stated opinion that the Collections widget is a duplication of the Arrangement view \[P6 1:19:20\]. To conclude, then, the three-part design of the interface is clearly a major source of either confusion or dissatisfaction, and points towards unifying the three separate views into a single one.

Other minor sources of confusion include: date and time indicators in the YYYY-MM-DD HH:MM format, instead of in the more comprehensible “time elapsed” format (“three hours ago”, “one month ago”) \[P1 1:21:19\]; and the mouse wheel zooming direction which is in CAMBRIA opposite from as implemented in standard and widely-used applications \[P3 24:48\][P3 28:43][P6 37:27][P6 37:50]. Finally, as another unsatisfactory feature, the juxtaposition parallel editor found to be somewhat “intimidating” \[P2 38:34\].

### 8.2.3.2 Intriguing Remarks

Participant P1 stated in a rather emphatic fashion that he would not use collections at all, but a single canvas to manually group all the alternatives into proximity-based clusters instead \[P1 1:23:55\]. Furthermore, this participant stated that a tool such as CAMBRIA would probably be effective for graphical work, but unsuitable for prose and code editing. For instance, when working with several alternatives in the Arrangement view, the participant realized the value of seeing multiple thumbnails of visual alternatives, but claimed that this could not be extended to text-based alternatives. Even when I mentioned Juxtapose \[151\], a parallel source code editor as an example of text-based alternatives as well, the participant maintained his scepticism.

\[P1 52:23\] For interface things, like changing the colour, or certain modest changes to the interface, that’s going to work well. But code is so tightly woven to what came before and what came after and everything else.

Accordingly, the participant expressed his conviction that maintaining variable linkages properly in the context of text-based alternatives might be really hard, because prose and code are “tightly woven” thus probably not very amenable to being easily maintained in consistent (i.e., synchronized) state at all times; the linkage would sooner or later break down \[P1 52:23\][P1 53:34\].
CHAPTER 8. PHASE C: CAMBRIA 1 PROTOTYPE

Incidentally, this sentiment was echoed by participant P4 who claimed that parallel editing will probably lead to a “destabilization” of linkages between items [P4 28:23][P4 31:01].

Somewhat related to the scepticism exhibited by participants P1 and P4, participant P2 was of the opinion that CAMBRIA would be effective for refining (“tweaking”) existing ideas only, and ineffective for creating radically different ideas [P2 50:22].

[P2 50:22] It’s interesting how you can move ideas, I can move elements of different ideas around very easily. (...) I wonder if by being able to so easily transfer elements it would cause people to be more likely to have small iterations of their ideas rather than big iterations. (...) So if you wanted to get many divergent ideas, it’s almost beneficial to make it hard to duplicate.

To reiterate this point, participant P2 stated that the prototype makes idea generation simple and efficient, however he wondered whether this would prevent the user from coming up with significantly different yet still acceptable design solutions [P2 52:31].

[P2 52:31] Yeah, like it (the prototype) is encouraging me to have N+1 ideas, where they are only slightly modified from my original idea. It’s curious, because it’s trying to make idea generation phase faster, and more efficient, but are we taking away something that is actually quite important, that difficulty to make new ideas? I don’t know, it is like this philosophical conundrum, this copy-paste sort of benefit, like a detriment I guess to our ideas.

One possible workaround for this potential problem of not being able to generate “drastically different” yet still feasible designs, is to provide interaction methods and techniques for exchanging “sufficiently large” parts of the design, with other, sufficiently compatible large parts, thus achieving significant variations of the original design.

8.3 Discussion

The formative study of CAMBRIA 1 provided a plethora of useful high-level, quantitative information on the prototype. Despite the small sample size $N = 6$, the study revealed a number of issues that were reiterated by several participants, thus indicating agreement. Table 8.5 summarizes expert feedback on major functionalities of CAMBRIA 1 as described in the previous section (§ 8.2), with the second column containing qualitative descriptors (with the number of participants that deemed a feature or functionality to be useful and desirable, or not), ordered by degree of affirmation.

---

3 I suggest several ways to achieve this goal on pages 204, 272, and 314.
8.3. DISCUSSION

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Feedback:</th>
</tr>
</thead>
<tbody>
<tr>
<td>View multiple alternatives</td>
<td>very positive (6)</td>
</tr>
<tr>
<td>Compare multiple alternatives</td>
<td>very positive (5)</td>
</tr>
<tr>
<td>Arrange alternatives manually</td>
<td>very positive (6)</td>
</tr>
<tr>
<td>Move, copy, duplicate alternatives</td>
<td>positive (4)</td>
</tr>
<tr>
<td>Parallel editing</td>
<td>either positive (4) or negative (2)</td>
</tr>
<tr>
<td>Linking items</td>
<td>either positive (5) or negative (2)</td>
</tr>
<tr>
<td>Collection management</td>
<td>very negative (6)</td>
</tr>
</tbody>
</table>

Table 8.5: Expert feedback per major functionality of CAMBRIA 1.

In the following, I will detail each of the entries given in Table 8.5, and then conclude the chapter by general considerations.

Positive Feedback

By far, the most positive feedback related to CAMBRIA 1 was obtained on the ability to view and thus visually compare many design alternatives at once; in one of the participants’ terms, the ability to “look at variants that quickly and easily”. These findings tie into the existing, substantial body of related work that provides evidence about the benefits of displaying multiple objects at once\(^4\). Succinctly, human vision is the sense with the highest bandwidth of all, thus enabling the process of visual comparison [309] which can reveal and recombine structural relationships [129, 128]. Viewing and comparing multiple objects can assist the user with relating information dispersed across multiple alternatives in a set, and to “highlight and explicate design issues and decisions” [284]. Further, tools that offer juxtaposition of multiple objects thus help the user to “identify connections” and gain “cognizance of how given structures are used” [284].

The ability to view and compare was immediately followed by the ability to group (or “arrange”) alternatives into user-defined, proximity-based clusters, groups, or agglomerations. Using this feature, the participants were able to partition a set of design alternatives into disjunct subsets, or clusters, so that the alternatives within one cluster are similar to each other, according to some set of objective or subjective criteria. This way, the users were allowed to “create their own meaning” or “to make sense” of alternatives contained in the collection.

In the literature, Card et al [61] claim that grouping can be considered “an elementary form of sensemaking”, and demonstrate how users can manipulate a set of objects (i.e., a collection of web pages) as a unit within their WebBook prototype application. Proponents of the 20th century Gestalt psychological movement, likewise, deemed “grouping” to be the principal, fundamental process related to our visual perception; according to the so-called Law of Proximity, objects placed near each other are perceived as “group” [402].

---

\(^4\)I covered much of related work in Section 4.1.7 on page 76, as well as in Section 5.2 on page 92.
CHAPTER 8. PHASE C: CAMBRIA 1 PROTOTYPE

Figure 8.14: Clustering results in two new entities, or “gestalts”.

Figure 8.14 demonstrates this intuitively; once the user has grouped a number of alternatives together, they will now be perceived as “group” or, in this case, as a “pile” of alternatives. Using the Gestalt terminology, the user has produced two “proximity Gestalts”, or two new “wholes” that will now be perceived as two new, separate entities. Another perceptual principle, the Helmholtz principle of visual perception, states that “whenever some large deviation from randomness occurs, a structure is perceived”[100]. Referring back to Figure 8.14, one immediately perceives two clusters because their likelihood of occurring by chance is vanishingly small.

In addition to the three (inter-related) activities of viewing, comparing, and grouping, participants deemed standard operations such as move, copy, and duplicate alternatives important and useful in a multi-state tool such as CAMBRIA 1.

Neutral or Mixed Feedback

On the other hand, parallel editing has received mixed feedback; while several participants deemed simultaneous editing “interesting” or at least potentially useful, others expressed concern that it could introduce “vertigo”, and require cognitively demanding hand-eye coordination. Accordingly, probably as a way of coping with these issues, some participants of CAMBRIA 1 study engaged in the solo-edit-solo-edit-... episodes of editing, modifying just one design variant at a time, thus in a way refraining from working in the parallel editing mode.

These issues seem to be echoed by the findings reported in the literature; for instance, Terry [367, p. 157] states that “the ability to manipulate sets of alternatives simultaneously (i.e., apply an operation to all of them at the same time) seems less vital”. Likewise, Zaman et al [416] report that their participants preferred “sandboxing” mode, wherein participants worked with just one design at a time, and that some requested “parallel editing only be available on demand”.

Presently, it is unclear how to ameliorate these issues. Perhaps, it may well be the case that the increased editing power in multi-state systems, due to the availability of parallel editing methods and techniques, comes at the cost of sacrificing simplicity and straightforwardness of single-state editing. Future work might come up with ways to curb the apparent cognitive complexity of parallel
editing while still retaining its power, and evident applicability to interaction involving more than one design state at a time.

Related to parallel editing, some participants were concerned that the need to maintain links among items would eventually lead to the destabilization or gradual deterioration of such linkage. Another related objection was that CAMBRIA 1 would naturally lead to “incremental” design variations only i.e., produce designs which are only “insignificantly different” relative to each other, which presumably implies that tools such as CAMBRIA 1 would be able to effectively deal with “gentle” design variations only.

Regarding the issue of linkage, for all the interactive prototypes described in this thesis, and as I already presented in Section 7.2.2 on page 134, I decided to adopt the static, identity-based approach to model matching [15, 352], which is characterized by the use of unique identifiers to match different instances of the same item. As its two major advantages, this approach (1) requires no particular additional configuration by the user, and (2) is fast. Two major disadvantages, however, include [196]: (1) unsuitability for computational models that do not support automatic maintenance of unique identifiers throughout the evolution of the model, and (2) its unsuitability for models which were constructed independently by two or more users (for example, two users creating design alternatives independently from each other, but for the same design project).

To address the first issue, in CAMBRIA 1 unique identifiers are preserved across all operations that modify models. However, the second issue (inability to match independently created models) would require different approaches to maintaining correspondences across alternatives, and across items contained within; possible candidates include [15, 196, 352]: (1) signature-based matching, whereby the identities of pertinent objects (alternatives and items) are calculated dynamically, (2) similarity-based matching, whereby the identities are determined based on aggregated similarities of contained elements, and (3) custom language-specific matching which, in this case, would be specific to the underlying low-level representation used in CAMBRIA.

**Negative Feedback**

By far, the most negative feedback was associated with the Collections view, and the way management of collections was implemented there. The feedback ranged from labelling that view as “completely unnecessary”, being confusing, appearing as if related to managing “projects”, and that its functionality should be best integrated with the Alternatives view.

Even if we take into account the fact that we are dealing with a rough, rapidly-developed research prototype, the first aggravating factor is the presence of small yet frustrating implementation imperfections such as: buttons executing unexpected functions, mouse-wheel actions reverting the zoom-in and zoom-out directions; and double-clicking on thumbnails leading to the unexpected action of renaming objects, instead of opening such objects within suitable views (e.g., a collection
in the Arrangement view).

Secondly, some experimental studies [163, 31] have shown that interfaces featuring two or more views might lead to longer task completion times and lower user satisfaction rates; possible causes include the need to switch focus between separate views, and the associated, increased motor effort (e.g., eye and head movements), as well as mental effort (i.e., attention switching among multiple views, which places substantial demands on short-term memory, working memory\(^5\), and long-term memory retrieval processes). Possible remedies include: (1) integrating separate views into a single view, thus reducing the attention switching problem, or (2) adapting another approach to interacting with collections, such as e.g., “tabs” or container-widgets that can be selected by the user by clicking on their short strips projecting from the bottom of the tool, similarly to how “worksheets” have been implemented in contemporary spreadsheet applications. Such a solution would (1) remove a separate Collections widget, (2) devote the entire screen area to a single collection, and (3) still allow the user to directly manipulate multiple collections within the tool.

**Other Considerations**

Desired functionality includes the ability to not only being able to inspect and compare multiple design alternatives visually, but also to find out differences and similarities among a set of alternatives automatically i.e., computationally, similarly to the Unix utility `diff`, or various equivalent GUI tools such as `WinDiff`.

Other strongly requested features included a full support for parametric models based on one-way data-flow constraints, as well as support for the so-called “subjunctive snapping”\(^6\). Other features desired by the study participants included the ability to work with design alternatives for parts of designs as well, and not only refinements of existing ones. Experts also desired a better integration of the three views, means to annotate groups of alternatives as well as transitions among design states, collaboration features, and implementing an equivalent of CAMBRIA in the field of 3D modeling. Finally, participants expressed their desire for more comprehensive linking among items (for example, the users should be able to link arbitrary items, not only cognate ones), as well as being able to “browse” subjunctive items: if a number of items are linked, then all of them should be viewed and manipulated as a single unit.

Table 8.6 summarizes desired features and functionality for an improved and polished version of CAMBRIA, as well as some envisioned or probable uses (3D modelling, collaboration).

---

\(^5\)Working memory is thought to consist of several components working together [29]. For example, the visual working memory (VWM), which temporarily stores information obtained through the sense of vision, and retains this information across saccades, is estimated to have the capacity of around four chunks or items [77].

\(^6\)As requested by my academic advisors, to be implemented in the next high-level prototype (CAMBRIA 2).
8.4 Chapter Summary

This chapter described the design, implementation, and evaluation of CAMBRIA 1, the first higher-fidelity prototype developed in this thesis. I evaluated the prototype by conducting “think aloud” interview sessions with six HCI and CAD experts, as they were using and testing out the tool. I analyzed the interview transcripts in order to discover recurrent themes, statements and opinions, which in turn I then coded into corresponding categories. The findings show that the participants overwhelmingly preferred being able to view and compare many alternatives at once, and arrange them manually. In addition, they suggested several desirable features such as an integrated view, and the ability to snap control points in parallel. Likewise, negative feedback indicated which features of the prototype were problematic, and therefore less likely to be effective for alternatives-enabled work. Taken as a whole, the experience acquired and the findings from the evaluation study will now help me with the next iteration of CAMBRIA, as well as inform the synthesis of design guidelines for specifying alternatives-enabled pCAD tools.

<table>
<thead>
<tr>
<th>Functionality:</th>
<th>Desirability:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differencing of alternatives</td>
<td>strongly desired</td>
</tr>
<tr>
<td>Simple, uncluttered interface</td>
<td>strongly desired</td>
</tr>
<tr>
<td>Subjunctive snapping</td>
<td>strongly desired</td>
</tr>
<tr>
<td>Collaboration features</td>
<td>strongly desired</td>
</tr>
<tr>
<td>3D modelling</td>
<td>strongly desired</td>
</tr>
<tr>
<td>Working with parts of alternatives</td>
<td>strongly desired</td>
</tr>
<tr>
<td>Significant variations</td>
<td>desired</td>
</tr>
<tr>
<td>Integrated view</td>
<td>desired</td>
</tr>
<tr>
<td>Annotation</td>
<td>desired</td>
</tr>
<tr>
<td>Object browser</td>
<td>desired</td>
</tr>
<tr>
<td>Generalized linking</td>
<td>desired</td>
</tr>
</tbody>
</table>

Table 8.6: Feedback on CAMBRIA 1: features desired by experts.
Chapter 9

Phase D: CAMBRIA 2 Prototype

Phase D of research is characterized by the development and evaluation of the second higher-fidelity prototype, named CAMBRIA 2 [193] and shown in Figure 9.1, as well as by the further refinement of design requirements, and of the design specification.

**Figure 9.1:** CAMBRIA 2 screenshot, showing multiple bicycle designs derived from each other.

To evaluate CAMBRIA 2, in the following section I conduct an analysis of its features using the *Cognitive Dimensions* framework by Green, Blackwell *et al* [139, 138, 41, 140]. This can be regarded as a suitable technique to evaluate almost any kind of cognitive artifact [42].
9.1 Main Changes from CAMBRIA 1

In addition to a number of minor changes to be explained in full in the following sections, CAMBRIA 2, respective to CAMBRIA 1, incorporates design changes based on expert feedback from the formative study, including some design decisions by my research group. These changes are as follows:

1. **Removal of the Collections view.** According to experts who participated in CAMBRIA 1 study, this view was either deemed as not very useful [P1 1:19:59] [P3 1:32:26] [P4 1:23:40] [P6 1:19:20], that is should either be incorporated into the Alternatives view [P2 33:27] [P2 34:56] [P3 1:32:26] [P6 1:19:20], or even removed outright [P1 1:21:03] [P1 1:23:55] [P2 34:56] [P4 1:25:49].

2. **One-level deep collections.** There was some disagreement whether collections should be infinitely nestable ([P3 50:50] [P3 53:28]) or one-level deep only ([P6 58:10]) thus making moving alternatives (a) simple to implement, and (b) to use [P6 1:02:21]. For CAMBRIA 2, I decided to adopt the second approach due to the same reasons (simplicity and ease of use).

3. **Alternatives as fully editable Multiple Document Interface (MDI) subwindows.** Three experts expressed their desire to be able to edit design alternatives in the Arrangement view as well [P1 1:15:25] [P1 1:17:35] [P1 1:18:09] [P1 1:23:55] [P2 34:21] [P2 34:56] [P2 44:36] [P3 1:30:30] [P3 1:52:45]. Accordingly, CAMBRIA 2 enables the user to both move manually views around, as well as edit alternatives directly within their respective views or windows.

4. **Full parametric data model based on one-way data-flow constraints.** The data model in CAMBRIA 2 is now fully parametric, with a number of specific one-way data-flow constraints available to the user, e.g., project-coordinate-x or parameterize-point-on-line. In addition, the underlying data structures can easily be expanded to accommodate additional constraint types.

5. **Subjunctive snapping.** This feature has been implemented upon the suggestion by one of my academic supervisors. Control points can now snap onto curves as well, and not only onto other control points as in CAMBRIA 1. A control point $P$, when snapping onto a parametric curve for some $t \in [0, 1]$, can now optionally snap onto all of its instances onto all the corresponding curves, for all linked alternatives, and for the same $t$, thus achieving "subjunctive" or "multi-state" parallel snapping.

6. **File management.** Differently from CAMBRIA 1 which used in-memory models only for the sake of development speed, CAMBRIA 2 now provides full file management allowing the user to save projects, collections within projects, and design alternatives themselves.
(7) **Undo capability.** Last but not least, CAMBRIA 2 provides full undo capability, as suggested by some by CAMBRIA 1 study participants [P1 57:19] [P2 27:57], with one undo stack per design alternative. Undo provides both the means to conduct near-term, “what-if” design experiments, as well as the means to recover from errors.

### 9.2 Design of CAMBRIA 2

CAMBRIA 2 represents a further evolution of CAMBRIA 1, and should be best understood as another test bed for evaluating the effectiveness of ideas, interaction modes, methods, and techniques. In addition to the lessons learned from designing, building, and evaluating CAMBRIA 1, other sources of ideas for CAMBRIA 2 included further review of related work, as well as direct input from my academic advisors (foremostly, the idea for “subjunctive snapping” interaction technique in parametric CAD).

In CAMBRIA 2, design alternatives belong to **collections**, and both the alternatives as well as collections are, in turn, organized into **design projects**. I present this schematically in Figure 9.2.

![Figure 9.2: Schematic depiction of the hierarchy of constructs in CAMBRIA 2.](image)

In the following, I will describe the features of CAMBRIA 2 classified into the following task-related categories: (1) managing projects, collections, and alternatives; (2) arranging alternatives for viewing; (3) navigating alternatives; and (4) editing alternatives.

#### 9.2.1 Managing Alternatives and Related Constructs

The functionality for managing (loading, opening, closing, renaming) the three main constructs (i.e., alternatives, collections, and projects) are found in several places in the tool. To begin with, Figure 9.3 shows the **File** menu, listing commands to manage the three main constructs, such as open, close, rename, duplicate, and remove.
Starting from the highest-level construct, a design project serves as a vehicle for organizing the user’s work, and is saved into its own physical file, separate from files containing design alternatives. A project file contains information such as the project’s name, its unique identifier, references to design alternatives that have been worked on by the user during the life of the project, as well as the last GUI states’ information (such as a window positions, sizes, and status). Figure 9.3 shows the options available to manage projects:

1. **New project.** Creates a new, empty project, with no collections and no alternatives.
2. **Open project.** Opens a dialog which allows the user to open an existing project.
3. **Close project.** Closes the currently open project.
4. **Recent projects.** A sub-menu offering the user the list of most recently opened projects, for faster access.

A collection serves as a means to organize and gather a set of design alternatives. A design project can have multiple collections. In terms of GUI, in CAMBRIA 2, each collection is implemented as a “tab”, i.e., a devoted container-widget that can be selected by the user by clicking on its short strip projecting from the bottom of the application. Figure 9.4 illustrates this functionality, with a total of five different collections currently being utilized in the project (one tab per collection).
(1) *Insert new collection.* Creates a new (empty) collection, with no alternatives.

(2) *Duplicate current collection.* As per name, duplicates the current collection \( C \) by: 1) creating a new collection \( C\text{-copy} \), and then 2) making a copies of all the design alternatives contained in \( C \) and placing these copies into \( C\text{-copy} \).

(3) *Rename collection.* As per name, changes the user-defined name of the collection.

(4) *Remove current collection.* Removes the current collection \( C \) from the project. Note that the alternatives contained in \( C \) are not removed from CAMBRIA’s database; in fact, they are preserved in the CAMBRIA’s database, and can be subsequently reopened and/or added to other collections. The entire database can therefore be regarded as one large, “global” collection.

Design alternatives are the top-level final outcomes (artifacts) of designing i.e., of interacting with the tool. CAMBRIA 2 is a multi-state tool and enables the user to view and edit multiple design alternatives in parallel. In practical terms, each design alternative is being apportioned its own view within the tool’s main window, and each tab (i.e., collection) widget then acts as the container for multiple such sub-windows. The user can drag a sub-window within the workspace, make it smaller or bigger, and change its window status (e.g., from maximized to normal to minimal and back). All the activities of viewing and editing of a specific alternative, take place within this sub-window, or view.

All design alternatives ever created and modified by the user are stored in a specific folder in the file system; furthermore, each design alternative is serialized into its own physical data file. As shown in the *File* menu (Fig. 9.3), the user can:

(1) *Insert new (blank) alternative.* Creates a new (blank) alternative and places it into the current collection.

(2) *Insert an existing alternative.* Opens a dialog which allows the user to point to an existing design alternative to be opened and placed into the current collection.

(3) *Duplicate the current alternative.* As per name, duplicates the current alternative \( A \) by: 1) creating a new (blank) alternative \( A\text{-copy} \), 2) copying all of the content of \( A \) into \( A\text{-copy} \), and 3) placing \( A\text{-copy} \) into the current collection. Copies can be considered “clones” because the operation of duplicating preserves items’ identities.

(4) *Move the current alternative.* Moves the current alternative into another collection.

(5) *Close alternative.* Closes the current alternative within the current collection. Again, this command does not delete the alternative from CAMBRIA’s database, but merely removes it from the collection and the project.
9.2.2 Arranging Alternatives for Viewing

This class of features subsumes the capabilities for displaying (i.e., presenting, laying out) multiple (or just one, in the special case \( m = 1 \)) design alternatives. An “arrangement” can be defined as a specific set of GUI state properties for all the alternatives (i.e., views) in a collection: view position, view size, and view z-order (i.e., the order of how all the views stack on top of each other). At any time, the user can manually change the position and size of the view of each alternative in a collection, change the window status of the view (minimize, maximize, restore), and group them into proximity-based clusters. Figure 9.5 shows the Arrange menu containing options for arrangements.

![Figure 9.5: Arrange menu with options for presenting a set of alternatives in CAMBRIA 2.](image)

Specifically, the user is able to:

1. Change the current arrangement by running Grid, Slideshow, Design history, or Random arrangement commands, which automatically reposition and resize the views from their current configuration, onto selected configuration.

2. Save the current arrangement, either through Save arrangement or through Save arrangement as... command.

3. Load a previously saved arrangement, which automatically repositions and resizes the views from their current configuration, and

4. Rapidly select an arrangement from the list of most recently saved arrangements.

Available arrangement modes, shown graphically in Figure 9.6, include:

1. **Grid (pack all)** arrangement. All the currently visible alternatives in the collection (thus, minimized alternatives are not included) are being packed into all the available application area, following a rectilinear grid (Figure 9.6 a).

2. **Slideshow arrangement.** All the alternatives in the collection are maximized and can be browsed sequentially, in the order of their creation (Figure 9.6 b).
(3) **Design history arrangement.** This arrangement type visualizes the **lineage** (i.e., the descen-
dance or derivation relation, created through duplicating alternatives) of design alternatives in a collection. In other words, alternatives are arranged into a directed graph, with links (edges) denoting lineage (Figure 9.6 c).

(4) **Random arrangement.** This arrangement type repositions and resizes the views in a random fashion, thus possibly achieving a configuration that lends itself well to a serendipitous discovery of interesting or relevant parts, shapes, and structures (Figure 9.6 d).

In CAMBRIA 2, I implemented animated transitions among different arrangements by interpolating both the **positions**, as well as **sizes** of alternatives’ windows. This was to avoid abrupt dislocations whereby the user might lose track of the position of an alternative in the new arrangement. Using animation, one arrangement of design alternatives smoothly and continuously transitions into the next arrangement. The transition times of 0.8 up to 1.0 seconds seem to work well, providing the right balance of quickness (i.e., sufficiently fast dislocation of views) and the
user’s ability to track the transition effectively.

9.2.3 Navigating Alternatives

Navigation refers to the combination of 1) wayfinding (i.e., the activity of developing and utilizing a cognitive map of the environment being navigated) and 2) motion (i.e., the activity of changing the viewpoint within an environment) [400, 282, 93]. In the case of CAMBRIA 2, this “environment” includes design alternatives, their inner composition and makeup, as well as how those alternatives are grouped into collections within a specific design project. The means to navigate collections and alternatives include various pan and zoom interaction techniques.

Panning

Panning allows the user to view parts of the computational model that currently cannot fit into the available display area, while maintaining the current display scale (i.e., zoom factor) of the view. Panning in CAMBRIA 2 occurs when the user drags the “scrollbar thumb” on either of the two scrollbars (horizontal or vertical), or alternatively presses the mouse button within an area of the view that does not contain the model, and then continues “dragging” the mouse pointer.

Panning at the collection level. Panning at the collection level is implemented in a standard fashion, for which the user utilizes standard functionality (scrollbars, and mouse wheelbutton action). This is a standard single-state panning mechanism (if we regard a collection as a “single state”), which allows the user to reach alternatives which are currently obscured from the main view.

Panning at the alternative level. For panning at the alternative level, the user has to press the mouse button in an empty area of the alternative (i.e., in an area currently not showing any part of the computational model), and then “drag” the mouse pointer, thus translating the alternative’s rendering in its plane. Two different ways to pan at the alternative level have been implemented:

• Single-state panning. This modality is a standard one, as found in current software applications such as e.g., Google Maps. By simultaneously pressing the left mouse button and the Shift key, and dragging the mouse pointer, just one computational model (the currently active one) translates accordingly.

• Multi-state (parallel) panning. By pressing the Shift + Alt keys while simultaneously dragging the mouse pointer, the user is effectively panning all active alternatives by the same vector (as measured in each alternative’s local coordinate system).
Parallel panning of multiple alternatives (before)  

Parallel panning of multiple alternatives (after)  

Figure 9.7: Panning in parallel across multiple alternatives in CAMBRIA 2, using mouse press and drag, with the key Alt depressed.

Figure 9.7 illustrates parallel panning, whereby the user has dragged all four canvases to the left, in parallel, for the same delta vector distance.

Zooming

Zooming denotes a class of navigation techniques based on a temporal segregation of the overview view and the detailed (zoomed-in) view [72]. In CAMBRIA 2, one can distinguish among 1) single-state (conventional) zooming, and 2) multi-state zooming.

Single-state zooming. Single-state zooming in CAMBRIA 2 is similar to any other standard and/or mainstream application that in any way requires zooming in and out of the document. Standard zooming commands include:

1) **Type-in zoom factor.** For this command, the user types in a specific value for the zoom factor (scale factor) into the corresponding current zoom factor input field, for instance “75%”, or selects a menu option with a concrete zoom factor value (e.g., “125%”).

2) **Zoom in and out incrementally.** For these commands the user presses the corresponding buttons on the toolbar, selects corresponding menu options, or rotates the mouse wheelbutton in either direction.

3) **Fit selected items.** For this command, the user selects one or more items in the alternative, and then presses the corresponding button on the toolbar, or selects the corresponding menu option. As a result, the view zooms in into the exact bounding box of the selection.

4) **Fit whole alternative.** This is similar to Fit selected items, only applied to all the items in the alternative. As a result, the user is able to view the alternative in its entirety.
**Multi-state zooming.** Multi-state zooming (Figure 9.8) in CAMBRIA 2 occurs when the user simultaneously depresses (and keeps depressed) the Alt key while performing any of the single-state zooming operations above.

1. **Type-in zoom factor.** In this case, the same zoom factor (for example, “125%”) typed in by the user is applied to all the alternatives. This allows the user to view all the alternatives at the same scale.

2. **Zoom in and out incrementally.** In this case, all alternatives are zoomed in or out in parallel. Still, their relative zoom factors are preserved. Thus while the zoom factor increases or decreases for all the alternatives by the same amount, their absolute zoom factors might still remain different. This way, the user can incrementally zoom in or out of multiple areas of interest in parallel.

3. **Fit selected items.** In this case, both the items in the current alternative as well as matching items in other alternatives are fitted into the available area of the view. This way, the user is able to view multiple instances of an item at the largest scale factor that fits into the view, thus facilitating visual inspection and examination of all the instances.

4. **Fit whole alternative.** Similar to **Fit selected items**, only applied to **all** items contained in an alternative. In multi-state zooming, therefore, all the respective alternatives are fitted into their respective views. This allows the user to see complete scenes at once, thus obtaining their overview.

Note that some of the commands above actually require a combination of panning and zooming. For example, parallel **Fit selected** and **Fit entire alternative** require both zooming and panning to dial in into their respective selections, because all the matching items do not necessarily reside at the same geometric location in their respective alternatives, nor do they necessarily have the same size across alternatives.
9.2.4 Editing Alternatives

Editing can be understood as the exploration of a design space. CAMBRIA 2 edit capabilities contain the largest number of techniques and methods, including a number of novel ones within the context of parametric design. Each editing operation executed by the user can thus be regarded as a move in the associated design space. This section describes such editing operations, which in CAMBRIA 2 can optionally operate in parallel, i.e., affect multiple design alternatives simultaneously.

Differently from CAMBRIA 1 where parallel editing takes place within a separate “parallel editing” area, CAMBRIA 2 allows for parallel editing directly within the arrangement view. Thus, parallel editing can take place within the views that have been arranged manually by the user, tiled, cascaded, or arranged in any other manner.

To start with conventional, single-state operations, CAMBRIA 2 features a set of standard editing operations, such as choosing among various drawing modes (Select, Point, Line, Box, Circle, Curve), standard commands for undoing or redoing work (Undo, Redo), commands for transferring items from a source to a target destination (Cut, Copy, Paste), as well as commands for modifying the colour and thickness of the pen/outline (Increase pen width, Decrease pen width, Pen colour). Figure 9.9 shows the main Edit menu containing various editing commands; equivalent actions can be found on the toolbar of CAMBRIA 2, for easier access.

![Edit menu in CAMBRIA 2.](image)

Figure 9.9: Edit menu in CAMBRIA 2.
Editing “Modes”, and Their Functionalities

Editing “modes” in CAMBRIA 2 include Select, Point, Line, Box, Circle, and Curve, which enable the user to, respectively, select, drag, and examine (i.e., right-click) selected items on the canvas, as well as allow the user to draw points, lines, boxes, circles, and general (free-hand, free-form) curves:

(1) Select Mode. Again, enables the user to, foremostly, select multiple items either in one alternative, or all linked items in all active design alternatives. In this mode, the user can also drag (translate) selected items in any direction within their plane, as well as right-click items in order to invoke the contextual (popup) menu specific for selected items.

![Parallel select](image1.png)

**Figure 9.10:** Parallel (item-by-item) selection in CAMBRIA 2.

Parallel select (Figure 9.10) takes place if the user is concurrently depressing the Alt key and clicking on items.

![Parallel rubberband selection](image2.png)

**Figure 9.11:** Rubberband selection of items in CAMBRIA 2 (BEFORE).
Another selection mechanism in CAMBRIA 2 is the standard rubberband selection (Figures 9.11 and 9.12). As the user drags the mouse pointer, the tool draws a dynamically-drawn “rubberband” rectangular outline which, when the user releases the mouse button, selects all the items within the outline. Inspired by the SketchUp modeller\(^1\), I implemented two rubberband modes: (1) dragging the outline box from left to right selects only the items whose bounding boxes (BBs) fall entirely within the outline box, and (2) dragging the outline box from right to left selects all the items that fall into the outline box at least partially.

In Select mode, the user can also invoke contextual (“popup”) menus for right-clicked items (Figure 9.13). In general, each menu is custom-tailored for each selected item, showing commands available, item type on top (point? line? . . . ) as well as all the currently available and active constraints acting on the item. Thus, in addition to providing command options, such menu in a way also (partially) assumes the role of a details-on-demand view.

\(^1\)A modeller originally developed by @Last Software, later acquired by Google Inc. and then Trimble Navigation Inc.
(2) **Point Mode.** Points are the fundamental, basic geometric items governing other items in the parametric model. At an abstract level (as presented in Chapter 7 on the conceptual model), points can be considered composite items composed of two variables (coordinates $x$ and $y$). At the geometric level, however, a point can be considered to be a basic, atomic element which cannot be split or divided further. As the user edits a drawing, a point can be converted to and from the following subtypes of points:

(a) **Free points.** These can be moved freely around the canvas, thus influencing the underlying parametric model in real time. In addition, either $x$ or $y$ (or both) coordinates can be "locked" i.e., held constant at all times. Figure 9.14 (left) shows several free points, out of which four are selected, and one is being brushed i.e., mouse-hovered as indicated by the red colour.

(b) **Parametric points on line.** These points are "attached" to any curve-like object (such as e.g., line segments, curves, Béziers, circles, rectangles), and their position is defined by an arbitrary but fixed real number $t \in [0, 1]$. For example, if the user drags a free point and "snaps" it somewhere in the middle of any line object, that free point is automatically converted into the "parametric point on line" type with $t \approx 0.5$. Figure 9.14 (right) shows one parametric point on a Bézier curve, with $t \approx 0.4$.

(3) **Line Mode.** This mode allows the user to draw straight line segments, which in turn are defined by two points (either free points or parametric points on line). In order to do that, the user should first depress the left mouse button in order to place a new point, or select an existing point. Then, he should drag the mouse pointer, and release it once it is in empty area (thus creating a new free end point), or release the button over an existing point (thus re-using an existing point as the end point for the line segment).
(4) **Rectangle Mode.** Similar to Line mode, only that the act of dragging produces a rectangle defined by two corner points (top left, bottom right). Effectively, this command inserts a parametric sub-graph consisting of four points and four line segments, whereby points are connected through “project $x$” and “project $y$” constraints (see the section on constraint editing below).

(5) **Circle Mode.** Circle mode, likewise, through the action of click+drag+release inserts a parametric sub-graph consisting of two points (center point, and a point on radius) as well as the resulting curve (i.e., the outline of the circle). By dragging the radius point, the user can adjust the circle’s radius. Both points (center, radius) are snappable to other objects in the model.

(6) **Curve Mode.** In this mode, the user can draw Bézier curves defined by two end points, and two control points.

![Figure 9.15: Various graphical items than can be created in CAMBRIA 2.](image)

For illustration, Figure 9.15 shows a dummy model with several items created in *Point, Line, Box, Circle*, and *Curve* modes.

**Constraint Editing**

Constraints are objects that govern the items in the model (see a more general definition in Section 7.2.3 on page 136). Constraints are added, and deleted by right-clicking on a point, and then choosing the corresponding option in the contextual popup menu for the item.
9.2. DESIGN OF CAMBRIA 2

Figure 9.16: Various constraints available for a free point in CAMBRIA 2.

Figure 9.16 illustrates a typical contextual menu that appears once the user selects a free point, offering the following constraint types.

(1) Coordinates held constant. These are perhaps the simplest constraints, whereby either the coordinate $x$ or the coordinate $y$ (or both) are “locked” i.e., held at their present values, at all times. This is equivalent to one-way constraints

$$ x \leftarrow a \quad \text{or} \quad y \leftarrow b $$

... whereby $a$ ($b$) is a real value indicating the point’s abscissa (respectively, ordinate) within the model. A “locked” coordinate can later on be “unlocked”, as indicated by the third option in the menu.

(2) Coordinates projected from another point. Either the coordinate $x$ or the coordinate $y$ (or both) can be “projected” i.e., their value can be equal to the respective coordinate’s value of some other point. If a point receives a projected $x$ ($y$) coordinate, then its abscissa (respectively, ordinate) moves in unison with the source’s (or projector’s) abscissa (ordinate). This can be concisely written as:

$$ p.x \leftarrow q.x \quad \text{or} \quad p.y \leftarrow q.y $$

If one or both coordinates are projected for a point, CAMBRIA 2 adjusts the contextual menu accordingly, and offers options to “unproject” all projected coordinates, thus enabling the user to remove the constraint from the point.

(3) Delta dependency. With this type of one-way data-flow constraint, a point $p$ moves “in unison” with another (parent) point $q$. Concisely, this may be written as in any of the following three
wants:

\[ p \leftarrow q + \delta \]

\[ (p.x, p.y) \leftarrow (q.x + \delta.x, q.y + \delta.y) \]

\[ p.x \leftarrow q.x + \delta.x \quad \text{and} \quad p.y \leftarrow q.y + \delta.y \]

In other words, as the parent point \( q \) moves, \( p \) moves as well and stays at all times at the same vector distance \( \delta \) from \( q \). Again, in the contextual menu for a point, CAMBRIA 2 offers an option for removing the delta dependency if a point already has one.

4. **Mirror dependency.** With this constraint a point \( p \) moves as a “mirror” image of a point \( p' \), through a “center image” point \( q \). Concisely:

\[ p.x \leftarrow 2 \cdot q.x - p'.x \quad \text{and} \quad p.y \leftarrow 2 \cdot q.y - p'.y \]

As the end effect of the formulas above, \( p \) is mirrored at all times from \( p' \), through \( q \).

5. **Parameterization on line.** This constraint forces a point to stay at a certain “parametric distance” \( t \in [0, 1] \) along a line or curve, irrespective of how much the underlying line or curve changes its shape. In order to change the current value of \( t \), for example from 0.12384 to 0.62, the user has to select the constrained point, choose the “Change t...” option in the contextual menu, type in the new value 0.62 into an input field, and press OK. The point now moves to its new position.

In order to remove this constraint from a point, the user has to select the point, and then choose the option “Convert to free point”, which will transform the point into a free point.

6. **Other constraint types.** CAMBRIA 2 is easily extensible, in the sense that writing new constraint types is a matter of writing a matching “update” method in C++, and inserting the corresponding option into the contextual menu.

**Undo Mechanism**

Undo/redo is a standard functionality which allows the user to recover from editing errors, but also to experiment with an alternative in the near-term, “what-if” fashion. In CAMBRIA 2, each alternative is equipped with its own undo stack, or “command history”, which is being populated whenever a single-state command (for one alternative) or a multi-state command (for multiple alternatives) has been executed by the user. Figure 9.1 shows multiple design alternatives with their undo views shown as well.
Pass Items and Item Properties

In addition to conventional, single-state commands *Cut, Copy, Paste* for transferring items from a source to a target destination, CAMBRIA 2 features two novel, parallelized transfer mechanisms: 1) “pass items”, and 2) “pass properties”.

**Pass Items (Variables, or Objects)** command might be considered a generalization of the standard *Paste* command, in the sense that it transfers all the selected items into *all* the currently visible alternatives in a collection, not just the current one. Figure 9.18 demonstrates the pass
items mechanism. To the left, the user has selected a number of items in the top left alternative (all items comprising the rear part of the bicycle). To the right, the effect of the pass items command: all selected items have now been “passed” into all other active design alternatives.

![Pass pen colour (before)](image1) ![Pass pen colour (after)](image2)

**Figure 9.19:** Pass property (in this case, pen colour) mechanism in CAMBRIA 2.

Another transfer command, the Pass Properties (Values) command, allows the user to transfer or propagate a property value (such as e.g., position, size, length, line thickness, colour, and so on) to all the linked items in all the other currently linked alternatives, for all selected items in the originating alternative. Figure 9.19 demonstrates pass property in CAMBRIA 2; to the left, I changed the colour of one part (seat tube) in the top left alternative. Upon executing pass pen colours variant of the mechanism, the pen colour is passed to all the matching items\(^2\) in all the active alternatives.

As a side note, the operation of parallel moving (dragging) of a set of graphical items might also be considered a form of Pass Properties, in the sense that it continuously transfers one specific property value (point position) as the user drags the mouse pointer on canvas in real time.

**Subjunctive Point Snapping**

Point snapping, a common feature in vector graphics editing and CAD packages, refers to the interaction mechanism where a dragged point automatically “falls into” a pre-existing position, for instance a position on a grid, a position belonging to another point, or a position on a curve defined by an arbitrary but fixed parameter \(t \in [0, 1]\). The main purpose of snapping is to decrease the complexity of designing, as well as to provide means to impose a certain order or regularity to the model.

In addition to this conventional, single-state mode of snapping, CAMBRIA 2 also features the so-called subjunctive point snapping, whereby a control point snaps to corresponding, matching

\(^2\)The pass items command preserves items’ identifiers, thus CAMBRIA 2 is able to later on find linked items, in this case matching seat tubes.
pre-existing positions in *all* currently active alternatives in a collection, and not just in the current alternative i.e., in the alternative wherein the user is currently dragging a point. Thus, as the user drags a point $P$ and snaps it onto a line or curve $L$ in the currently active alternative, the tool will snap all instances of $P$ onto all instances of $L$ within all active alternatives. Therefore, subjunctive point snapping, differently from conventional, single-state snapping, takes into account the *identities* of involved items (points, and lines/curves), instead of their geometric positions.

![Creation of a simple alternative [9]](image1)

![Duplicating [9] three times into [14], [15] and [16], and then modifying them slightly.](image2)

**Figure 9.20:** Point snapping in parallel across multiple alternatives in CAMBRIA 2.

To illustrate subjunctive snapping, in Figure 9.20 we begin with a simple design alternative [9], which is then duplicated three times into alternatives [14], [15] and [16]. To keep this example simple, each alternative contains only two items: a line segment, and a curve. Since all three new alternatives have the same lineage, all four line segments are linked (i.e., have the same unique id); the same holds for curves, as well as all of their control points. As seen in the right screenshot, after some moving an adjusting, the position of one end point of the line segment is different from the position of the remaining three linked control points. The shape and position of curves have been likewise changed in all four alternatives.

Figure 9.21 (left) now illustrates the act of user dragging a control point in alternative [9] in order to snap it onto the curve, while keeping the Alt key depressed at all times, thus dragging the linked control points in the remaining three alternatives as well. Once in position, the user now releases the mouse button. Figure 9.21 (right) shows how these four linked control points snapped into the respective parametric points on curve *with the same* $t = 0.14904$, although in geometric terms those four positions are completely different from each other. Therefore, subjunctive matching functions on the basis of *identities or semantics* of the items involved, instead of on their geometric positions.
9.3 Evaluation of CAMBRIA 2

As per Chapter 2 (Section 2.7 on page 30) I view computational tools, including the class of tool being investigated in this thesis, as a type of “cognitive artifact” that helps humans improve their cognitive performance. Appropriately, I will structure the evaluation of CAMBRIA 2 around the Cognitive Dimensions (CD) framework by Blackwell, Green, and others [139, 138, 41, 140], and which can be regarded as a suitable technique to evaluate almost any kind of cognitive artifact. The CD framework can be viewed as an informal, broad-brush and qualitative yet powerful and effective discussion and evaluation tool that allows HCI/interaction designers to talk about and manage various design trade-offs that inevitably occur during the HCI/IxD decision making process. The CD framework can thus help the HCI/interaction designer to pinpoint the weaknesses (as well as strengths) of an HCI/interaction design, and then proceed correspondingly with redesign activities, or with synthesizing recommendations about the future iterations of the design. CD framework is based on a number of so-called “cognitive dimensions” along which the designer can discuss and evaluate a certain aspect of the HCI/interaction design, including:

1. Visibility  
2. Juxtaposability  
3. Hidden dependencies  
4. Viscosity  
5. Terseness  
6. Error-proneness  
7. Hard mental operations  
8. Abstraction gradient  
9. Premature commitment  
10. Progressive evaluation  
11. Secondary notation

---

Although the originators of the framework predominantly use the term “notation” for referring to the object being evaluated, thus perhaps indicating a certain character- or symbol-based system, the CD framework can actually be employed to evaluate any kind of cognitive artifact [42].
9.3. EVALUATION OF CAMBRIA 2

While the authors use the term “dimensions” which, in most cases implies orthogonality (i.e., a change along one dimension does not lead to changes in other dimensions), this is not the case with the CD framework; in fact, a certain score in one dimension might influence scores in other dimensions. Blackwell and Green [41] present a diagram of “trade-offs”, whereby:

- Higher abstraction can reduce viscosity, increase the need for lookahead, increase hidden dependencies, and increase visibility.

- Secondary notation can increase viscosity, because if one changes the model, one also has to update secondary notations.

In other words, the activity of design necessarily involves tradeoffs, and it is up to the interaction designer to carefully weigh arguments in favour, or against, any particular design decision.

In the following, I provide analytical reasoning behind CAMBRIA 2’s scores along each of the cognitive dimensions listed above, and propose one or several design interventions which would ameliorate any weaknesses found through such reasoning, using the following format:

(Analytical evaluation, per cognitive dimension.)

• (Design intervention.)

9.3.1 Visibility

This particular dimension deals with the visibility (or lack thereof) of the items (elements) constituting an HCI design, and places the design along the (low visibility $\leftrightarrow$ high visibility) axis, based on whether the elements of the model, as well as their inner details, are or are not visible, perceptible, or obvious to the eye, and to what degree. In general, the more visible the elements of a computational model are, the better.

In CAMBRIA 2, if one first considers that the user can view multiple design states at once, visibility is very high at the level of individual design states, especially when compared with conventional single-state systems. The visibility of model elements (i.e., items) is likewise high—what you see is what you get. The visibility of links among items can be considered good, since the user can view which items are linked to each other through the brushing mechanism. Since CAMBRIA 2 is based on 2D parametric models, visibility will likely remain good even for more intricate 2D models.

On the other hand, one-way dataflow constraints governing the geometric configuration of items, are less visible, in the sense that the user can obtain an indication of constraints associated with a specific item, only by raising the contextual (pop-up) menu when right-clicking on the item, thus providing a list of available constraints for the item, as well as the options to remove currently
active constraints from said items. In other words, contextual menus can be considered to be a workaround for reducing the effect of hidden constraints.

**Design intervention.** In order to remedy the discovered weakness (low visibility of constraints), one might adopt different approaches.

![Figure 9.22](image.png)

**Figure 9.22:** A way to increase visibility of constraints in CAMBRIA 2, by superposing constraint “boxes” (here coded in red) connecting input to output items/variables, whenever an item is selected in the alternative.

For example, constraints could be represented as graphical items embedded directly in the model, as shown in the mockup in Figure 9.22, and then toggled on or off (i.e., shown or hidden) as needed. I suggest a way to embed constraints directly in the parametric design model in one of my own research reports [190]. Also, Maleki *et al.* present the “programming in the model” approach [235, 234, 236] which provides additional ideas about how to embed and visualize computational logic directly within a parametric model.

Among other potential remedies for the partial visibility of dependencies in CAMBRIA 2, the tool might provide a separate “dependency graph” view, as is common with current parametric CAD modellers. In this case, standard graph visualization techniques would apply to arrange graph nodes and edges, see e.g., [210, 190]. As another idea, a separate details-on-demand view (one per alternative) could display a list of constraints attached to all the selected item(s), and express them as e.g., a sortable data table.
9.3.2 Juxtaposability

This dimension is closely related to the visibility dimension, with the difference that it describes how easy it is to display objects side-by-side, along the following aspects: (1) different objects; (2) different parts of an object; (3) different expressions of the object being displayed. Thus, the tool should afford the user the ability to easily view various objects (i.e., alternatives or items); simultaneously display different parts of the same object; and display objects, or their parts, using different expressions such as images, thumbnails, text, data tables, and other types of representation.

With regards to aspect (1), CAMBRIA 2 is effective in juxtaposing different objects (alternatives); it provides one view per alternatives, and several methods for arranging them into different layouts. However, the capabilities for viewing different parts of a design alternative, within the same view allocated to the alternative, are missing. Likewise, CAMBRIA 2 lacks the ability to display objects using different representations; at the moment, all alternatives are rendered as 2D images only, and likewise all collections are rendered as arrangements of 2D views. As a side note, CAMBRIA 1 had the ability to render a collection both as an arrangement of thumbnails of alternatives, as well as a data table listing each alternative as a row of alphanumerical values such as the alternative’s name, date of creation, the count of items contained with the alternative, and so on.

Design intervention. Due to considerations described above, the next iteration of the tool should allow the user to create separate smaller views (or “viewlets”) within the view displaying a design alternative. As an example from existing systems, many plain text editors allow the user to “split” the current view into two or more views. Likewise, in addition to the “main” view or “master” view, each viewlet could then display a different part of the alternative, and at different zoom levels. Furthermore, such viewlets could be programmed to be constructed or destroyed automatically i.e., as a function of the part currently being viewed in the master view; alternatively, they could be opened and closed by the user manually.

Next, for any construct of importance in CAMBRIA 2 (collections, alternatives, and items), the tool should provide means to display each of these constructs in a different manner, such as an image, thumbnail, informative text, or data table. Each of the “viewlets” described above, as well as the master view, should have the ability to switch between different representation types for any of those concepts that can all be displayed at the same time.

9.3.3 Hidden Dependencies

In many systems, notations, or data models, an element $a$ frequently depends on another element $b$. For example, in CAMBRIA 2, such dependencies include constraints and links between items$^4$.

$^4$Note that Green et al, when talking about “hidden” dependencies, refer to unplanned dependencies, or dependencies that appear as a surprise to the user. In CAMBRIA 2, constraints and links are obviously “explicit”, however I will
If such dependencies are hidden from the user of the system, perhaps due to a bad HCI/interaction design, it might lead to excessive user error rates, confusion and excessive debugging activities [110, 281, 138].

The analysis along the related dimension, that of *visibility*, has shown that the design of CAMBRIA 2 could be improved by providing means to visualize and manipulate constraints directly within the model; while not exactly hidden, constraints for an item can be examined only by invoking the contextual popup menu listing existing, as well as permissible constraints for the item. As for links between items, at the moment CAMBRIA 2 allows the user to detect this type of dependencies via the brushing mechanism, which visually highlights linked item as soon as the user moves the mouse pointer over an item of interest. Hence, while both the constraints as well as links cannot exactly be called “hidden”, they cannot be called “fully visible” either: both dependencies thus score “averagely” or “poorly” along this dimension.

Next, one might argue that the of *reachability* of nodes in a parametric DAG could represent a potential hidden dependencies problem as well. In other words, if two constraints $a \rightarrow b$ and $b \rightarrow c$ exist, the user might be interested to know whether item $c$ depends (directly or indirectly) on item $a$ as well. In general terms, if a path $v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_n$ exists in the parametric DAG, currently there is no way in CAMBRIA 2 to indicate that $v_n$ depends (indirectly) on item $v_0$, and on all the preceding items as well.

![Figure 9.23: A mockup of the interaction functionality for detecting linkage among instances of item #6.](image)

**Design intervention.** Figure 9.22 already suggests how to improve the visibility of constraints. The other type of dependencies, that of links between items, can at the moment be revealed or “unhidden” through brushing. However, perhaps this might be improved on, for example by modifying

still review the main “dependency” constructs (constraints, links) within this dimension.
the “spot-the-differences” differencing method that I present in § 10.2.5.5 (p. 260).

As per Figure 9.23, if the user were to select an item, all the linked items in other active design alternatives could be visually emphasized by a local “focus” area, while the rest of the model could be darkened out for easier viewing. Multiple local focus areas could be rendered if the user selects multiple items. Advantage of this technique would include (1) better visibility of which items is linked to which item(s), and (2) not having to rely on colour coding to indicate linked items, as is currently the case with brushing which uses red colour for this purpose.

For solving the reachability problem, i.e., whether an arbitrary item $v$ is dependent (directly or indirectly) on another item $u$, possible strategies include [76]: (1) standard BFS traversal of the DAG, with linear time complexity $O(|V| + |E|)$, or by (2) pre-computing the transitive closure of the set $E$ of directed edges, which can be computed with time complexity $O(|V|^3)$ and space complexity of $O(|V|^2)$ using the Floyd-Warshall algorithm. The second method would therefore probably be more effective than the first method, since once the transitive closure has been computed, the user can lookup whether an item is dependent on another one rapidly i.e., in constant time $O(1)$.

### 9.3.4 Viscosity

A term borrowed from fluid mechanics, viscosity describes the extent of a fluid’s resistance to forces acting on it. For example, honey is much more resistant to changes than water. Likewise, in HCI/IxD, a highly viscous system is the one that does not allow for easy and effortless local (i.e., relatively small) changes to the underlying computational model. In general, the less viscous the system is, the better.

At the level of design states, viscosity can be considered to be very low (i.e., CAMBRIA 2 has an excellent score) because the user can edit all active alternatives in parallel. Thus, instead of separately loading and then modifying each alternative in turn as is the case with single-state tools, in CAMBRIA 2 the user can load multiple design design alternatives at once, keep them loaded at all times, and then execute editing operations, either per alternative or in parallel if so desired.

At the item level, since CAMBRIA 2 is a tool based on the direct manipulation paradigm, it is in general rather easy to exert changes to items’ positions and sizes, by first selecting them, and then dragging them to a new location on the canvas. Likewise, changing items’ property values is also easy, by choosing the appropriate menu option (for instance, Increase pen width in the Edit menu) or clicking on the toolbar button. In particular, all the operations listed above are easy to execute for multiple alternatives in parallel, which is consistent with the main goals of CAMBRIA.

Viscosity of adding, deleting, and modifying constraints can, however, be considered to be high, perhaps with the exception of adding “parametric point on line” constraints which are, as described on page 192, taken care of through the subjunctive snapping technique. Deleting and
modifying constraints can be universally considered, however, to be very viscous in CAMBRIA 2.

**Design intervention.** The subjunctive snapping technique shows the way for decreasing viscosity of operations dealing with constraints within the context of multiple design alternatives. For example, the operation of deleting the “parametric point on line” constraint from a point could be made less viscous by parallelizing it; as the user chooses to convert a parametric point on line to a free point, the user might select the “All alternatives” scope beforehand (perhaps by switching a flag in a separate GUI control), thus deleting the same constraint from all the linked points in other alternatives. The same holds for the operation of modifying this constraint (i.e., changing the value \( t \)), by parallelizing it in the same manner. Likewise, other types of constraints should be parallelized, either in terms of creation, deletion, or modification, using this suggested approach.

Additional level of parallelization of constraints might be achieved by the approach shown in Figure 9.22 on page 196, where constraints are visualized by their own types of items within the model. In this case, the user might employ the already-implemented parallel-select mechanism to select multiple constraints, within multiple alternatives, and then perform appropriate parallelized operations on them, such as delete or modify.

Yet another means to decrease the viscosity would be a global search-and-replace method, which could exchange entire sets of items satisfying search criteria, by some other type of item. For instance, all circle items might be replaced with ellipse items; at the moment, if the user were to do this in the current version of CAMBRIA 2, he or she would have to replace each such item manually and individually.

From the considerations above, I conclude that, for multi-state systems, a conscious effort should be made to parallelize as many operations as possible, in order to decrease viscosity scores for managing multiple models at once.

### 9.3.5 Terseness

Terseness measures whether a notation is short and concise i.e., how well it expresses and captures intended meaning in as few words or symbols as is possible. In general, the more terse the system or notation is, the better it is, of course up to a limit where the notation starts losing its expressive power. A less terse (or diffuse, verbose) system, on the other hand, prevents the reader/viewer of efficiently packing the notation into his or her working memory because more symbols demand more visual scanning from the user [138]. Thus, exceedingly verbose notations will, as a rule, slow down apprehension and understanding of the notation.

In CAMBRIA 2, the notation consists of graphic items (points, lines, curves, . . . ) on screen, with constraints governing their layout, and is therefore as terse as is possible considering that the notation is in this case equal to the intended final representation (2D parameterized vector drawings). One might compare this representation with an alternative one, for example a scene or
9.3. EVALUATION OF CAMBRIA 2

Design alternative described using some kind of formal syntax. For instance, a single Bézier curve might be described by the following string of symbols based on the SVG standard\(^5\):

\[
\text{<path d="M100,100 C150,150 300,200 300,450"/>}
\]

Thus while the notation above consists of around ten objects (words), the notation of CAMBRIA 2 consists of around five objects only (two end control points, two tangent control points, and the curve). Likewise for other items, and entire drawings; if one were to make an an estimate, then the notation of CAMBRIA 2 is maybe 50\% terser than the equivalent SVG notation.

As for constraints, again using the alternative SVG notation, items might be named via the “id” attribute, and a constraint connecting two or more items could then refer to the items using their ids, thus achieving a degree of terseness using the SVG (XML) format. For example, if one considers two free points \(p_1\) and \(p_2\), and a “project \(p_1.x\) onto \(p_2.x\)” constraint, one might write:

\[
\begin{align*}
\text{<point id="c1" cx="200" cy="100" r="10"/>} \\
\text{<point id="c2" cx="350" cy="240" r="10"/>} \\
\text{<constraint type="projectXcoord" source="c1" target="c2"/>}
\end{align*}
\]

Again, I would argue that the SVG notation above is less terse than the equivalent notation in CAMBRIA 2, where constraints such as the one above are expressed with a dotted line connecting two points \(p_1\) and \(p_2\) (when shown).

**Design intervention.** Not applicable; CAMBRIA 2 is optimally terse.

### 9.3.6 Proneness to Error

This dimension measures the degree to which the system causes the user to make inadvertent, small errors and mistakes (or “slips”) as he or she works with the system. If one considers CAMBRIA 2 as a single-state tool, i.e., when the user works with just a single design alternative at a time, I would judge the proneness-to-error score to be **good**, i.e., equal to any other single-state tool.

However, in the context of multi-state tools, I feel that this dimension should be reinterpreted somewhat. Firstly, I consider that the proneness to error is high in CAMBRIA 2, when one considers *parallel editing operations*, for which initial evidence shows that they place high demands on the user’s cognitive performance. For instance, when moving an item in parallel for several design alternatives, the end positions of the item might not be as desired by the user, for several or most design alternatives. However, this discrepancy between the expected/desired and achieved position should probably not be characterized as a “slip”, but merely as an “unwanted divergence” due to the user’s inability\(^6\) to effectively track changes in all active alternatives, or simply due to

---

\(^5\)SVG, or Scalable Vector Graphics standard by W3C: [http://www.w3.org/TR/SVG/].

\(^6\)Here “inability” denotes the intrinsic limitations of human cognition in general.
the inner composition of the alternatives constraining all permissible positions of the items within their respective alternatives.

The point remains, though, that since parallel editing operations lead to simultaneous changes in multiple alternatives, these changes may need to be fine-tuned further in order to produce acceptable outcomes, for each respective design alternative. Whether these divergences (small or large) caused by parallel operations are regarded as “inadvertent errors”, or as natural consequences of using parallel operations characteristic of multi-state tools, is open to interpretation. In case of the latter, one might perhaps introduce a correlate of the “Proneness to Error” dimension, perhaps named as “Proneness to Divergence” or “Controllability of States”; if so, then the score of CAMBRIA 2 is poor along this dimension, probably as a tradeoff for increased power along the visibility and viscosity dimensions (because one can view and edit many design states at once).

**Design intervention.** Not applicable\(^7\). The ability to view and edit multiple design alternatives in parallel can be considered as powerful, novel functionality that allows the user to quickly modify multiple design states at once, but obtained at the cost of increased “state instability”, or model divergences that must be tweaked afterwards by the user in order to produce acceptable designs. In other words, high proneness-to-error due to parallel editing is a tradeoff for excellent visibility and viscosity scores.

### 9.3.7 Hardness of Mental Operations

Is interaction with the system “hard” in the sense that the tool imposes high demands on the user’s cognitive resources and capabilities? Must the user recur to other, external tools and utilities (such as e.g., hand-written notes, other software, etc.) to successfully conclude tasks with the tool? In other words, the goal here is to come up with a tool or system that allows its user to work out the task *in his head*, without having to recur to external aids in order to complete the task.

First off, compared to conventional (non-parametric) tools, parametric tools already “may require additional effort, may increase complexity of local design decisions, and increase the number of items to which attention must be paid in task completion”\(^8\). Initial evidence, both collected in this work as well as in related work, shows that (some, or all) parallel editing operations impose high cognitive demands on the user, due to the need to track changes taking place concurrently in multiple places at once. In this sense, this provides the first, obvious justification for giving a poor score along this dimension, not only for CAMBRIA 2 but for all multi-state tools with parallel editing capabilities. Thus, parallel editing, and especially in the context of parametric modelling, is hard.

This trickles down to other aspects of multi-state tools. Users of parametric modellers should (at least in theory) master a certain body of mathematical knowledge \(^411\) in order to be able to

\(^7\)Or better yet, “probably not applicable”. Future work may or may not discover means or techniques for limiting state instability while preserving excellent visibility and viscosity scores.
use such modellers effectively. Applying such mathematical knowledge might be cognitively more demanding in the context of multi-state tools; for example, differing mathematical methods might be associated with different instances of the same subjunctive item, thus forcing the user to pay attention to more entities in the design process than is the case with single-state tools.

Another issue concerns the activity of debugging parametric models, since they can be regarded as code bases as well. Just like any other programming project, they are plagued by bugs or computer program defects. The activity of “debugging” requires the user to locate and correct parametric nodes that led to unexpected result [260]. In all likelihood, highly cognitively demanding activity of debugging will be exacerbated by the presence of concepts unique to CAMBRIA 2, such as subjunctive (linked) items, and by the existence of separate constraint solving processes (at least one per alternative).

Finally, the tools based on one-way dataflow constraints must keep their parametric models free of cycles at all times. This makes it “necessary for users to plan ahead how the features of the model should be controlled and may require manually uncoupling relationships that would otherwise give rise to cyclic dependencies” [38]. This problem, likewise, might be exacerbated in multi-state tools, since a parallel editing operation that leads to a legal (acyclic) model in one alternative might lead to a cyclic DAG in another alternative; this constitutes yet another potential issue that the user must pay attention to.

**Design intervention.** Regarding the parallel editing operations, similarly to the explanation that I provided in the proneness to error section, increased power along the visibility (for viewing) and viscosity (for editing) dimensions for multi-state tools clearly leads to higher cognitive demands placed upon the user. Again, this can be considered as a design tradeoff, and a price that must be paid for being able to edit an entire set of models. It should be noted, one ad hoc way to work around this issue is that the user is not obliged to use parallel editing operations at all if he feels that they are too hard; the user can edit separate alternatives on the one-by-one basis, perhaps by activating the solo status for each alternative. This workaround will still enable the user to enjoy the benefits of having multiple models loaded in the tool, thus facilitating model switching as well as viewing and comparison, without having to expend cognitive resources on executing hard parallel editing operations.

As for debugging multiple states in parallel, I might imagine equipping each alternative with its own instance of a debugger, i.e., a separate computational tool that allows the user to go through the data propagation process in a stepwise fashion, thus facilitating the discovery and elimination of defects. At this point, however, it is not completely clear how one would implement a debugger for parallel operations: either multiple debuggers (each attached to an alternative) that work in a synchronized manner, or a global debugger that tracks all operations taking place at once, within a single debugging process.

To ameliorate the problem of user having to pay attention to cyclic dependencies, the tool could
either break up the loops automatically [389], or switch off the solver based on data propagation, temporarily employ a solver based on numerical methods, solve the model, and revert back to data propagation [361, 362, 46, 349, 203, 38]. For subsequent instances of dataflow solving, the solver then utilizes saved (cached) values, and performs data propagation along acyclic branches of the global DAG only.

I conclude this section by emphasizing that multiple potential sources of complexity exist within the context of multi-state tools, and which thus place high cognitive demands on the user. This is an overview of some most obvious ones; others are likely to be discovered as these tools start being used in design workflow.

9.3.8 Abstraction Gradient

Abstraction gradient measures the ability of the notation at hand to group objects together, and then treat such groups of objects as a single entity. The activity of grouping objects together (and its twin operation of ungrouping composite items) thus implies the act of creating new composite objects, from existing objects (see the discussion in Section 7.2.2 on page 134 about “simple” and “composite” items). Depending on this ability, a notation can be characterized in broad strokes as either abstraction-resistant, abstraction-neutral, or abstraction-promoting⁸.

In this work, both CAMBRIA 1 and 2 lie rather low along this dimension i.e., both high-fidelity prototypes can be considered to be abstraction-resistant, simply due to the fact that the ability to group items has not been implemented, although I gave the provision for that in Chapter 7, detailing the conceptual design. At the most basic level, in CAMBRIA 1 and 2, the only items that can be regarded as groups of sub-items, are elements such as points (groups of \(x\) and \(y\) coordinates), line segments (i.e., groups of two points defining the end positions), and curves (a series of control points). Also, the user is able to treat all currently selected items as a “selection”. However, beyond this point, no significant grouping ability exists in any of the research prototypes developed in this work.

Design intervention. Due to the considerations described above, two new commands Group items and Ungroup item should be added to CAMBRIA 2. Upon grouping a set of items, the newly created object should then be considered as just another item with full rights, in the sense that it should be given its own unique identifier, a name, and geometric extent; the user can then manipulate this composite or group object, create alternatives of it, and so on, just like with any other item used in CAMBRIA 2.

A potential complication stemming from this intervention, in addition to the increased complexity of the application, is the question of regrouping and redistributing one-way dataflow constraints.

⁸Green et al in fact use more anthropomorphic expressions abstraction-hating, abstraction-tolerant and abstraction-hungry.
9.3. EVALUATION OF CAMBRIA 2

(1) **Grouping items.** Let us assume that items $i_1, i_2, \ldots, i_n$ are grouped into a composite (group) item $g$. Let us assume that each of the grouped items $i_k$ emits or receives values from a set of other items $I(i_k)$ in the model. Then, existing constraints might for example simply be “upgraded” (or “promoted”) so that all the affected items $I(i_k)$ emit to, and receive from, new group object $g$ instead.

(2) **Ungrouping an item.** The reverse problem occurs when a composite item $g$, consisting of parts $i_1, i_2, \ldots, i_n$, is dissolved or ungrouped. Suppose that a number of constraints exist from $g$ to a number of other items $I(g)$ in the model. One possible approach for solving this issue would be to store constraints for $i_1, i_2, \ldots, i_n$ at the moment $g$ was created in the past, and then restore them when the user decides to ungroup $g$.

Other opportunities for improving the abstraction capabilities of CAMBRIA include (1) the “subjunctive item browser”, and (2) global search-and-replace function (also described in the viscosity section). The first feature, the “subjunctive item browser”, would allow the user to view and examine linked instances as a whole i.e., as “subjunctive items”, which was in fact suggested by one of the participants in CAMBRIA 1 study [P4 1:05:37]. A global “search-and-replace” function could retrieve sets of items fulfilling some (arbitrary) criteria, such as items’ names, types (points? lines? curves?), pen colour, curvature values, and many other possible properties, and then replace such sets with other sets of new items; for example, a global search-and-replace operation might exchange all the line segments in the model with a simple (one-part) Bézier curve.

9.3.9 **Premature Commitment**

This dimension measures the extent to which a tool forces its user to commit to certain design choices, although the user may still not be in possession of all the necessary information in order to make an optimal decision. As one undesired consequence of this lack of information, it thus may turn out later on that the initial choice was wrong. A tool that scores poorly along this dimension can be improved directly, i.e., by somehow improving on the premature commitment dimension, as well as indirectly: for example, by decreasing the tool’s viscosity i.e., modifying it so that it allows for easy modifications, and thus correction of poor early design choices and decisions.

I should first mention that in contemporary (commercial) parametric modelling tools, discrete modification and variation of the model is to some degree separate from continuous modification and variation. For example, the user might first decide to focus on the structural properties of the system, by editing the data propagation “graph” (i.e., the model’s DAG), instead of varying the values of the model’s free variables to obtain different geometric instances of the model. This partial separation between the graph-related tasks and the instantiation-related tasks results in

---

9 Although the user may be focusing on the structure, he still observes a (variational) instance of the model in order
both the beneficial and detrimental effects [8]. One beneficial consequence is that the users can defer (i.e., postpone) their instantiation decisions, thus effectively avoiding premature commitment for one aspect of parametric design work.

While CAMBRIA 2 does not offer this (partial) separation of structural and variational tasks, there is no particular order that must be followed creating and modifying a model. Therefore, the CAMBRIA 2’s score along the premature commitment dimension can be considered to be acceptable.

Design intervention. CAMBRIA 2 does not force the user to make premature commitments, however as it is used in design, new challenges may be observed.

9.3.10 Progressive Evaluation

This dimension measures a tool’s or notation’s ability to enable the user to evaluate, test, debug, and/or execute “programs” which are not complete or fully specified by the user yet. For example, Green and Petre [138] provide an example in the case of programming language Pascal, where programmers in the early phases of software development use “skeleton” procedures (i.e., procedures with nothing but a declaration and empty body), thus enabling them to run and test even partially written programs, thus getting around the Pascal's poor score along this dimension. The same with the markup language HTML and the stylesheet language CSS, for which the progressive evaluation score is likewise rather poor [140], since a web page must be shown in full in a web browser before the user is able to visually evaluate the code written so far.

In contrast to languages above, in CAMBRIA 2, if one considers each design model to be a “program”\(^{10}\), it has an excellent score along this dimension, since the users can visually evaluate each drawing, and augment it with additional logic (i.e., constraints) as they go along. Items (points, lines, curves, ...) can be added as desired to a scene, initially without any constraints at all, and added later at will by the user.

Design intervention. Not applicable; CAMBRIA 2 allows for easy progressive evaluations of 2D parametric models.

9.3.11 Secondary Notation

Secondary notation denotes information that is not a part of the design state in the strict sense, but which is nevertheless used to create or apprehend it. Such information includes e.g., labels, various types of annotation (textual, audio, video, sketching, ...), comment sections, and formatting conventions (such as e.g., indentation in programming languages). In other words, secondary

---

\(^{10}\) Which can be done in a completely valid fashion, since constraint-based computational models belong to the declarative programming paradigm.
notation allow the user to “adopt the preferred cognitive strategy even in apparently adverse circumstances” [139].

As an illustrative example, when faced with a task to write a complex computational method using a formal language, the user might recur to first outlining such a method in pseudo-code using the “comment” construct in the language, and then commence with implementing this pseudo-code in the programming language’s formal syntax itself.

CAMBRIA 2 can be said to score poorly along this dimension, since it offers no secondary notation capabilities; all information managed within the tool is exclusively used to describe design states in the strict sense.

**Design intervention.** Possibilities for adding secondary notation are numerous. Starting with textual annotation, each item (point, line, curve, . . .) might be augmented with the ability to reference one or more “comment sections” which in turn might be implemented as pure text, or even as full formatted-text pages allowing for unlimited prose-based annotation.

### 9.3.12 Custom Dimension: “Alignment to Chosen Metaphors”

Finally, if one were to evaluate CAMBRIA 2 against the golden standard and the quintessential technique for designing, that of sketching, how would working with CAMBRIA 2 fare with regard to the designer armed with just the paper & pencil combination? In a sense, the following exposition might be considered as yet another, “mini” evaluation of the prototype, along a new, hypothetical “Alignment to Chosen Metaphors” dimension. I note here that, of course, that this “custom” dimension is not a part of the CD framework.

With CAMBRIA 2, the desired outcome is being able to produce and work with multiple (2D) parametric models, while abiding by the high-level metaphors defining the conceptual model (see § 7.1 on p. 130): design gallery, tracing paper, and evolutionary tree. To distill this further, the desired overall outcome with CAMBRIA 2 can be stated as: to be able to work effectively with a number of related parametric 2D drawings. In the following, I will thus examine the following three aspects, or the “degree of alignment”:

1. Comparison with sketching.
2. Assessment of benefits of parametric design tools.
3. High-level metaphors defined in conceptual design.

If we start at the level of individual alternatives, and remind ourselves about the beneficial properties of sketches (see § 2.6.1 on p. 25), sketches are [404, Ch. 5]: quick, timely; inexpensive; disposable; plentiful; suggestive; with minimal detail needed; with appropriate level of refinement; and ambiguous. How does CAMBRIA 2 fare along these qualities?
Quick. In CAMBRIA 2, it suffices to load the tool, create a new or open an existing design project, and then either create a new drawing or continue working on an existing drawing. While probably not as quick as grabbing a sketchbook and a pencil, it is still rather quick. However, drawing with a pencil on a sheet of paper is probably much faster than by dragging the mouse pointer on screen. This could be ameliorated by a version of CAMBRIA 2 running on touch interfaces, whereby the user draws on screen using either his finger, or a stylus.

Timely. Can a drawing in CAMBRIA 2 be provided when needed, just like a sketch can? For CAMBRIA 2, one requires a reasonably capable personal computer which are still much less ubiquitous than paper and pencil combination. In addition, initially one needs to download, setup, and learn CAMBRIA 2, while anyone can start sketching immediately.

Inexpensive. Drawings in CAMBRIA 2 are expensive to make, compared to sketches. Creating a drawing in CAMBRIA 2, especially one with a number of constraints, requires time, thus increasing its cost in terms of this particular resource (time). Still, editing existing drawings in CAMBRIA 2 might be cheaper than sketching, since the user can take advantage of undo/redo capability, as well as generative capabilities of parametric models.

Disposable. Yes, drawings done in CAMBRIA 2 are disposable. In addition, since CAMBRIA 2 preserves all states in its design history database, anything that was “disposed” of in the past, can be easily retrieved at any point of time.

Plentiful. Yes, drawings in CAMBRIA 2 do not exist in isolation, but rather as a part of a collection or a series. In fact, on this aspect CAMBRIA 2 probably fares better than sketchbook + pencil combination, because duplicating and modifying duplicates is much quicker and easier.

Suggestive. Probably worse than sketches, which permit a much wider range of drawing styles such as hatching, tonal values, gradations, contour styles, etc. One way to improve CAMBRIA 2 would be to implement non-photorealistic (NPR) or illustrative rendering techniques. Other potential ways could include implementing “what-if” scenario capabilities, as well as the ability to substitute items with their own alternatives.

With minimal detail needed. Probably as good as sketches.

With appropriate level of refinement. With CAMBRIA 2, one could in theory produce rather refined 2D drawings, however probably still less so than sketches.

Ambiguous. CAMBRIA 2 could probably improve on this aspect, for example by implementing the aforementioned NPR or illustrative rendering techniques. Also, by deferring design decisions, incomplete models can be ambiguous as well.
According to the considerations above, while CAMBRIA 2 is moving in the right direction, additional features should be implemented in order to approximate the set of qualities of sketches listed above.

As another important part of the desired outcome for CAMBRIA 2, one would of course want to examine whether it brings about the benefits of parametric design tools. How does CAMBRIA 2 fare in this respect? Benefits of parametric design include [8]:

1. **Enhancing the search for designs well adapted to a set of design requirements.** This benefit implies a certain set of computational abilities that connect the end design(s) with the requirements on the design (for example, a set of data describing the environment in which the end artifact will be embedded; cost and quality criteria, and so on). CAMBRIA 2, being a research prototype, does not provide this benefit.

2. **Discovery of new geometric forms.** This is easy to achieve in CAMBRIA 2, for example by simply duplicating an alternative, and then further modifying the copy, for example by varying the position of a free point, thus influencing and modifying the entire model.

3. **Reduction of time required for change and reuse.** This benefit is related to above; the user can easily change an alternative through any of the interaction techniques or methods described. Likewise, reuse is facilitate either through duplicating entire alternatives, or just a subset of items contained within alternatives.

4. **Better understanding of the structure of the designed artifact.** CAMBRIA 2 also provides this benefit, through various means to examine the structure of related designs through interaction functionalities such as brushing and parallel dragging. Thus, although multiple alternatives might be different in the geometric (variational) sense, they may still be the same in structural sense.

Another important part of the overall desired outcome is that CAMBRIA 2 faithfully follows the ideas or metaphors chosen: design gallery, tracing paper, and evolutionary tree. As for the design gallery metaphor, various arrangement methods provide this outcome. Tracing paper, a metaphor for placing alternatives on top of each other while being able to see through them, has not been implemented in CAMBRIA 2 due to time constraints (this metaphor was implemented in CAMBRIA 1 though, see Figure 8.5 on page 148, showing the parallel editor based on juxtaposition). Finally, the evolutionary tree metaphor has been followed by providing the “arrange by design history” arrangement type, see Figure 9.6 on page 180.

**Design intervention.** The analysis above has already suggested a number of potential design improvements. To recapitulate, at the drawing level, in order to approximate beneficial properties of sketching, I suggest the following design recommendations: (1) implement CAMBRIA 2 on
touch surfaces, so that the user can utilize either a stylus or his finger for drawing, in that quick and intuitive manner characteristic of sketching; (2) cross- and multi-platform implementation of CAMBRIA 2, which would make the tool widely available and thus better on the timely criterion; (3) provide richer library of drawing styles and elements, for deeper level of refinement; and (4) implement non-photorealistic (NPR) or illustrative rendering techniques, in order to approximate the ambiguous and suggestive nature of sketches.

At the level of interaction metaphors, improve on the design gallery metaphor by providing additional arrangement methods, and improve on the tracing paper metaphor by re-implementing an improved version of the “superposition parallel editor” initially demonstrated in the CAMBRIA 1 prototype.

9.4 Discussion

While CAMBRIA 2 presents a number of advances and improvements compared to CAMBRIA 1, the analytical evaluation of CAMBRIA 2 based on the CD framework has nonetheless uncovered a number of areas of potential improvement and exciting future work. Table 9.1 summarizes the scores of CAMBRIA 2 along the set of cognitive dimensions, further split into sub-scores (per aspect of the tool).

<table>
<thead>
<tr>
<th>Dimension:</th>
<th>Score:</th>
<th>Design intervention(s):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visibility (alternatives, items)</td>
<td>good</td>
<td></td>
</tr>
<tr>
<td>Visibility (constraints, linkage)</td>
<td>adequate to poor</td>
<td>“constraint boxes”; graph view; list of constraints</td>
</tr>
<tr>
<td>Juxtaposability (alternatives)</td>
<td>good</td>
<td>–</td>
</tr>
<tr>
<td>Juxtaposability (parts of alts.)</td>
<td>poor</td>
<td>“viewlets”</td>
</tr>
<tr>
<td>Hidden dependencies</td>
<td>adequate to poor</td>
<td>local “focus” on linked items; transitive closure; see also Visibility interventions.</td>
</tr>
<tr>
<td>Viscosity (alternatives, items)</td>
<td>good</td>
<td>–</td>
</tr>
<tr>
<td>Viscosity (constraints, linkage)</td>
<td>poor</td>
<td>parallel constraint/linkage edits; search&amp;replace</td>
</tr>
<tr>
<td>Terseness</td>
<td>good</td>
<td>–</td>
</tr>
<tr>
<td>Proneness to error</td>
<td>adequate</td>
<td>(design tradeoff)</td>
</tr>
<tr>
<td>Hardness of mental operations</td>
<td>adequate</td>
<td>(design tradeoff)</td>
</tr>
<tr>
<td>Abstraction gradient</td>
<td>poor</td>
<td>grouping; “subjunctive item” browser; search&amp;replace</td>
</tr>
<tr>
<td>Premature commitment</td>
<td>good</td>
<td>–</td>
</tr>
<tr>
<td>Progressive evaluation</td>
<td>good</td>
<td>–</td>
</tr>
<tr>
<td>Secondary notation</td>
<td>non-existing</td>
<td>add features for annotation</td>
</tr>
</tbody>
</table>

Table 9.1: Evaluating cognitive dimension of CAMBRIA 2: Summary table.

Scores in the second column are based on the simple three-level descriptive scale (“poor”, “adequate”, and “good”) as per analysis for each dimension. The third column of the table also provides a condensed summary of recommended design interventions for the next, improved iteration of CAMBRIA. While most of the design interventions have been well described in the preceding sections, I feel that grouping functionality requires further elaboration. After that, I
conclude this chapter by discussing the overall system architecture of multi-state tools.

**Groups and Parts**

The analysis of CAMBRIA 2 along the abstraction gradient dimension has resulted in a poor score in this aspect. All CAMBRIA 2 models are flat, as there is no grouping and ungrouping ability, with the exception of “select items” capability. The set of currently selected items can therefore be abstracted as the “current selection”, and then be worked with as a single, separate object. The analysis has already suggested some design interventions, such as enabling (permanent) grouping and grouping of items, and how these operations might lead to issues with bundling, unbundling, inheriting, and then re-distributing constraints and other types of linkage.

In general, the question of modularity of designs has been subject to some debate. For instance, Simon [342, 339, 343] writes about “hierarchic”, “nearly-decomposable systems” which can be sub-divided into a small or moderate number of sub-systems. According to Simon, the links or associations among parts of a sub-system are supposed to be much stronger than links among the sub-systems themselves. Also, the rate of propagation among parts of a sub-system are supposed to be much greater than rates of propagation among sub-systems. It is not clear how these concepts would apply to parametric modelling; I surmise and speculate, however, that Simon’s notions would find natural use in very large and complex computational models which require either solvers based on numerical methods, or dataflow solvers which would heavily pre-cache computed values in large parts (or sub-graphs) of the global parametric model. Solvers based on numerical methods can easily model slower or faster propagation rates (e.g., by solving a set of differential equations modelling diffusion processes); solvers based on data propagation less so.

In product design, modularity implies the ability to decompose into subassemblies and components. In programming languages, the word module refers to a “manageable portion” of the code; the success of IT industry as whole can be attributed to its ability to build increasingly more complex modules, from simpler modules. In design engineering, a module can be defined as “a component or group of components that can be removed from the product non-destructively as a unit, which provides a unique basic function necessary for the product to operate as desired”. Modularity, therefore, permeates all aspects of practical product design.

On a more fundamental and abstract level, the “part-whole science” is premised on identifying, investigating, and using parts and wholes. According to this new branch of science, there are (1) multiple ways, or partitioning frames, to partition a model; (2) each partitioning frame

---

11“Small” or “moderate” here imply the order of magnitude equal to 10 or 100 at most. This is due to the fact that a system of, say, 1000 or more sub-systems will, according to Simon, inevitably experience the process of “coalescing” of these 1000 sub-systems into 10 or so “new” sub-systems, thus again ending up with a system with 10 or at most 100 sub-systems.
provides a different yet incomplete explanation of the model; and (3) multiple partitioning frames can be integrated or unified into a single “vision” of the model. Likewise, one might imagine the capability of partitioning a parametric model into several (or many) such “partitioning frames”, each explaining a different aspect of the model:

(1) The “historical” explanation (or partition) that would place a design alternative, or an individual item, “in their contextual whole”, thus creating a narrative, biography, or phylogenetic tree.

(2) The “structuralist” explanation which would show how a design alternative, or an item, developed in the morphogenetic or morphological sense.

(3) The “mechanistic” or “interactionist” explanation of how a design alternative, or an item, interacts with any of the following: contained items, container item, container alternative, or container collection. In particular, how items interact with each other, thus explaining the “system behaviour” of the entire design alternative, or just an individual item.

According to Winter [405], it holds that: (1) no single explanation reduces the remaining explanations; (2) no single explanation is fundamental (however, each explanation can be considered to be important); (3) each explanation “has its legitimate norms, explananda, and aims”; (4) each explanation partitions a model into parts in its own specific manner; and (5) despite their differences, each explanation is still a “part-whole” explanation.

In practical computing, the importance of modularity is clearly demonstrated by the commercial success of numerous tools and packages that allow for grouping and ungrouping of objects, including parametric design tools such as GenerativeComponents by Bentley Inc. and its “generated features”, as well as Rhino Grasshopper and its “custom components”. Further non-parametric examples include “blocks” in AutoCAD; “cells” in MicroStation; “models” or “cells” in Modelica; blocks, scopes, methods, classes, files, and namespaces in C++; container elements such as ‘g’, ‘a’, and ‘symbol’ in SVG; and words, sentences, paragraphs, subsections, sections, and chapters in prose writing.

To conclude, then, the abstraction issues call for many different design interventions, but in particular interventions that improve on modularity of design states as implemented in CAMBRIA 2. Such approaches might include: (1) grouping and ungrouping of items; (2) implementation of a variant of Simon’s hierarchic, nearly-decomposable systems; (3) non-destructively removing and substituting a parametric sub-graph from the parametric model, similarly to product design; (4) enabling the user to create and maintain multiple partitioning frames within a parametric model, as inspired by “part-whole” science approaches, and (5) paying attention to existing approaches to modularity in contemporary parametric CAD tools, programming languages, modelling languages, and markup languages, as evidenced by their successful use and wide adoption by their respective user communities.
The Overall Architecture of Multi-State Tools

While the change from static images and thumbnails to fully editable MDI sub-windows in CAMBRIA 2 does bring some advantages, the experience gained throughout development and research of CAMBRIA 2, as well as my further study of related work, compels me to make an argument that, in fact, it would be advisable to conceptually separate the viewing functionality from editing functionality in a subsequent iteration of CAMBRIA. In other words, it would be helpful to conceptualize the system architecture of multi-state tools as occurring in stages or layers of functionality, with each layer dealing with different main concepts encountered in multi-state work.

In other words, while CAMBRIA 1 lacked many advanced features of CAMBRIA 2, it probably got one thing right: there should, in all probability, exist a level at which the user just views and arranges alternatives, and another (deeper) level where the user edits just the selected alternatives. In addition to my own experience and impression that this might be the case, Shneiderman’s taxonomy of seven fundamental interaction tasks [333] provides additional support for this claim: users of computational tools, in general, first try to obtain an overview of the information, then they zoom into information, filter it, and finally work with details and relationships. The user then reverts to any of the previous levels, as desired.

To be more specific, at level $A$, the user should be able to view, organize, and select one design project to work with. Once a design project has been selected and opened, the user moves to level $B$ where he views, organizes, and selects collections of alternatives. From there, the user can move to level $C$ where he views, organizes, and selects alternatives within collections. The user can then edit just these selected alternatives at level $D$, e.g., by employing any of the parallel editing operations described in this work. Finally, once at level $D$, the user can move further up and work at level $E$ where he can manipulate design alternatives for specific items. From there, the user can revert back to any of the previous levels, as needed or desired. Concisely, the levels would include:

Level $A$. Design projects. View, browse, search, open design projects.

Level $B$. Collections of alternatives. View, browse, search, manipulate collections.

Level $C$. Design alternatives. View, browse, search, manipulate (but not edit) alternatives.

Level $D$. Selected design alternatives. Edit selected alternatives.

Level $E$. Items. Work with items’ own alternatives.

Such an architectural organization of multi-state tools, based on “functional layers”, would bring about a conceptual clarity, as well as result in a beneficial separation of concerns.
Figure 9.24: Visual summary of interaction levels, as the user drills in and out of data structures: (1) projects, (2) collections within the open project, (3) view and select multiple alternatives within multiple collections, (4) edit selected alternatives, and (5) working with selected items within alternatives. Of course, the user should be able to switch levels easily and rapidly.
9.5. CHAPTER SUMMARY

Figure 9.24 depicts this idea schematically. In addition of the fact that any of the Shneiderman's tasks can, and, as a rule, will apply to any major construct used in this work (design projects, collections, alternatives, items), Figure 9.24 can likewise serve as a basic, fundamental “structural pattern” for alternatives-enabled pCAD tools: the user first has to open a design project; view, navigate, and manage collections; work with individual design alternatives within collections; and view, select, manage, and edit individual items within multiple alternatives.

9.5 Chapter Summary

This chapter introduced CAMBRIA 2, the second high-fidelity interactive prototype. Compared to CAMBRIA 1, this prototype presented a number of new features such as a full implementation of one-way data-flow constraints, full project and file management, improved “pass variable” and “pass value” mechanisms, as well as an implementation of a specific “subjunctive snap” interaction technique. Evaluation of CAMBRIA 2, based on the cognitive dimensions framework by Green et al [139, 138, 41, 140], provided additional findings and details, to be integrated with all the bodies of knowledge generated in previous phases of research.
Chapter 10

Phase E: Consolidation

While the cognitive activities of reflection, introspection, and synthesis of higher-level research artifacts have been on-going throughout the duration of this research, they began in earnest upon the completion of all the preceding phases of research that were predominantly concerned with the production of the research-through-design artifacts, such as design requirements, GUI mockups, and interactive prototypes. Hence the name of this phase, consolidation, suggesting the activity of integrating research results obtained so far, into the following higher-order cognitive artifacts:

1. A classification of interaction primitives for alternatives-enabled pCAD tools (§ 10.1). A classification of goals, tasks, methods, and techniques, grouped into logical categories based on their similarities, mostly building on existing taxonomies described in Chapter 5, Section 5.1.1, as well as on the experience that I acquired by conducting this research.

2. Design guidelines (§ 10.2). A set of interaction design guidelines for specifying alternatives-enabled pCAD computational tools, with special emphasis on the rich interactivity nowadays commonly expected of such tools.

3. Design principles (§ 10.3). Having a set of design guidelines in place, from them I then finally derive what is arguably the highest-order body of knowledge produced in this thesis, i.e., the set of high-level design principles for specifying and evaluating alternatives-enabled pCAD computational tools. While design principles, just like design guidelines, aid HCI designers in specifying and heuristically evaluating interaction designs, they are considered to be of a more general, universal, and comprehensive nature.
10.1 Classifying Interaction Primitives for Multi-State Systems

If we consider that both design guidelines and design principles represent some of the highest-level and most sophisticated bodies of knowledge produced in HCI design\(^1\), and which can be synthesized only after a researcher-designer has obtained significant personal experience building, specifying, and studying tools and systems of interest, how does one actually go about synthesizing said bodies of knowledge?

Reminding the reader about the research approach that I adopted for this thesis\(^2\), HCI design guidelines are usually derived from “experimental results, experience with existing interfaces, and knowledgeable guesswork”[333, p. 40]. Yet, that does not tell us much. For instance, how can one assess that any given set of design guidelines is reasonably complete?

Since the literature tells us little about how to achieve the completeness, or reasonable completeness of any set of design guidelines, and in a quest to approach this matter in a structured and systematic manner, I decided to first develop a support base upon which I can scaffold the sought-for set of design guidelines. Thus, I first set out to find a reasonably complete characterization (or characterizations) of the “interaction design space” in the most general and abstract sense, i.e., for general computational tools. Such characterizations are readily provided in the form of various taxonomies, classifications, and typologies of interaction primitives, such as e.g., [333, 311, 49] and which I described in Chapter 5, Section 5.1.1. By studying existing classifications and reflecting on all the work completed so far in this thesis, I then synthesize and present (in this section) my own classification of interaction primitives for multi-state pCAD tools.

Armed with such a classification, and continuing with the metaphor of interaction design “spaces”, such spaces can be regarded as being spanned by a corresponding set of “basis vectors”, or “bases”, that consist of general interaction methods present in any reasonably complete computational tool. For instance, almost all computational tools should enable the user to search for data objects; to navigate the underlying data model; to be able to edit underlying data structures; and so on. Thus, the set \{search, navigate, edit, ...\} induced from the classification of interaction primitives, would then constitute such a “base” spanning the aforementioned, abstract “interaction design spaces” of any reasonably complete, general computational tools.

In the following, I will now propose, based on my experience developing the design artifacts throughout this research as well as performing the overall research activities, a model (or perspective) of tasks and methods that can be applied to any computational tool in general and multi-state tools in particular. In order to do this, I will also borrow heavily from existing models of designing and interaction design, foremostly from the Norman’s seven-step interaction model [273, 271],

\(^1\)As shown in Figure 3.5, design guidelines and design principles are located at the very top of the HCI design knowledge pyramid.

\(^2\)See Section 3.2.
Schön’s “reflective practice” model of designing [323], as well as classifications of interaction primitives by Brehmer and Munzner [49] and Roth [311].

**10.1.1 High-Level Tasks**

While Brehmer and Munzner [49] provide a beneficial and well-thought-out separation of abstract interactive visualization tasks into the three “why”, “what”, and “how” categories (see Section 5.1.1.1 on page 87), they do not provide a “missing link” between the two: the moment when the user “reflects” on his next move. Thus, at the highest level, and in concordance with both the Schön’s model of designing as well as with Norman’s seven-step action cycle, I maintain that the designer basically performs the following three high-level tasks while interacting with a multi-state computational tool:

1. **Perceive.** Perceive the design situation via representations offered by the tool. The act of perceiving is followed by the activities of:
   
   (a) **Choose a working subset.** This task delimits and determines the current “working” subset of information gleaned from perceiving, and is sometimes also called *Search* task. The user can delimit a subset of information by performing any of the following sub-tasks: *Lookup, Locate, Browse, Explore, Preview*, and *Select*.

   (b) **Examine working subset.** Once the user has chosen a working subset, he can proceed with examining it, by performing subtasks *Compare, Arrange, and Detail-on-Demand*.

   Note that the sequence of *Perceive* → *Choose subset* → *Examine subset* can be interpreted as the “analyze design move” within Schön’s model or as the “consume” task by Brehmer and Munzner (see Figure 10.1, upper branch).

2. **Reflect.** Once the user has examined the subset of information, he can reflect on the meaning thus recovered. This step can be considered to be a form of Schön’s “reframe the situation” step during which the user 1) names concepts, and 2) frames concepts i.e., “imposes discipline” on them (Figure 10.1, right).

3. **Act.** Influence the design situation by creating new or modifying existing representations (Schön’s “make the design move”, or Brehmer and Munzner’s “produce” step). Actions might be carried out through any of the following tasks: *Duplicate, Modify, Annotate*, and *Associate* (see Figure 10.1, lower branch).

Figure 10.1 shows a schematic task model that I obtained by “blending” the following three formal models: 1) the Norman’s seven-step action model, 2) Schön’s model of designing, and 3)
the interaction typology by Brehmer and Munzner, with the three highest-level tasks (*Perceive*, *Reflect*, and *Act*) highlighted in light blue colour.

Note that the ultimate *motivation* or *end goals* behind executing these three highest-level tasks, within the context of this work, include but are not limited to 1) creation of suitable designs, 2) casual enjoyment, 3) learning, or 4) evaluation of designs. While Brehmer and Munzner [49] emphasize the *why* the user executes a high-level task, the model in Figure 10.1 is more agnostic, in the sense that I allow for the possibility that the user might have a number of different motivations for interacting and working with a tool. Instead of trying to enumerate all possible motivations that a user might have, I postulate that any motivation (or end goal) will involve all or just some of the three high-level tasks of *Perceive*, *Reflect*, and *Act*. For example, the end goal of creating suitable designs will involve all three tasks, while the end goals of casual enjoyment, learning, and evaluation of designs will in general involve just *Perceive* and *Reflect* tasks, or even just the *Perceive* task (Figure 10.2).

This simple model, in a way, then represents a simplification as well as a slight improvement of the model presented by Brehmer and Munzner. Also please note that, when viewing the whole user+computational tool combination, of the three fundamental tasks *Perceive*, *Reflect*, and *Act*, the *Reflect* takes place entirely within the mind of the user. Tasks *Perceive* and *Act*, on the other hand, are directly connected to the tool under consideration. Accordingly, once the user stops
reflecting, he proceeds with either Perceive or Act.

As for “search” activities, if the user chooses to further perceive (i.e., view, or use any other sense) the representations offered by the tool, he must as the next step choose or obtain a “working set” of the information (or knowledge) for further immediate processing, in any of the following four ways:

1. **Lookup.** The user is said to perform the task of Lookup if he knows both what he is searching for, as well as where to find it. Examples of Lookup include:
   - Searching for an item with a certain name within an alternative, or for a word within a source code file. This type of Lookup is frequently also called Find.
   - Visually scanning for an item in a drawing, while knowing exactly which item, and where to find it.
   - Using various navigation techniques (e.g., panning and zooming) to reach an item in a drawing for which the user knows that it exists at this specific location within the model.

2. **Locate.** The user is said to perform the task of Locate if he knows that is he looking for, but is unsure about its location. For example
   - Searching for a specific item while not knowing the exact alternative containing it, or for an alphanumeric identifier while not exactly knowing which source file within a project contains it. This type of Lookup is frequently also called Find in All Files. In the context of this work, the following options are possible: Find in All Alternatives Within a Collection or Find in All Alternatives Within the Project (i.e., within all collections contained within the project).
   - Using various navigation techniques (e.g., panning and zooming) to reach an item for which he knows that it exists in a drawing, however not being sure where.
Browse. With this task, the user searches for items in a known location while not exactly knowing, or knowing just a part of, what he is searching for. For example, the user might be looking for appropriate rear bicycle wheels components in the “Rear Wheels” section of a repository of pCAD models.

Explore. The user neither knows exactly what he is looking for, nor where to find it.

To these four activities for defining the “working set”, however, I now add the following two that I feel should belong to the set of “choose” methods, that of Preview and Select, which likewise help the user to delineate and define his current working set:

Preview. Using a set $S$ of existing design states, the user rapidly generates a number of temporary design states from the set $S$ for the sake of quick “what-if” experimentation. The user can now add some of these preview states, if so desired, to his current working set.

Select. The user selects a subset of elements within the working set, for further editing and manipulation.

Again, once the user has chosen a subset of information of interest, using any of the tasks Lookup, Locate, Browse, Explore or Select, listed above, he or she can proceed with examining said subset of information. For this task, the user can execute tasks Details-on-Demand, Compare, and Arrange:

(1) Demand-Details. A task whose purpose is to obtain additional, in-depth information about the working set, thus facilitating the user’s examination and apprehension of the same.

(2) Compare. A task whose end goal it to help the user to identify both the similarities, as well as differences, among a set of objects contained in the working set.

(3) Arrange. A task whose purpose is to lay out (i.e., present) elements in the working set in order to discover their relative “qualities”, “kinship” or “similarity” according to some set of criteria. For example, the elements might be ranked according to some objective metric, or or clustered into proximity-based groups by the user according to some subjective criteria (e.g., aesthetic appeal).

Summary of High-Level Tasks

Figure 10.3 shows the hierarchy of high-level tasks, expressing in the pseudo-“operator” form using uppercase letters.
(1) **PERCEIVE:**

(a) **CHOOSE-WORKING-SUBSET:**
   - CHOOSE-WORKING-SUBSET-MAINTAIN (i.e., maintain the current subset)
   - CHOOSE-WORKING-SUBSET-SEARCH (includes *Lookup, Locate, Browse, and Explore*)
   - CHOOSE-WORKING-SUBSET-PREVIEW
   - CHOOSE-WORKING-SUBSET-SELECT

(b) **EXAMINE-WORKING-SUBSET:**
   - EXAMINE-WORKING-SUBSET-COMPARE
   - EXAMINE-WORKING-SUBSET-ARRANGE
   - EXAMINE-WORKING-SUBSET-DEMAND-DETAILS

(2) **REFLECT:**

(a) REFLECT-NAME-THINGS

(b) REFLECT-FRAME-THINGS

(3) **ACT:**

(a) ACT-DUPLICATE-WORKING-SET

(b) ACT-MODIFY-WORKING-SET

(c) ACT-ANNOTATE-WORKING-SET

(d) ACT-ASSOCIATE-ELEMENTS-OF-WORKING-SET

---

**Figure 10.3:** Classification of high-level tasks for working with multi-state pCAD tools.

Note that the classification of high-level tasks shown in Figure 10.3, while presented within the context of multi-state tools, can also be applied to any general computational tool or cognitive artifact.

### 10.1.2 Interaction Methods

As the second component of the two-part classification of interaction primitives, in addition to high-level tasks described in Section 10.1.1, I now proceed with proposing a sub-classification of *methods* for interactive visualization of multiple design alternatives. Methods can be thought
of as families of affiliated interactive visualization *functions* or *features* which enable the user to accomplish any of the high-level tasks described in § 10.1.1. In the literature, alternative names for *methods* include *techniques* [49], *interactive functionality* [311], *operators* [71, 397], or *manipulations* [54].

As the first salient point, methods are orthogonal to high-level tasks described in previous section. A task can be performed by a number of methods; for example, the task of *Locate* can be completed by applying any of the navigation methods, such as *Scroll*, *Pan*, *Zoom*, and *Find by keyword*. Therefore, the relationship cardinality for any given high-level task and the set of applicable methods, is in general $1 : N$.

On the flip side, a given method can likewise be applied to a number of high-level tasks. For example, *Pan* navigation methods can applied to any of the high-level tasks *Lookup*, *Locate*, *Browse* and *Explore*. Therefore, the relationship cardinality for any given method and high-level tasks, is in general also $1 : M$.

Based on literature review (see Chapter 5, Section 5.1) as well as on the experience that I acquired in this work, I now propose a classification of interaction methods for multi-state pCAD tools (Figure 10.4).

<table>
<thead>
<tr>
<th>Classification of interaction methods for multi-state pCAD systems:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) <strong>CHOOSE.</strong> Methods for demarcating (choosing, selecting, isolating) or augmenting the current “working set” i.e., the scope of information that is currently under consideration by the user, and which may or may not be modified further.</td>
</tr>
<tr>
<td>(a) <strong>CHOOSE-SEARCH.</strong> This family of methods allows the user to search for alternatives in corresponding depositories or databases, using either search terms (i.e., keywords), images (i.e., image-based search), or whole alternatives.</td>
</tr>
<tr>
<td>i. <strong>CHOOSE-SEARCH-KEYWORD.</strong> For example, Fujima <em>et al</em> present the C3W prototype tool [123, 124] for ‘subjunctive” WWW searching, i.e., searches done in parallel. In [224], Lunzer describes subjunctive interfaces for exploring online resources in parallel.</td>
</tr>
<tr>
<td>ii. <strong>CHOOSE-SEARCH-IMAGE.</strong> Similar to <strong>CHOOSE-SEARCH-KEYWORD</strong>, only the user uses an image of the alternative to conduct search.</td>
</tr>
<tr>
<td>iii. <strong>CHOOSE-SEARCH-MODEL.</strong> Similar to keyword- or image-based search, only the user utilizes a computational model to conduct search.</td>
</tr>
</tbody>
</table>
| (b) **CHOOSE-PREVIEW.** Methods for (in most cases, quick) generation of “temporary” alternatives for the purposes of “what-if”, near-term experimentation. In practical terms, a tool might take the set of current alternatives, and generate $n$ “temporary
alternatives” (perhaps in-memory), thus enabling the user to conduct quick what-if experiments and maybe choose some for further work, thus augmenting his current working set.

(c) CHOOSE-SELECT. Choosing and visually demarcating one or more items within current alternatives, for subsequent processing. A good alternative name for these methods would be CHOOSE-VISUAL-SCAN, because in most cases of selection the user scans presented information visually, in order to choose the appropriate set. In this thesis, CAMBRIA offers a mechanism for parallel select whereby upon selecting an item in one alternative, all the linked items in other active alternatives will be selected in parallel as well.

(2) EXPRESS. Methods for rendering a data object i.e., displaying it visually. For example, in CAMBRIA 1, a collection can be expressed as a 2D scene, a row of data, or as a thumbnail.

(a) EXPRESS-IMAGE. Object is expressed as an image or thumbnail.

(b) EXPRESS-IMAGE+NAME. Image and name.

(c) EXPRESS-DATA-ROW. As a row of alphanumeric data.

(d) EXPRESS-ADD-VIEWLET. Adds a new view (viewlet) for the object.

(e) EXPRESS-REMOVE-VIEWLET. Removes a view (viewlet) for the object.

(3) ARRANGE. Methods for spatially organizing a set of data objects for viewing include but are not limited to:

(a) ARRANGE-AUTOMATIC. A family of methods that arranges objects (alternative, projects, items, etc.) automatically.

   i. ARRANGE-AUTOMATIC-RANK. According to rank (relative positions).

   ii. ARRANGE-AUTOMATIC-GRID. Juxtaposition in a rectilinear grid.

   iii. ARRANGE-AUTOMATIC-LAYERS. Superposition i.e., on top of each other.

   iv. ARRANGE-AUTOMATIC-SLIDESHOW. A sequence of maximized views.

   v. ARRANGE-AUTOMATIC-DATA-TABLE. A sequence of alphanumeric rows.

   vi. ARRANGE-AUTOMATIC-DISTANCE-METRIC. Arranged due to some objective distance metric [239, 240].

   vii. ARRANGE-AUTOMATIC-DESIGN-HISTORY. As a function of their derivation order (lineage).
(b) ARRANGE-MANUAL. Includes manual arrangement of objects i.e., using some personal (subjective) criteria.

i. ARRANGE-MANUAL-FREE. The user can adjust positions, sizes, and orientations of views.

ii. ARRANGE-MANUAL-GRID. The user can interactively move an object from one position in the grid, to another. Thus while the user can change positions, he cannot change sizes or orientations of views.

(4) NAVIGATE: Methods that help the user to create a mental map of alternatives, by setting the viewport extent i.e., by defining which part of the alternative is currently visible.

(a) PARALLEL-PAN. Move the center of the viewport to an off-screen portion of the model. This method can be executed in parallel for multiple alternatives.

(b) PARALLEL-ZOOM. Temporal separation between contextual and focused views, performed by changing the scale and/or resolution of the model, so as to magnify/shrink an area. Again, zooming can occur in parallel for multiple alternatives, by the same differential zoom factor.

(c) PARALLEL-OVERVIEW+DETAIL. Spatial separation between contextual and focused views (e.g., a small inset overview, or a thumbnail strip). For multi-state system, an overview window showing zoom extents (viewports) for all active alternatives; also, a thumbnail strip where each thumbnail represents an alternative.

(d) PARALLEL-FOCUS+CONTEXT. Displaying the focused view within the contextual view, e.g., fisheye lens [125, 320] within multiple alternatives. These methods might involve multiple “subjunctive cursor” instances within multiple alternatives.

(5) EXAMINE. Methods that allow the user to examine the current working set, thus contributing to his learning and apprehending the set.

(a) EXAMINE-PARALLEL-DEMAND-DETAILS. Request specific information for an item, e.g., by a separate panel showing additional data/information about the item. This can occur in parallel for multiple alternatives; for example, demanding details about a selected item may invoke a separate view listing details for both the selected item as well as for all the linked items in other active alternatives.

(b) EXAMINE-PARALLEL-FILTER-OUT. Suppress or hide items meeting user-defined conditions; filter out undesired elements of alternatives. Again, can occur in parallel for multiple alternatives, if so indicated and desired by the user.
(c) EXAMINE-PARALLEL-FILTER-IN. Similar to PARALLEL-FILTER-OUT, only it shows just the desired elements (for example, those matching a search keyword) while suppressing or hiding other elements.

(d) EXAMINE-PARALLEL-HIGHLIGHT. Similar to both PARALLEL-FILTER-OUT and PARALLEL-FILTER-IN, only it neither filters out or filters in anything, but highlights the item matching a search keyword.

(e) EXAMINE-PARALLEL-BRUSH. View, edit, and understand what items are linked or associated to each other. Again, can take place in parallel, for multiple alternatives. For example, keeping the mouse pointer over an item in an alternative might highlight all the linked items in all the other active alternatives.

(f) EXAMINE-DIFFERENCE. Determine similarities and differences between multiple objects or alternatives.

(6) EDIT. Methods that allow the user to ACT on the current “design situation” i.e., somehow modify or generate new data belonging to the underlying computational models or states.

(a) EDIT-SELECT. Methods for demarcating the items that will undergo subsequent editing operations. These methods are equivalent to CHOOSE-SELECT methods in the CHOOSE category.

(b) EDIT-CHANGE-STATUS. Methods for managing (setting, unsetting) the “linked” status of an alternative. An alternative is considered “linked”, it is currently receiving modifications from other (parallel) commands. An alternative can be (un)linked by clicking on a “toggle linked” button, by choosing a menu option, or minimizing the alternative’s view. In addition, an alternative can be “soloed” which unlinks all the linked alternatives except this one, thus allowing the user to focus on just the current alternative.

(c) EDIT-MODIFY. A number of methods for modifying states.

   i. EDIT-MODIFY-PROPERTY-VALUE. Generic, standard methods for changing certain properties of items, within the “direct manipulation” paradigm. For example, “change pen colour” technique for changing the pen colour of all the currently selected items, by invoking the standard “colour picker” dialog. Property modification can occur in parallel, for multiple states.

   ii. EDIT-MODIFY-INSERT. Several methods for inserting sub-states, thus modifying (i.e., augmenting) the structure of the design state(s).
A. EDIT-MODIFY-INSERT-SEARCH-RESULT. Insert search result(s) obtained via CHOOSE-SEARCH.

B. EDIT-MODIFY-INSERT-PREVIEW. Insert preview(s) obtained via CHOOSE-PREVIEW.

C. EDIT-MODIFY-INSERT-GENERATE. Insert newly generated objects, via a number of “generate” sub-methods.

iii. EDIT-MODIFY-DELETE. The twin method for EDIT-MODIFY-INSERT, in the sense that it also modifies the structure of design state(s), except by removing sub-states instead of inserting them.

iv. EDIT-MODIFY-GROUP and -UNGROUP. Two methods for grouping and ungrouping items.

v. EDIT-MODIFY-TRANSFER-ITEMS. Standard methods such as copy, cut, paste, as well as some novel, parallelized ones such as “pass items”, and “pass values”.

A. EDIT-MODIFY-TRANSFER-ITEMS-COPY. Places all the selected items into the temporary memory area called “clipboard”.

B. EDIT-MODIFY-TRANSFER-ITEMS-PASTE. Copies items currently on clipboard into the current alternative.

C. EDIT-MODIFY-TRANSFER-ITEMS-CUT. Places all the selected items into the “clipboard”, and if user now executes EDIT-MODIFY-TRANSFER-ITEMS-PASTE, copies items currently on clipboard into the alternative and deletes them from the originating alternative.

D. EDIT-MODIFY-TRANSFER-ITEMS-PASS-ITEMS. A type of generalized copy-paste that works with multiple alternatives in parallel.

E. EDIT-MODIFY-TRANSFER-ITEMS-PASS-PROPERTIES. Methods that propagate property values from a source item, to all the linked (associated) items.

vi. EDIT-MODIFY-SUBSTITUTE. Methods for replacing an item in-place, for example as shown in Figure 5.12 on page 106.

vii. EDIT-MODIFY-SNAP. A technique for automatically attaching a point onto other geometric objects, e.g., onto lines or other points.

viii. EDIT-MODIFY-ASSOCIATE. Methods for adding, removing, and modifying “links” or “associations” among items. For example, an item could be associated with several other items for the “pen colour” property; subsequent “pass property” commands will then propagate the current pen colour to all the associated items.
ix. EDIT-MODIFY-UNDO and EDIT-MODIFY-REDO. Methods for recovering from inadvertent errors, or for conducting quick “what-if” experiments.

(d) EDIT-ANNOTATE. Methods for adding data that are not part of the design state stricto sensu, but help the user to understand or navigate the state or states.

**Figure 10.4:** Classification of interaction methods for multi-state pCAD tools.

Note that any of the methods listed in Figure 10.4 might apply to different concepts in the conceptual model. For instance, the user can apply CHOOSE-SEARCH for any of the main data objects: projects, collections, alternatives, and items within alternatives.

### 10.2 Design Guidelines for Alternatives-Enabled pCAD Tools

Having a *classification* or *taxonomy* of interaction primitives in place (Section 10.1), I can now use it to “scaffold” a set of *system design guidelines*, which are high-level suggestions or recommendations that can help interactive system designers to approach, frame, and solve complex system design problems, as well as heuristically evaluate proposed design solutions or candidates [333, p. 40]. A set of guidelines always applies to a certain range of design problems, and to a specific target domain. Design guidelines therefore [268, p. 374]:

1. Help enhance designs,
2. Raise awareness of concepts,
3. Assist in design choices,
4. Offer problem-solving strategies, and
5. Provide yet another, “heuristic”\(^3\) means to evaluate user interfaces.

For instance, in the field of software design, Henninger [154] writes about “usability guidelines” that software developers should consult early and often in their development process, and which are a widely accepted method of bringing the cumulative knowledge to the software development process:

*While it is generally accepted that guidelines cannot replace the golden rules of interface design—user involvement, user feedback from early prototypes and iterative development—guidelines can play a role in improving the quality of the iterative steps, leading to an improvement in quality and reduction (but not elimination) of the number of iterations involved in the design-evaluate-redesign cycle of HCI development.*

\(^3\)Or broad rules of thumb [269].
As an example of HCI design guidelines, Wickens and Hollands [403][333, p. 60] present the following set of eight guidelines for getting the user's attention in the context of user interfaces:

1. Use two levels (low, high) of intensity only, with very limited use of the high level.
2. For marking an item, do any of the following: underline it, enclose it in a box, point to it with an arrow, or use an indicator such as bullet, "+" sign, or "X" sign.
3. Use up to four sizes of items on screen (larger size attracts more attention).
4. Use up to three typefaces (font styles) only.
5. Use inverse video (i.e., inverse colouring or negative) where appropriate.
6. Only use blinking elements (2–4 Hz) with great care.
7. Use up to four colours only, with fifth and more colours used only occasionally.
8. Use soft tones for positive and harsh tones for negative feedback.

**Figure 10.5:** Design guidelines for attracting user’s attention [403][333, p. 60].

As can be observed from Figure 10.5, that particular set of design guidelines is concerned with rather low-level GUI implementation details, such as the number of colours used, how many different typefaces should be used, or that an item should be delimited by a box.

I now build upon the proposed classification of interaction methods described in Section (Figure 10.4), and proceed to detail design guidelines for alternatives-enabled pCAD tools, grouped into the corresponding categories induced by the following categories of interaction methods:

1. Choose
2. Express
3. Arrange
4. Navigate
5. Examine
6. Edit

Figure 10.6 below contains a summary of design guidelines; I will then detail each guideline in subsequent sections.

**Design guidelines for multi-state pCAD systems:**

**Choose:**
• DG-1. Provide powerful search capabilities, e.g by means of search keywords, images, or models (p. 232).

• DG-2. Provide powerful preview capabilities, for rapid “what-if” experimenting (p. 234).

• DG-3. Provide flexible selection mechanisms (p. 236).

Express:

• DG-4. Support various, multiple ways to express objects visually (p. 237).

Arrange:

• DG-5. Support a variety of automatic methods for arranging multiple objects for viewing (p. 242).

• DG-6. Allow the user to arrange objects manually, in order to create his own meaning (p. 250).

Navigate:

• DG-7. Provide parallelized panning methods (p. 251).


• DG-9. Provide various overview+detail navigation methods, such as thumbnail strips or overview insets (p. 254).

• DG-10. Provide various focus+context navigation methods, such as subjunctive cursors for parallel fisheye lens techniques (p. 256).

Examine:

• DG-11. Provide powerful details-on-demand capabilities, so that the user can obtain as much information on items of interest as is possible (p. 257).

• DG-12. Provide filter-in and filter-out capabilities, so that the user can obtain as much information on items of interest as is possible (p. 258).

• DG-13. Provide capabilities that enable the user to highlight items of interest within a certain scope (p. 259).

• DG-14. Enable brushing for associated (linked) objects (p. 260).

Edit:

• DG-16. Enable the user to modify the editing status of an object, for example from "linked" to "unlinked", and other types of status as needed (p. 264).

• DG-17. Enable the user to easily and quickly modify property values, e.g., through direct manipulation techniques, contextual menus, and property editors (p. 267).

• DG-18. Support the easy and quick insertion of search results into the corresponding scope (p. 268).

• DG-19. Support the easy and quick insertion of previews into the corresponding scope (p. 268).

• DG-20. Support multiple, flexible and powerful methods for generating new objects of interest (p. 268).

• DG-21. Enable parallel delete of associated items (p. 270).

• DG-22. Allow the user to group and ungroup items (p. 271).

• DG-23. Enable capabilities to move or transfer items from one container object, to another, such as copy, paste, cut, and parallelized versions thereof (p. 271).

• DG-24. Enable the user to substitute (replace) objects of interest (p. 272).

• DG-25. Enable the parallel snapping technique, whereby the points attach automatically to other objects based on their names, not geometric positions (p. 275).

• DG-26. Allow the user to associate (link) objects in multiple, flexible ways (p. 276).

• DG-27. Allow undo and redo capabilities, with separate undo stacks for each alternative (p. 277).

• DG-28. Offer rich annotation capabilities (p. 278).

Figure 10.6: Summary of design guidelines for multi-state pCAD systems.

In the following, for each of the design guidelines listed in Figure 10.6, I will 1) describe design rationale, and 2) provide examples of typical (emblematic) multi-state functionality, using the following format:
Additionally, I will describe each emblematic multi-state functionality by including either (1) screenshots of prototypes developed in this thesis (CAMBRIA 1 and 2, as well as early prototypes), or (2) as a wireframe GUI mockup suggesting a potential interaction technique.

10.2.1 CHOOSE Methods

CHOOSE methods help the user delimit or augment his current “working set”. In the literature, such techniques are commonly and generally called “search” techniques. However, I maintain that the highest-level PERCEIVE task is always first followed by the user's definition and demarcation of the “working set” which will be examined and manipulated further (even if that is the current working set; in that case, CHOOSE is an identity/no-op task i.e., the user continues to work with the current working set). Thus, in a way, “Choose” can be considered to have a slightly wider meaning of “Search”, as understood in the conventional sense.

10.2.1.1 Search (CHOOSE-SEARCH)

DG-1. Provide powerful search capabilities, e.g., by means of search keywords, images, or models.

These methods, CHOOSE-SEARCH methods, allow the user to search for and possibly find pre-existing objects of interest in external or internal depositories, and then possibly include a subset of search results into his current working set.

As indicated in the classification of methods (Figure 10.4), various ways to conduct searches exist, for example by entering search keywords, by submitting a representative image (image-based search), or even by submitting a whole computational model to the search engine. A full review of different search methods is beyond the scope of this work; for up-to-date surveys the reader is pointed to sources on information retrieval (IR), for example [344, 142, 254].

Design rationale. Justification for CHOOSE-SEARCH includes the ability to select and consequently work with pre-generated information artifacts (or, within the context of this work, pre-generated design states). By doing this, the user eschews the need to generate such artifacts beforehand. In many cases, such artifacts are highly specialized, or require significant effort and resources to produce. The CHOOSE-SEARCH methods, thus, enable the user to re-use such pre-made artifacts stored elsewhere, perhaps in a repository of pCAD models.
As a simple yet illustrative example, if designing an electric bicycle, the user might decide to simply insert a pre-existing model of the small electric motor into his or her current design project, possibly for a fee, thus precluding the need to develop such a relatively complex artifact by himself, and therefore accelerate the completion of the design project.

Within this dissertation, no CHOOSE-SEARCH methods have been implemented with any of the prototypes developed. However, one can easily imagine a corresponding repository of 2D parametric drawings similar to the ones used in CAMBRIA, as well as implementing the networking functionality that stores items and design alternatives into an online repository.

**Emblematic multi-state functionality.** As an example of how one might proceed with multi-state search, Lunzer, Fujima et al present an approach to parallel WWW searches [223, 123, 124]. Following their lead, I provide a mockup of how one might conduct parallel, or “subjunctive” searches.

![Figure 10.7: An example of “parallel” or “subjunctive” search functionality.](image)

Figure 10.7 illustrates a possible multi-state search functionality, whereby the interface consists of the following three main parts (top, middle, and bottom parts):

1. **Search input field.** Top part, where the user inputs search terms or keywords into a single input field, but which will be passed into different, multiple searches.

2. **Scope switch.** Middle part, which contains multiple instances of a special, replicable “switch” cell that defines the scope of the search. The user can duplicate or remove an instance of the switch cell at will in this part, as many times as he or she desires; each act of replicating this switch cell will then:

   a. Create a new search process,
(b) Pass the search term into this process, and

c) Automatically create a new GUI view (bottom part) that will be populated with search results from this new search process.

(3) Search results per scope. Bottom part, which contains views which in turn contain search results for each search process.

Note that each instance of the switch cell and its associated search results view are colour-coded, for easier discernment of which set of search results belongs to which switch cell. Also note that conventional, single-state searches do not feature the middle part (the scope switch), since searches in single-state tools always produce just a single set of search results.

10.2.1.2 Preview (CHOOSE-PREVIEW)

**DG-2. Provide powerful preview capabilities, for rapid “what-if” experimenting.**

The category includes methods that allow the user to perform open-ended, “creative”, or “what-if” operations. In most cases, the tool or system generates a set of computationally inexpensive (maybe of slightly degraded quality) “preview” design states obtained by the application of corresponding transformation operators, “effects”, or “filters”. The user can then view and analyze this set of rapidly obtained preview states, and possibly decide to include them into his current working set.

*Design rationale.* CHOOSE-PREVIEW methods support open-ended, rapid and “what-if” style experimentation in a contingent fashion [369, 368, 370, 367]. For this capability, approaches reported in the literature include e.g., *Side Views* and *Parameter Spectrums* [369, 368]. Experimental evidence shows that, using preview capabilities, users explore design problems more broadly, and succeed in finding unexpected but still viable design solutions [367]. Again, while that work has been completed in the context of raster image editors, one might conjecture that the authors’ conclusions would probably hold in the domain of 2D pCAD drawings as well, considering that in both cases the previews are generated by the tool.

If one compares CHOOSE-PREVIEW methods with CHOOSE-SEARCH methods, one can argue that CHOOSE-PREVIEW methods score more highly on the “locality” and “immediateness” measures. In other words, CHOOSE-PREVIEW allows the user to rapidly and relatively cheaply (in terms of time and space) generate multiple design alternatives or variants that differ only slightly from the original design state(s), while CHOOSE-SEARCH might find design state(s) that differ more significantly from the design states currently under consideration by the user. Thus, in a way, CHOOSE-PREVIEW generates more “immediate” design alternatives. Another way to put this is that CHOOSE-PREVIEW is a type of CHOOSE-SEARCH, but for close, “proximal” searches.
In addition, temporary states generated through CHOOSE-PREVIEW methods are generally not
stored in external repositories of design states, but simply discarded when the user has reviewed
them and decided not to include them.

**Emblematic multi-state functionality.** Within the bounds of the work described in this thesis,
emblematic preview features include (a) the “generate alternatives” command in the “2D Font
Designer” early prototype (see § 6.2.3.2 on p. 122), and (2) the “random perturbation of selected
items” in CAMBRIA 1 (see item 5 on 149). The first feature, while not a part of a formal study,
obtained favourable informal comments by the individuals testing the tool; the second feature was
likewise found “interesting” [P1 41:16].

![Interactive prototype for 2D parametric design (“Font Designer”), with 15 preview states.](image)

**Figure 10.8:** Interactive prototype for 2D parametric design (“Font Designer”), with 15 preview states.

Figure 10.8 (top third of the screen) presents this early prototype, featuring an approach to
generating randomly-perturbed instances of the master “S” parametric model (see § 6.2.3.2 on
page 122). Informal feedback obtained from faculty and graduate students on such randomly
perturbed set of 15 design alternatives was overwhelmingly positive and deemed interesting. For
large, full-scale parametric models, one could imagine a tool which automatically down-samples
a full model (consisting of, say, 100,000 nodes) to say 1,000 nodes only, and reuse this simplified
parametric for rapid generation of multiple temporary, contingent low-density models created in
the “what-if” manner, just for the sake of preview and visual comparison. If the user decides to
include a preview state, the tool would then apply the same transformation on the full model, and
insert it into the user’s working set.
Figure 10.9 shows another possible approach, a preview of the effects of a “filter” that affects the geometry of items. The *Before* view shows the collection before applying the effect, and *After* view the collection after the effect has been applied to all the selected items in all visible alternatives. In this particular example, the “twirl” effect has been applied to the geometry of several item instances selected in parallel within three active alternatives. The user can increase or decrease the total exerted effect by manipulating the “Effect setting” slider control within the Preview window.

### 10.2.1.3 Select (CHOOSE-SELECT)

**DG-3. Provide flexible selection mechanisms.**

Selection refers to the action of choosing one or more parts of alternatives (i.e., items) that might or might not receive any of a number of possible subsequent actions or manipulation operations. It can thus, in a way, be regarded as the precursor action to all the other editing methods (to be described in § 10.2.6 on page 263).

**Design rationale.** While CHOOSE-SEARCH can intuitively be regarded as “long search” methods, CHOOSE-PREVIEW methods as “proximal search” methods, CHOOSE-SELECT methods could, on the other hand, probably be best regarded as “ultra-proximal search” methods. In other words, instead of searching external repositories of pre-generated states, or searching the “near” design space of preview states, the user searches directly for items within the current design state(s). In many cases, this CHOOSE-SELECT searching is conducted by the user himself, i.e., by directly employing his visual system: the user *visually scans* the current design state(s), and then demarcates a set of items of interest by interacting with them. Subsequently, the user
edits the selection, thus moving his traversal of the design space forward.

**Emblematic multi-state functionality.** Both CAMBRIA 1 and 2, as well as some early prototypes, feature single-state or multi-state select capabilities.

![Multi-state selection in CAMBRIA 2](image)

Figure 10.10: Multi-state selection of multiple (linked) items in CAMBRIA 2.

Figure 10.10 shows an example of the end result of using the multi-state rubberband technique. The act of selecting items in one alternative will, if so desired by the user, automatically select other instances of those same items in the other active alternatives. From my own experience acquired by implementing and evaluating all the research prototypes described in this work, parallel selection should be considered a fundamental, essential part of any interactive multi-state tool.

### 10.2.2 EXPRESS Methods

**DG-4. Support various, multiple ways to express objects visually.**

This interaction technique, or system capability, changes how a data object is displayed on screen, while preserving its visual isomorph\(^4\). Ways to visually re-express an object (i.e., a project, collection, or alternative) can be classified into the following three main categories: graphic, alphanumeric, and hybrid. Graphic expressions can be further roughly classified into raster graphics (a representation that can be magnified only up to a certain degree, before the loss of visual quality becomes apparent), vector graphics (a representation that can be magnified without any loss of quality), and glyphs (abstracted visualization objects which display several values at once, for an alternative).

\(^4\)See the definition of visual isomorph in Section 6.1.
**Design rationale.** Different visual representations, even though they might be informationally equivalent, have different effects on the cognitive processes of visual search, scanning, recognition, and inference [206, 32]. For instance, Larkin and Simon famously inquired into “why a diagram is (sometimes) worth ten thousand words”[206]. To support multi-faceted, complex, and “messy” processes of designing (and indeed, of any type of cognitive work), a computational tool should therefore as a rule afford many different object representations; different representations will as a rule have their strengths and their weaknesses distributed across different activity areas. Hence, multiple representations will most likely complement each other, and thus lead to a more effective and powerful tool or system.

For instance, *thumbnails*, or scaled-down images, afford the following benefits to the user: (1) visual recognition of past states [232, 152], (2) proxy management of data objects represented by them, and (3) aiding with comparing multiple (similar) objects, i.e., objects with similar *structures* repeated across all such objects, see “small multiples” [380, 381]. As for the most effective size of thumbnails, a controlled empirical study of web page thumbnails by Kaasten *et al* [181] demonstrates a tradeoff between thumbnail size and recognition rate: sizes of 80 and 150 pixels result in 50% and 90% recognition rate, respectively. Alphanumerical representations, on the other hand, allow the user to pack many data objects in relatively small space (a table with rows and columns), thus achieving high information density, or information/space ratio.

As for design tradeoffs, for instance the clarity of thumbnails is inversely proportional to their size; likewise, speed of access is directly proportional to their size [72]. Smaller thumbnails, while allowing more rapid access to distant thumbnails, result in an unclear overview of the represented visual object i.e., slower comprehension or higher detection error rates. Bigger thumbnails, on the other hand, while providing a clearer overview of a thumbnail, necessarily result in more scrolling by the user to reach distant thumbnails. One way to address this tradeoff is to allow the user to interactively change the size of thumbnails, thus achieving either faster access speeds or faster comprehension rates, as per current need, or the immediate goal at hand.

**Emblematic multi-state functionality.** Figures 10.11 and 10.12 show some ways to express collections and alternatives, respectively.
10.2. DESIGN GUIDELINES FOR ALTERNATIVES-ENABLED PCAD TOOLS

Figure 10.11 demonstrates four different ways to express a collection graphically. From left to right: (1) as a “pile” of alternatives contained within the collection, with different sizes, positions, and orientations for each alternative; (2) as a representative alternative for the collection; (3) as a snapshot of an arrangement of the alternatives contained within the collection; (4) as an abstract mathematical set (Venn diagram), containing elements (i.e., alternatives). Undoubtedly, many other ways to express collections graphically exist, limited only by the interaction designer-researcher’s imagination.

Figure 10.12: Expressing an alternative (purely) graphically.

Figure 10.12, likewise, demonstrates three different ways to graphically express an alternative. From left to right: (1) as a (small) image (thumbnail); (2) as a glyph; (3) as simply a 2D point on a plane, thus perhaps indicating its position in some two-dimensional criteria space.

Figure 10.13 shows three different potential ways to express an alternative as a text. From left to right: (1) as name only; (2) as name + textual information; (3) as a row of data.

Figure 10.13: Expressing an alternative alphanumerically.

Similar text-only expression types are, of course, available and possible for collections, or any other constructs of interest.

Figure 10.14 demonstrates five (among many other possible permutations) ways to express an alternative in a hybrid way. From left to right: (1) as a thumbnail + name; (2) as a glyph + name; (3) as a 2D point on a plane + name; (4) as thumbnail + text.

Figure 10.14: Some ways to express an alternative using a hybrid method.

Again, similar hybrid expression types are of course possible for collections, projects, or any other pertinent objects; other ways to express object in a hybrid way are likewise limited only by the interaction designer’s creativity.
10.2.3 ARRANGE Methods

Narrowly related to the EXPRESS family of methods (§ 10.2.2), ARRANGE methods enable the user to view many objects at once, arranged into different configurations or arrangements. A set of alternatives can be arranged either automatically, or manually. Some possibilities include:

(1) Arrange automatically.
   
   (a) Rank.
      i. Rectilinear grid (juxtaposition).
      ii. Layers (superposition).
      iii. Slideshow.
      iv. Table with rows and columns (sortable grid of property values).
      v. Lifestream (coverflow).
   
   (b) By distance metric.
   
   (c) As design history (i.e., lineage graph).
   
   (d) Random arrangement.

(2) Arrange manually on a 2D canvas.

   (a) Free arrangement.
   
   (b) Rectilinear grid (juxtaposition).

Figure 10.15 shows an example of eight differently-arranged collections of design alternatives, within a hypothetical design project.

Design rationale. Just as was the case with EXPRESS methods, different representations have different effects on the cognitive processes of visual search, scanning, recognition, and inference. To support the complex process of designing, a computational tool should therefore as a rule afford many different (visual, i.e., geometric) arrangements of data objects on screen.

Next, as I already described in some detail in Section 4.1.7 (p. 76), being able to view multiple objects at once can enable and facilitate the cognitive process of visual comparison which, in turn, can reveal common structure and combine partial structures and thus promote transfer.

As for other design rationale for ARRANGE methods, in the literature related to multi-state interactive visualization, Lunzer [220] recommends that effective tools should offer multiple, powerful, and flexible ways to view and compare multiple states at once. In [226] Lunzer and Hornbæk recommend “multiplexing”both input and output widgets, in order to prevent excessive information retrieval of past states, and in order to facilitate exploration of different combinations of multiple
parameters. In the empirical evaluation of subjunctive interfaces featuring parallel display and control of alternative scenarios and in comparison with conventional (non-subjunctive) interfaces [227, 230], it was found that users are more satisfied with subjunctive interfaces, report lower workload, work faster, and rely less on writing down notes i.e., work less outside the interface itself.

In CAMBRIA 1 expert study feedback, being able to view and visually compare many alternatives at once was “probably the most interesting and useful feature” of the prototype [P1 47:01] [P1 48:16] [P1 1:12:30] [P2 52:31] [P3 1:15:21] [P3 1:27:37] [P5 39:25] [P6 1:10:01], [P6 1:15:59], [P6 1:17:07]. Further, being able to “look at variants that quickly and easily” [P3 1:11:01] further strengthens the evidence and design rationale for arrange methods.

As for potential weaknesses and design trade-offs, due to the limitations of human cognition, users frequently use external representations in order to offload information [321]. This might naturally lead to the detrimental effect of attention switching, which in turn can potentially lead to the interruption [246] of psychological “flow” [89]. Further, in some instances, multi-view interfaces might lead to deteriorated task completion times [163, 31], due to increased motor effort (eye and head movements), as well as attention switching issues. Accordingly, in CAMBRIA 1 expert study, one participant would probably not view many variants of documents i.e., (in prose and code

Figure 10.15: A mockup of eight different ways to arrange design alternatives.
writing), and would focus on only one version of the document at a time [P1 48:16]. (The same participant would, however, still want to view multiple documents such as photos and graphic designs at once.) In other words, the interaction designer of multi-state systems should pay adequate attention to this design tradeoff and related issues, and perhaps adapt the total number of design alternatives shown concurrently to the task at hand.

10.2.3.1 Automatic ARRANGE Methods

DG-5. Support a variety of automatic methods for arranging multiple objects for viewing.

These methods, ARRANGE-AUTOMATIC methods, allow the user to automatically (computationally) arrange a set of objects according to some pre-defined objective criteria e.g., date created, date last modified, user rating, and so on. The following types can be distinguished in this category:

1. Rank. This category contains methods which arrange objects in a sequence, while preserving their relative positions according to some set of objective criteria. Some of the possible layouts include:
   
   (a) Rectilinear grid (juxtaposition).
   
   (b) Layers (superposition).
   
   (c) Slideshow.
   
   (d) Table with rows and columns (sortable grid of property values).
   
   (e) Lifestream (coverflow).

2. By distance metric.

3. As design history (i.e., lineage graph).

4. Random arrangement.

Rectilinear Grid (Juxtaposition) Arrangement

In this view type, alternatives are arranged into a regular, rectilinear grid, and ordered from top to bottom and from left to right. Thus, this arrangement type is based on the principle of juxtaposition, or highly-regular side-by-side placement\(^5\).

---

\(^5\)For clarity, juxtaposition implies the general act of positioning close together (or side-by-side), regardless of whether such positioning has been done in a regular or irregular fashion. However, I cover irregular side-by-side arrangements in other types of arrangement, see below.
Design rationale. In addition to design rationale for ARRANGE methods in general, juxtaposing multiple design alternatives should be considered a fundamental capability of multi-state tools [230]. As I mentioned before, but I repeat here for completeness, neatly- and regularly-arranged thumbnails facilitate comparison of multiple (similar) objects, i.e., objects with similar structures repeated across all such objects; this stability of structure allows the user to focus on changes, instead of having to comprehend the structure first. If arranged into a spreadsheet-line interface [169, 33], with one parameter per axis (X, Y), miniatures would also facilitate the user’s exploration of the design sub-space spanned by these two parameters; precise vertical and horizontal alignment of miniatures would then enable easier visual detection of similarities and differences.

As for potential design tradeoffs, while not strictly a grid arrangement but still based on the principle of juxtaposition, in [370, 367], Terry et al report that Parallel Pies (an approach to visualizing multiple alternatives based on pie-like juxtaposition of “slices”, where each slice shows one alternative), albeit facilitating broader exploration of alternatives, can lead to lower-quality design solutions. While this experimental finding related to Parallel Pies may be simply due to the fact that the user is able to view only portions of alternatives at a time, additional research is needed for uncovering possible reasons for this shortcoming of the approach.

Emblematic multi-state functionality. Figure 10.16 shows two possible arrangements into a grid. By changing and choosing a different sequencing criterion (e.g., from date created to date last modified) the user can easily and rapidly re-sort (i.e., re-order, re-sequence) a set of alternatives either in the ascending or descending order. Alternatively, alternatives could be arranged in an X-Y grid where each axis represents different values for two different parameters; by modifying those values the user could then easily explore the underlying two-parameter sub-space of designs.

![Figure 10.16: Multiple alternatives arranged as a sequence in a rectilinear grid, with two different expression types for alternatives (thumbnails and glyphs).](image)

Layers (Superposition) Arrangement

With these arrangement methods, all the alternatives in the collection are arranged on top of each other, following some z-order. Some means to navigate layered alternatives should be provided.
**Design rationale.** In the literature related to multi-state systems, for example in [220] Lunzer suggests semi-transparent displays whereby multiple states can be rendered on top of each other (i.e., superposed) in a single view.

In CAMBRIA 1 expert study feedback, a participant stated that he would predominantly use the superposition when editing in parallel [P4 37:37], and that “this is where the action is”. Further, this arrangement reminded him of tracing paper as commonly used in architectural design. Another participant found superposition “most interesting” [P6 1:06:30].

As for potential weaknesses, obviously, stacking alternatives on top of each other can quickly lead to visual clutter, inability to discern among items, as well as the inability to effectively select, edit and manipulate individual items. The IxD researcher-designer should pay attention to this design trade-off accordingly; one workaround might consist of being able to select and display just some (i.e., a few) design alternatives within the collection, while temporarily hiding all the rest.

**Emblematic multi-state functionality.** Figure 10.17 show an example of displaying multiple design alternatives layered on top of each other. Navigation is achieved e.g., by means of a thumbnail strip shown in, or through appropriate key shortcuts.

![Figure 10.17: Multiple alternatives arranged into (optionally transparent) layers.](image)

Finally, CAMBRIA 1 has already shown the way on how such an arrangement view might be implemented (Figure 10.18). In this view based on superposition, the currently active alternative

![Figure 10.18: Superposing multiple alternatives on top of each other in CAMBRIA 1.](image)
is displayed normally, using default colours, while all the other alternatives are rendered using an “inactive”, or grayed-out palette. The user can select the “current” design alternative as desired, thus moving it to the top of the $z$-order and enabling its editing.

**Slideshow Arrangement**

In this arrangement mode, the view (i.e., the entire screen) is at all times occupied by one alternative only. The user navigates between alternatives in one-by-one fashion, either moving forward or backward in the current sequential order for the collection.

*Design rationale.* The advantages of this arrangement include the ability to view each alternative in the maximum size desirable, or possible. This both (1) allows the user to take the full advantage of the entire display area, thus possibly increasing the viewing angle and increasing visual acuity, as well as (2) engage in single-state workflows, if so desired by the user.

The advantages above should be assessed against the inability to view other alternatives in the set, thus introducing discontinuity as well as disruption in the comparison process, and associated demands on the user’s cognitive faculties.

*Emblematic multi-state functionality.* As shown in Figure 10.19, the user moves forward (or backward, as desired) through a sequence of alternatives, ordered by some criterion (i.e., modified date, created date, size, rating, and so on).

![Figure 10.19: Mockup of multiple alternatives arranged as a slideshow.](image)

**Data Table Arrangement**

Another way to sequence multiple alternatives is through a data grids of alphanumerical values, containing rows divided into columns such as Name, Created, Modified, etc.

*Design rationale.* By rendering objects as data tables, the interaction designer can achieve higher density of information, i.e., the user can view much more information on screen as compared to some other (or even most) arrangement types. In addition, the user can easily and rapidly sort (order) objects by selecting column headers, thus ranking all rows by the criterion captured by that column.
This advantage comes at the increased cognitive cost of having to visually scan for textual (alphanumerical) symbols and identifiers within densely-packed information displays. This design trade-off might be ameliorated, for instance, by augmenting data rows with another column containing thumbnails of the alternatives, or by an filter input field for narrowing the number of objects considered to a necessary minimum.

**Emblematic multi-state functionality.** Figure 10.20 shows an example of alphanumerical rows arranged into a data table. The user can click on a column header, and thus re-sort (re-sequence) the set of rows either in the ascending or descending order.

![Table Example](image)

**Figure 10.20:** Multiple alternatives arranged as a sequence of rows displaying alphanumeric data.

**Lifestream (Coverflow) Arrangement**

Similarly to the slideshow arrangement, in this mode, the user either flips (using mouse and touch gestures) or scrolls (using the horizontal scrollbar) through a series of alternatives.

**Design rationale.** In the literature, Freeman and Gelernter present “lifestreams” [122] approach to organizing electronic information based on time-ordered streams of documents, as an alternative to the dominant “office desktop” GUI metaphor. The authors claim that, combined with stream filters to organize, monitor and summarize information, lifestreams can subsume a number of separate views.

Within the context of multi-state systems, a “stream” already implies a plurality of entities. In the CAMBRIA 1 expert study, one participant in fact suggested a version of lifestream arrangement whereby the older an alternative is, the more it “fades” into the background canvas, thus indicating its age [P1 1:21:55]. In industry, this presentation mode achieved wide user acceptance, for example through the “CoverFlow” browser [17] as originally implemented in *iTunes* and *Mac OS X Finder* by Apple Inc., and other applications by different vendors.

As a potential design tradeoff, trying to reach alternatives at the far end of sequence could lead to longer navigation times, and thus to longer task completion times.

**Emblematic multi-state functionality.** Figure 10.21 shows an example of the “lifestream” arrangement, using the horizontal alignment.
10.2. DESIGN GUIDELINES FOR ALTERNATIVES-ENABLED PCAD TOOLS

Figure 10.21: A mockup of “lifestream” or “coverflow” arrangement.

The user can also “play” the sequence (buttons Backward, Play, and Forward), as well as render the current alternative over the entire screen (button Fullscreen). The alternatives themselves are ordered by some criterion (i.e., modified date, created date, size, rating, and so on). Another version of this arrangement could depict alternatives going “in and out of screen”, or even fade into the background.

An additional mode of lifestream arrangement would be where the user leafs through alternatives stacked on top of each other, perhaps ordered by their order of creation. The next alternative in the order might be shown either at the same geometric position (“geometric coverflow”), i.e., showing the same geometric extent. Or, the next alternative might be shown centered at the same item (or selection of items), i.e., showing the same semantic content (“semantic coverflow”).

Arrangement by Distance Metric

In this mode, alternatives (or miniatures thereof) are being arranged in a view according to some predefined, pair-wise objective “distance metric” \( f(x) \). Metric \( f() \) can be as general as possible, in the sense of being either a regular expression, a mathematical function, or even a general computational method.

Design rationale. The design rationale for this arrangement type include (1) constructing galleries of many (possibly computationally expensive) visual objects defined by different \( n \)-parameter settings, and (2) constructing galleries of said objects when the user has no well-defined idea of what is desired, thus lending itself to serendipitous discovery of interesting designs i.e., for creativity support [239], (3) navigation facilities for browsing galleries.

As for design trade-offs, while a tool might provide a set of predefined distance metrics, some effort would probably be required on the part of the user to define or choose a suitable distance metric, as well as define what constitutes acceptable “perceptual qualities” of designs, in this case visual qualities of pCAD models.

Emblematic multi-state functionality. The user can zoom in and out of the “thumbnail cloud” shown in the centre of Figure 10.22, with the corresponding updating of both the smaller thumbnails in the centre, as well as of the larger thumbnails placed on the outer rim of the view.
Design History Arrangement

In this mode, alternatives are being arranged according to their design history (i.e., lineage, or relation of derivation). Since lineage leads to directed acyclic graphs (DAGs) in general, the vast area of research related to general graph visualization applies, see e.g., [393].

Design rationale. Interactive visualization of design history is universally recognized to be a crucial capability for any computational tool [331]. Many approaches to visualizing design histories exist; I covered a number of such approaches to visualizing design histories in the chapter devoted to related work, see Section 5.3 on page 107.

In the CAMBRIA 1 expert study, participants expressed the desire to have a linear “chain” of sequentially derived alternatives (thus displaying their design history), whereby the user can go back to the initial sketch and then see how changes in that sketch affected sketches that had developed downstream, thus comparing all the sketches in the chain [P3 1:11:01]. This, in turn, indicates a desire for being able to modify the design history of the model. Another participant stated that he sees the juxtaposition mode as a type of design history as well, where each edited alternative shown in the grid represents “a snapshot of a particular evolvement of the design”, because it allows him to simultaneously see and compare multiple alternatives without being forced to execute rapid do-undo-redo cycles as in conventional single-state systems [P6 1:14:22]. In CAMBRIA 2 study, in my experience, the “design history” arrangement mode was very useful for understanding the lineage of design alternatives, and was finding myself reaching for that option rather often.

Emblematic multi-state functionality. Figure 10.23 presents just one (conventional) way to visualize a general graph, among many possible.
10.2. DESIGN GUIDELINES FOR ALTERNATIVES-ENABLED PCAD TOOLS

In this mockup, the directed acyclic graph expands from left to the right, and from top to the bottom. Clicking on the “year” field would cause the visualization to zoom out to a range of years, say 2010-2020, while clicking on any of the months would zoom into that month, and replace the row Jan-Dec of months with a row of days in a month (1–28, 1–30, or 1–31).

Figure 10.24 shows the “design history” arrangement mode as implemented in CAMBRIA 2. The user can cycle to and from the “design history” arrangement seamlessly, no matter which arrangement the tool currently finds itself in.

Random Arrangement

In this mode, alternatives are being arranged randomly i.e., their positions, sizes (widths and heights) and angles are being generated randomly, of course within reasonable ranges.
**Design rationale.** This would include the possibility to perceive and evaluate alternatives in a serendipitous manner, and perhaps the possibility to uncover an interesting or relevant detail exposed by random reshuffling of alternatives. In CAMBRIA 2 study, I was reaching for this command rather frequently when I wanted to “break up” the current configuration, and perhaps come up with a novel, “fresh” and revealing configuration. To a smaller degree, every activation of this command laid out alternatives in a different random arrangement or “pile”, thus repeatedly revealing different parts of different alternatives as well.

**Emblematic multi-state functionality.** Figure 10.11 (leftmost) on page 238 depicts one such arrangement into a randomized “pile” of alternatives.

### 10.2.3.2 Manual ARRANGE Methods

**DG-6. Allow the user to arrange objects manually, in order to create his own meaning.**

These arrangement types, differently from previous ones which imply an one-dimensional ordering, refer to the ability to place an expression into a two-dimensional plane.

**Free Manual Arrangement**

In this mode the user can freely reposition, resize, rotate, and group (cluster) multiple alternatives manually, e.g., by dragging the image on the canvas, resizing the image by dragging the border of the image, rotating them, and creating his or her own groups.

**Design rationale.** In this mode, the user can create proximity-based clusters of alternatives, i.e., create separate “sub-collections” within the current collection. In CAMBRIA 1 study, I found support for being able to group (i.e., arrange, cluster) thumbnails of alternatives [P1 1:13:57] [P2 56:30] [P2 1:02:30] [P3 49:38] [P3 1:27:37] [P4 1:23:40] [P6 55:06] [P6 55:36] [P6 56:29] [P6 57:23] [P6 59:51] [P6 1:02:55], and that this mode can even be effectively used as a substitute for collections, by creating “sub-collections” mentioned above. Further, the user should be able to change relative sizes of thumbnails, in order to indicate their relative importance, both for alternatives and collections [P2 33:27] [P2 34:21].

As for design trade-offs, some of the alternatives will unavoidably become hidden (occluded) behind alternatives at the front of the heap/cluster. A possible work-around might consist of rendering all clustered alternatives in a semi-transparent fashion [220].

**Emblematic multi-state functionality.** Figure 10.25 shows a typical multi-state functionality, in this case of multiple design alternatives arranged into two separate, manually created “heaps” or “piles”, thus in effect creating two “sub-collections” within the collection. The user can adjust the position, size, and rotation angle for each alternative separately.
10.2. DESIGN GUIDELINES FOR ALTERNATIVES-ENABLED PCAD TOOLS

Grid-Based Manual Arrangement

As an alternative way to arrange manually freely, manual arrangement in the grid allow the user to change the positions of objects shown in the grid, but not their sizes or orientations.

*Design rationale.* This arrangement would lend itself well for introducing beneficial order (in terms of alternatives’ sizes and orientations), while still being able to move and group design alternatives accordingly.

*Emblematic multi-state functionality.* See any of the multi-state functionality presented regarding the rectilinear grid (juxtaposition) arrangement on page 242.

10.2.4 NAVIGATE Methods

The user should be able to *navigate* (i.e., incrementally build a mental map of, understand, and comprehend) multiple design alternatives in flexible ways. Navigating a number of related alternatives presents a number of unique challenges, compared to navigating a single document. Based on my experiences building interactive prototypes, as well as on related literature, I propose the following design guidelines for alternatives-enabled tools, classified into the following categories:

(1) Panning
(2) Zooming
(3) Overview + detail
(4) Focus + context.

10.2.4.1 Panning


Panning moves the center of the viewport to an off-screen portion of the model. GUI commands that facilitate and enable panning include for instance *Scroll* through scrollbars (the user can click...
on *Up* and *Down* buttons on the scrollbar, or drag the bar located on the scrollbar “gutter”), as well by pressing *Page Up* and *Page Down* keys.

**Design rationale.** The amount of information describing a model or a set of models might be much larger than what can be displayed or fitted into available screen display area. Panning methods enable the user to view and interact with those hidden parts of the model.

With multi-state tools, the user might decide to focus on a certain region, for multiple design alternatives, and then continue editing in that region. The conventional single-state panning techniques would then be rather inadequate for this scenario, because the user would have to perform *n* panning operations for *n* visible design alternatives. Within the framework of CD, one might say that single-state panning would thus lead to excessive viscosity as well as poor visibility scores.

A weakness of panning includes the *visual discontinuity* between the current (visible) part, and panned-to parts of the model [72]. This discontinuity in turn might lead to excessive cognitive loads on the user [145, 72]. To ameliorate this weakness, which might only become worse (exacerbated) with multi-state tools, panning is thus best used in conjunction with other navigation methods such as zooming, to be described below.

**Emblematic multi-state functionality.** In alternatives-enabled pCAD tools, panning can (if so desired by the user) occur concurrently, for all the active alternatives. In other words, the operation of panning might be synchronized across a number of currently active alternatives: as the user pans across one alternative by a certain displacement \( \Delta = (dx, dy) \), all the other active alternatives should pan by the same amount \( \Delta \) (as measured in their own coordinate system).

![Figure 10.26: Multi-state (interactive) panning in parallel in CAMBRIA 2.](image)

Figure 10.26 shows an example of parallel panning as implemented in CAMBRIA 2. As the user drags the mouse point in one alternative and at the same time keeps the modifier key Alt depressed, all the active alternatives will pan in parallel by the same amount.
10.2. DESIGN GUIDELINES FOR ALTERNATIVES-ENABLED PCAD TOOLS

Figure 10.27: Multi-state (automatic) panning “to an item” in parallel in CAMBRIA 2.

Figure 10.27 shows another variant of panning, where the user pans to linked items in other alternatives. In *before* states, the “crankset” part is visible only in the top left alternative. By choosing the “pan-to-item” command, CAMBRIA 2 pans automatically to all the associated crankset parts in all the other alternatives, as shown in *after* states.

10.2.4.2 Zooming

**DG-8. Provide parallelized zooming methods.**

Zooming changes the scale and/or resolution of the model, so as to magnify/shrink an area, thus achieving a *temporal* separation between contextual and focused-in (zoomed-in) views. This way, the user can access more information than can be shown in the available display area.

The user should be able to interactively specify the zoom factor and the pan position defining the viewport into a region of the alternative that is currently being worked on. A variant of zooming, also called *semantic zooming (elision)*, modifies the appearance of items as a function of the current zoom factor. The user should also be able to execute such panning and zooming actions in a synchronized manner, across all currently active alternatives.

**Design rationale.** Just like with panning, the justification for zooming functionality is to enable viewing of currently invisible parts or details of the model.

In the CAMBRIA 1 study, the zooming operation was one of the most often used functionality of the tool, demonstrating its overall importance in the interaction. Additionally, one participant stated that a user should be able to “zoom into a detail”, and “work at various levels of detail” [P5 51:26].

As for potential weaknesses, Plumless and Ware [287] determined that error rates are much higher for zooming interfaces than for interfaces utilizing multiple windows, and recommend HCI designers to implement multiple windows when “visual comparisons must be made involving patterns of a greater complexity than can be held in visual working memory”. This problem might be
more pronounced for multi-state tools, due to higher count of displayed design states.

**Emblematic multi-state functionality.**

Figure 10.28 shows an example of parallel zooming as implemented in CAMBRIA 2. As the user clicks on the “zoom out” option or rotates the mouse wheel while keeping the modifier key Alt depressed, all the active alternative zoom in or out by the same scaling amount.

![Figure 10.28: Multi-state (interactive) zooming in parallel in CAMBRIA 2.](image)

A zooming equivalent of multi-state panning “to an item” is also possible; just like with panning, if the user selects the “zoom to item” command, the tool zooms to all the selected items in all active design alternatives. In most cases, this will require a combination of zooming and panning.

### 10.2.4.3 Overview+Detail

**DG-9. Provide various overview+detail navigation methods, such as thumbnail strips or overview insets.**

With these methods, there are two kinds of spatially separated views on display: (1) a broad overview of the set of alternatives contained within a collection, allowing the user to more rapidly acquire the mental map of the alternatives in the set, and (2) detailed view allowing the user to work with alternatives in detail. Overviews could be implemented in a number of different ways.

**Design rationale.** Just like panning and zooming methods, overview+detail methods facilitate the acquisition of the user's mental map of design alternatives and collections, as well as allow the user to show much more information on the display than permitted by its available physical display area alone.

As for potential weaknesses and necessary tradeoffs, the discussion from page 238 applies in this case as well: *thumbnails*, or scaled-down images, afford various benefits to the user (recognition of past states, proxy management of data objects, and aiding with comparison. Different
thumbnail sizes result in different recognition rates. The clarity of thumbnails is inversely proportional to their size, and the speed of access is directly proportional to their size.

**Emblematic multi-state functionality.** One way to navigate multiple design alternatives is the so-called *enhanced scrollbar*, which is a conventional scrollbar however enhanced with previews of alternatives as the user moves the mouse pointer over it. Every location on the scrollbar maps into a specific alternative in the set.

A *strip of thumbnails*, on the other hand, can be considered a generalization of the enhanced scrollbar. The width of a thumbnail strip should be adjustable by the user, from narrow (thus allowing for smaller thumbnails) to wide (fewer thumbnails yet a better overview of individual alternatives). Also, the strip should be one-way synchronizable with the detail view: clicking on an alternative in the detail view synchronizes with the strip, however the strip can be scrolled independently from the currently displayed alternative(s) in the detail view, thus facilitating navigation within a collection [72].

![Figure 10.29: A mockup of a strip of thumbnails for overview+detail navigation of multiple design alternatives within a collection.](image)

Figure 10.29 (left) shows one such possible thumbnail strip, with each thumbnail representing one alternative in a collection. In the detail view, here using the juxtaposition arrangement, the alternatives themselves might or might not be transparent, both visually as well as in terms of editing operations.
Figure 10.30: A mockup of an inset overview pane for overview+detail navigation of an alternative.

Figure 10.30 showcases an *inset overview* (or “view-let”), or a relatively small, separate overview pane located in the corner of the detail view, and which allows the user to locate and place the current “view extent” within the detail view, thus facilitating wayfinding. The current zoom extent is indicated by the blue rectangle, thus helping the user to orient himself within the alternative.

### 10.2.4.4 Focus+Context

**DG-10.** **Provide various focus+context navigation methods, such as subjunctive cursors for parallel fisheye lens techniques.**

Differently from zooming (temporal separation) and overview+detail (spatial separation), distortion techniques do not separate the two areas (focal and context) neither temporally nor spatially, but embed the focal area directly (i.e., spatially) within the context area.

*Design rationale.* Due to the absence of discontinuity in distortion methods, they do not cause the user to switch his attention from the focal to context area and vice versa, thus placing less demand on the user’s short term memory. However, due to the distortion effects, these methods might lead to an inaccurate cognitive map of the underlying information space, as well as to certain imprecisions in interaction [72].

As for potential weaknesses, the distortion effect negatively influences both the screen target acquisition, and could cause the user to misinterpret data. One way to alleviate the misinterpretation problem is to impose rectilinear grids on top of images [417].

*Emblematic multi-state functionality.* One possibility for distortion-based methods within the context of multi-state tools is a combination of *subjunctive graphical cursors* and *fisheye lenses.*
Figure 10.31: A mockup of multi-state focus+context navigation functionality, based on a combination of “subjunctive cursors”, and fisheye lenses.

Figure 10.31 showcases such a functionality, with the areas around all instances of “subjunctive cursors” magnified through the fisheye distortion. Each instance of the subjunctive cursor might be moved either in parallel, or individually.

Figure 10.32: A mockup of multi-state focus+context navigation functionality, based on a fisheye lens applied to a sequence of rows displaying alphanumeric data.

Another example is shown in Figure 10.32, an example of table lens which, similarly to fisheye lens, is only applied to one or more rows in a table, thus visually distinguishing the same from the remaining rows. An advantage would then be being able to densely pack a lot of information into an region within the screen display area; the disadvantage would include the costly cognitive activity of visually scanning the grid of (numerous) cells, in order to find desired information.

### 10.2.5 EXAMINE Methods

These methods allow the user to further examine, inspect or study the working set, the objects in the current scope, or any set of computational objects in general.

#### 10.2.5.1 Details-on-demand

DG-11. *Provide powerful details-on-demand capabilities, so that the user can obtain as much information on items of interest as is possible.*
Refers to the functionality for retrieving more information about the object of interest. Ways to implement this functionality include for example “tooltip” indicators which appear when the user moves the mouse pointer over an object, or a separate view(let) showing additional information about the object.

**Design rationale.** To obtain additional information on the selection; such information is in conventional details-on-demand approaches commonly rendered in alphanumeric form (i.e., as text, prose, or data fields). In CAMBRIA 1 expert study feedback, when viewing multiple alternatives, it would be useful to have additional information displays (“dashboards”) showing alphanumeric information about each alternative (and not only their graphic rendering) [P3 1:39:17] [P3 1:40:04].

**Emblematic multi-state functionality.** Figure 10.33 shows one conventional technique to obtain additional information on a design alternative within a collection.

![Figure 10.33: Details-on-demand: tooltip (on mouse-over) and “detail” pane (populated with data from the selected object).](image)

Other potential approaches might include, for instance, “parallelized tooltips” which, in addition to brushing, might display additional information about all the (linked) instances of an item over which the user has placed the mouse pointer, for all active alternatives.

### 10.2.5.2 Filter-in and -out

**DG-12.** Provide filter-in and filter-out capabilities, so that the user can obtain as much information on items of interest as is possible.

Refers to the functionality for either “filtering-out” (i.e., hiding) or “filtering-in” (i.e., showing) certain items in the computational model.

**Design rationale.** Sometimes, the user desires to focus on just some aspects or elements in the computational model, in order to facilitate closer examination of the same. Filter-out methods suppress (i.e., exclude) all the items matching some criteria from the current scope. Filter-in
methods are similar, only that instead of suppressing items matching filter terms, they only show the items matching the terms.

**Emblematic multi-state functionality.** In Figure 10.34, items 4, 6, 2, 1 have been surpressed in parallel, for all the active alternatives in the collection (all except alternative #2, since it has been delinked).

![Figure 10.34: Filter out items within active alternatives. Note that alternative #2 is inactive, therefore filtering does not apply to it i.e., items 1, 2, 4, 6 are not surpressed.]

Alternatively, instead of typing in keywords, the user might conduct a search by uploading the image (screenshot) of an alternative, or upload the alternative itself in its native format.

**10.2.5.3 Highlight**

**DG-13. Provide capabilities that enable the user to highlight items of interest within a certain scope.**

Refers to the functionality that highlights certain items in the computational model, and that satisfy some search criteria.

**Design rationale.** These methods are similar to filter-in or filter-out methods, only that instead of showing or suppressing items, they highlight items matching some criteria, i.e., make those items more visible or prominent, again for the purposes of enabling the user to focus on some elements of the computational model, in order to facilitate his closer examination and apprehension of the same.

**Emblematic multi-state functionality.** For instance, in Figure 10.35, items 4, 6, and 2 have been highlighted in parallel, using colour coding, for all the active alternatives in the collection.
10.2.5.4 Brush

**DG-14. Enable brushing for associated (linked) objects.**

Brushing refers to the process of interactively highlighting related (linked, associated) items in different views.

**Design rationale.** By visually emphasizing linked items or objects, the tools thus helps the user to understand which item correspond to which item(s) in other alternatives; in other words, which items are *linked* (see p. 276 for a discussion of linking) or *associated*. The user can thus build a mental map of the alternatives under consideration.

**Emblematic multi-state functionality.** Figure 10.36 showcases a visual indication of linked items on mouse-over event, in this example of all four instances of rear wheels and which can be thus be considered as one, “subjunctive” item.

In this case, if the user moves the mouse pointer over an item in one alternative (i.e., the alternative’s view), then all the corresponding items will be visually highlighted in all the other active alternatives. In case of large zoom factors, an application setting might cause brushing to automatically pan to related items in all the other active alternatives.

10.2.5.5 Difference

**DG-15. Provide powerful and varied methods for differencing objects.**

An alternatives-enabled tool should allow and assist the user to find out similarities and differences between multiple alternatives, either computationally or purely visually. Note that the *brush* methods (see p. 260) already provide basic matching on mouse-over, however *match* provides functionality focused on determining *similarities* and *differences* among designs.
10.2. DESIGN GUIDELINES FOR ALTERNATIVES-ENABLED PCAD TOOLS

Figure 10.36: Brushing of linked (associated) items in CAMBRIA 2 using colour coding. (All four rear wheel instances rendered in red.)

**Design rationale.** Human beings *compare*, *rank*, *examine*, and *study* objects. In the classification of high-level tasks in Section 10.1.1 on p. 218, the activity of *comparison* figures prominently; while *compare* is a high-level *task*, the methods of *difference* allows the user to find out the relative degree of difference among two or more objects, using some set of criteria (objective, or personal/subjective).

In the CAMBRIA 1 expert study, one participant stressed the importance (or desire) of somehow bringing out the differences among multiple alternatives [P2 53:37]. The same participant also suggested a feature akin to the “spot the differences” game for kids [P2 1:00:26] and which I presented above. The inability to compare alternatives, and parts thereof [P2 58:12], [P2 1:00:26] is a major shortcoming of the prototype. Another participant would like to visually depict and/or indicate differences among multiple alternatives, using layers (superposition) [P2 36:53] [P2 52:31][P2 53:37].

**Emblematic multi-state functionality.** A multi-state tool might provide special visualization *modes* in order to facilitate the user’s visual scanning process, or offer various *computational* differencing methods.

1. “Spot-the-differences” differencing. In this mode, inspired by the popular “spot-the-differences” game and as suggested by a CAMBRIA 1 study participant, the user selects one or more items in the working alternative, and all the related items are highlighted automatically in all the
other alternatives in the collection as well, possibly by rendering a local “focus” area as shown in Figure 10.37. In addition, the three circular focal areas are connected by lines for easier parsing, especially when more than one focal area is present in one or multiple alternatives.

Figure 10.37: A mockup of the Spot-the-Difference mode (based on local focus) for three design alternatives. Here, item 6 matched and highlighted in all three design alternatives.

In addition, the rest of the model is visually de-emphasized (grayed-out) for additional wayfinding and visual scanning assistance to the user.

Figure 10.38 suggests an approach to differencing geometric designs inspired by the “parallel coordinates” approach to visualizing high-dimensional multivariate and geometric data [166]. In this approach, each point is located at the centre of an item; alternatively, those points can point at another arbitrary but fixed and well-defined positions, such as e.g., the top left corner of the bounding box.

Figure 10.38: Differencing (mode 2, colour-coded) for three alternatives, using “parallel coordinates” connecting centers of bounding boxes for items 1, 2, and 3. By observing slopes of lines connecting parallel coordinates, the user can easily perceive differences in placement along the vertical axis.

In the case of Figure 10.38, it is evident that item 2 is approximately at the same vertical position in all three alternatives (maybe somewhat lower in Alternative #1). It is harder, however, to discern that their horizontal positions are very different. Therefore, the weakness of this “parallel coordinates” mode is that it works well along one axis only; workarounds might include the ability to rapidly switch between horizontal and vertical alignments.

Figure 10.39 suggests another approach to directly compare design alternatives by stacking
them on top of each other, similarly to tracing paper as used in architectural drafting, and then utilizing identical colours for all the instances of an item. This could help the user to quickly identify the relative positions, sizes, and rotation angles for said instances within their alternatives.

Figure 10.39: Differencing three alternatives using superposition (layers). Here, items 1 and 4 highlighted in green and red, respectively, indicating their differences in placement, size, and shape.

In the case of Figure 10.39, and as mentioned in the section on arranging design alternatives in layers (§ 10.2.3), stacking alternatives on top of each other can quickly lead to visual clutter, inability to discern among items, as well as the inability to effectively edit and manipulate individual items; the interaction designer should proceed accordingly.

(2) List-producing differencing. This command algorithmically produces a text-based list of all the differences among two or more alternatives, aggregated per unique items (i.e., subjunctive items), and presented e.g., in a table.

For instance, if an item \textit{myItem} has multiple instances in multiple alternatives, the command would produce an entry \textit{myItem} in the list of differences, containing references to corresponding alternatives, as well as differing values for the attributes of \textit{myItem} in all active alternatives.

### 10.2.6 EDIT Methods

Editing refers to the activity of changing the structure or property values for an alternative. Optionally, an alternatives-enabled tool should allow for \textit{parallel} or \textit{concurrent} editing of multiple design alternatives at once. This category subsumes the following methods:

(1) Selection

(2) Changing the “linked” status of alternatives

(3) Methods that modify states:

- Modify properties
• Transfer items (copy, paste, cut, pass items, pass properties)
• Substitute an item (i.e., a variable) in-place
• Snap points
• Associate items
• Undo and redo
• Preview

(4) Annotation methods.

10.2.6.1 Select (CHOOSE-SELECT)

These methods, which allow the user to choose items which will then be subjected to consequent editing operations, are commonly considered an “editing” operation as well, which is the reason why I am mentioning them in this section. However, in this work, I assume the position that “select” methods and operations should be more properly be considered as a part of “choose” methods. Accordingly, I already described “select” methods in in CHOOSE-SELECT Section (see § 10.2.1.3 on page 236). Alternatively, one may adopt the alias EDIT-SELECT for CHOOSE-SELECT, and treat select methods as belonging to two categories at once.

10.2.6.2 Change Edit Status (EDIT-CHANGE-STATUS)

DG-16. Enable the user to modify the editing status of an object, for example from “linked” to “unlinked”, and other types of status as needed.

These edit methods allow the user to adjust the “edit status” of the alternative under consideration. Once set, the editing status will affect all subsequent design state editing operations in the strict sense.

Design rationale. In multi-state systems, the user might decide to work with just a subset of alternatives within a collection. The tool should thus allow the user to modify or adjust different types of “editing status” by manipulating corresponding visualization and GUI elements, such as setting the alternative as being “linked” or “unlinked”, thus accepting (or not) changes from other alternatives.

Emblematic multi-state functionality. Accordingly, the user should be able to manipulate an alternative (i.e., view thereof) within the collection. This includes displaying the object’s name, and the abilities to link and unlink, check and uncheck, maximize, minimize, and other GUI manipulations.
While many preceding figures in this chapter depict collections and alternatives in a highly abstracted fashion (i.e., as a simple rectangle), Figure 10.40 illustrates various decoration and view management features as well.

(1) The name of the object (in this case, alternatives).

(2) Indicator of selection status (selected, or not selected).

(3) “S” or “Solo” button, which toggles the alternative as the only editable alternative in the collection. Therefore, when this button is depressed, the user works with one alternative only.

(4) “M” or “Mute” button, which unlinks the alternative so that it does not receive any parallel command from elsewhere.
(5) Minimize button, which minimizes the view, and automatically makes the alternative unlinked.

(6) Maximize button, which displays the alternative in full screen, and at the same time makes it “solo” i.e., the only alternative receiving changes.

(7) Close button, which removes the alternative from this collection.

(8) Multiple viewlets or views-within-a-view. This refers to having multiple “viewlets” within a “master” view associated with a design alternative. Such viewlets could be opened and closed either manually by the user, or automatically as defined in the model, and as a function of what is currently displayed within the master view.

Figure 10.42: “Viewlets” within an alternative’s view. In this example, alternative #3 has three viewlets opened.

For example, Figure 10.42 depicts three different “viewlets” created or opened within the “master view” for Alternative #3. Each viewlet shows a different portion of the alternative, with different zoom factors and display extents. The design rationale for viewlets includes, foremostly, improving on the juxtaposability score of the tool; see Section 9.3.2 on page 197 for details.

10.2.6.3 Modify State (EDIT-MODIFY)

This large sub-family of EDIT methods, EDIT-MODIFY, changes or alters a design state (or multiple states) in some way, and consists of the following sub-families of methods:

(1) Modify property value directly.

(2) Insert.

   (a) Insert search results.

   (b) Insert previews.
(c) Insert new objects by generating them.

(3) Delete.

(4) Group/ungroup.

(5) Transfer.

(6) Substitute.

(7) Associate.

(8) Undo/redo.

Modify Property Value Directly (EDIT-MODIFY-PROPERTY-VALUE)

| DG-17. Enable the user to easily and quickly modify property values, e.g., through direct manipulation techniques, contextual menus, and property editors. |

This sub-family contains methods that are familiar from single-state systems, and which allow users to modify property values using direct-manipulation techniques. For example, pen thickness (for a selection of items) is usually changed by selecting an entry from a drop-down box GUI control, or from a contextual menu option (after right-clicking on the selection), or by typing the value directly into an input field. As another example, the colour of selected items can be modified by invoking a standard “colour picker” GUI control.

Note that all such methods can be parallelized, simply by the virtue of having linked items within currently visible (linked) design alternatives. Thus, for example, multiple items can be selected in parallel in multiple items; now invoking the “colour picker” on any of these selection sets will change the colour to all the selected items, in all linked alternatives.

Insert (EDIT-MODIFY-INSERT) A sub-family of methods that change a design state (or multiple states in parallel) by augmenting it with new objects. In a way, then, EDIT-MODIFY-INSERT changes the structural composition of the object being edited, which is to be distinguished from EDIT-MODIFY-PROPERTY-VALUE which changes a property value only.\(^6\)

There are many possible ways to insert new objects into the user’s current working set. For example, the user may select all or some search results from CHOOSE-SEARCH (§ 10.2.1.1), all

\(^6\)However, it must be said, just like I presented in Section 7.2.6 on page 139, that in a sufficiently strong or feature-full formalism, continuous modifications might cause discrete structural modifications. This holds, in particular, if a property contains or points to a “structure” (i.e., the value of the property is a “pointer”), instead of containing just an atomic value such as a number, or a colour code. This holds, as well, if the formalism allows deletions and insertions from the normal (i.e., embedded) control flow logic.
or some preview results from CHOOSE-PREVIEW (§ 10.2.1.2), or insert a new object by hand (for example, drawing a curve by dragging the mouse pointer, or by inserting a new blank collection into the current design project). Thus, classes of methods within EDIT-MODIFY-INSERT naturally include but are not limited to:

(1) Insert a sub-state found through searching, a type of CHOOSE described in § 10.2.1.1. Using the pseudo-operator form, this method could be called EDIT-MODIFY-INSERT-SEARCH-RESULT.

DG-18. Support the easy and quick insertion of search results into the corresponding scope.

(2) Insert a sub-state found through previewing, a type of CHOOSE described in § 10.2.1.2, and which might be called EDIT-MODIFY-INSERT-PREVIEW.

DG-19. Support the easy and quick insertion of previews into the corresponding scope.

(3) Insert a sub-state by generating it, a method which might be called EDIT-MODIFY-INSERT-GENERATE.

DG-20. Support multiple, flexible and powerful methods for generating new objects of interest.

The EDIT-MODIFY-INSERT-GENERATE can be considered as an important sub-class of methods within Insert, because it allows for generating or creating objects of interest (projects, collections, alternatives, and items). Figure 10.43 shows some major variants of EDIT-MODIFY-INSERT-GENERATE.
A new project $P$, for instance, can be created in any of the following ways:

1. Create a new (blank) project.
2. From an existing project. Duplicate an existing project $P'$, in order to create $P$.
3. From existing collections. Create a new blank project, and then import selected collections into it.
4. From a selection of alternatives. Create a new blank project, and then import selected alternatives into it.
5. From a selection of items. Create a new blank project, and then import selected items into it. Each selected item is treated as a design alternative in the new project.

Likewise, a new collection $C$ can be created in any of the following ways.

1. Create blank collection. Create a new, blank collection within the current project.
2. Duplicate collection. Create a new collection by duplicating an existing collection.
3. Merge collections. Create a new collection by merging all the alternatives contained within selected collections.
(4) *From a selection of design alternatives.* Creates a new blank collection, and then import selected alternatives into it.

(5) *From a selection of items.* Create a new blank collection and then import selected items into it. Each selected item is treated as a separate design alternative in the new collection.

(6) *Programmatically.* Finally, a collection might be created by executing arbitrary code, and then populating it with desired alternatives.

Some methods to create a new alternative:

(1) *Blank alternative.* Create a new, blank alternative (within a collection) into which the user can commence inserting pCAD objects.

(2) *From a selection of alternatives.* Create a new alternative by merging all the selected alternatives.

(3) *From a selection of items.* Create a new alternative by merging all the selected items.

(4) *Programmatically.* Finally, an alternative might be created by executing arbitrary code.

Finally, in order to generate items within design alternatives, one may:

(1) *Draw an item.* Create a new item, either within a single design alternative, or within multiple alternatives (i.e., through parallel drawing).

(2) *Duplicate item(s).* Duplicate the selected items in parallel, for all the active alternatives.

(3) *Programmatically.* An item might be created by executing arbitrary code.

**Delete (EDIT-MODIFY-DELETE)**


These methods are the counterpart of EDIT-MODIFY-INSERT, in the sense that they also change the structure of the edited object, only in this case some objects contained in a design state are being *deleted*, instead of inserted.

Within the context of multi-state systems, parallel DELETE is possible, for example, by executing CHOOSE-SELECT in parallel, and then selecting the corresponding *delete* menu option or command.

**Group and Ungroup (EDIT-MODIFY-GROUP, -UNGROUP)**
DG-22. Allow the user to group and ungroup items.

A class of EDIT-MODIFY methods that allow the user to group contained objects into groups, or container objects. This is an important concept, first described in Chapter 7 on the conceptual model, as “composite items”. The importance of group/ungroup methods, as well as their design rationale, is described elsewhere in this dissertation, see e.g., pages 204, 284, or 314.

Transfer (EDIT-MODIFY-TRANSFER)

DG-23. Enable capabilities to move or transfer items from one container object, to another, such as copy, paste, cut, and parallelized versions thereof.

This class of methods allow the user to transfer (move, propagate) objects or values from their (source) container object, into another (target) container object.

**Design rationale.** In the CAMBRIA 1 expert study, I found support for easy and flexible moving alternatives among collections [P2 23:31], [P3 50:38], [P6 1:02:21]. Several participants found the passing (and linking, see p. 276) mechanisms interesting [P1 1:23:30] [P2 50:22] [P3 1:11:01] [P4 28:23] [P5 35:46] [P5 36:32].

**Emblematic multi-state functionality.** Besides the standard operations of Copy, Paste and Cut, with multi-state systems the user should also be able to transfer items (i.e., variables, or parts) into other alternative, as well as the values of their properties onto associated items in active design alternatives.

(1) “Pass items”. A parallelized version of Paste, refers to the interaction mechanism whereby a part (along with all its properties) of a pCAD model is copied into other, currently active alternatives (Chapter 8, § 8.1.3).

![Figure 10.44: “Pass items” mechanism in CAMBRIA 2.](image)

Figure 10.44 demonstrates the “pass items” capability, as implemented in CAMBRIA 2.
(2) “Pass properties”. Refers to the interaction mechanism whereby a property value (such as e.g., position, size, length, line thickness, colour, and so on) is transferred to corresponding (or linked, associated) parts in all the other pCAD models currently active on screen.

![“Pass pen colour” (before) and “Pass pen colour” (after)](image)

**Figure 10.45**: “Pass property” (in this case, pen colour) mechanism in CAMBRIA 2.

Figure 10.45 demonstrates the “pass properties” functionality, in this case passing the pen colour of the currently selected item.

**Substitute (EDIT-MODIFY-SUBSTITUTE)**

| DG-24. Enable the user to substitute (replace) objects of interest. |

A family of methods for substituting or replacing parts of alternatives. If we view the model (or multiple models i.e., design alternatives) as directed acyclic graphs modeling data propagation channels, in practical terms this means that a certain sub-graph $g$ of the global graph $G$ is replaced by another sub-graph $g'$. This, in turn implies the following two processes of:

1. Removing $g$ from $G$, which entails disconnecting all the existing dependencies from $g$ to $G$ and from $G$ to $g$, and

2. Inserting a new sub-graph $g'$ into $G$ and then restoring (to the maximum possible extent possible, since in general $g'$ differs structurally from $g$ to a greater or lesser extent) previously disconnected dependencies, again from $g'$ to $G$ and from $G$ to $g'$.

**Design rationale.** Sometimes, a choice of a design component turns out to be wrong. In other words, it may well happen that a designer or user initially commits to a particular choice or design decision (in most cases, due to lack of information in the future development of the always-fluid design situation), and then subsequently discover that the initial choice or decision was not an optimal one. For example, in code design, at the outset of a coding effort, a programmer might initially commit to a particular control structure, only to go through multiple instances of changing
that same control structure [136, 138]. Substitute methods allow the user to correct his initial choice, thus making the design right. Within the CD framework, these methods thus alleviate the problem of premature commitment.

The second possible design rationale, or justification, for substitute methods hails from shape grammars [356, 355, 197, 256]. A shape grammar $SG$ can be defined as a tuple $(S, L, R, I)$ whereby $S$ is a finite set $S$ containing shapes, $L$ set of symbols denoting shapes, $R$ set of shape rules, and $I$ the set of initial shapes. Each shape rule can be written as:

$$s \rightarrow s'$$

...which effectively denotes the substitution of shape $s$ with another shape, shape $s'$.

Figure 10.46 depicts one such rule intuitively; the rule searches for the three-part configuration on the left-hand side, and substitutes it by the three-part configuration shown on the right-hand side.

Shape rules are not defined in a random fashion, but following established conventions of design in a certain field; in other words, there is always a sound reasoning and system behind each and every such rule. Repeated application of rules in $R$ can then lead to designs which closely resemble designs created by human designers; for example, the architectural designs by Andrea Palladio [356], or “prairie house” architectural designs by Frank L. Wright [197].

In the literature related to multi-state systems, Luzner [221] recommends recombining parts of solutions, in order to come up with better hybrid solutions. Terry recommends “standalone alternatives at any level of granularity” [367, p. 162] as well as to treat parts of models as “containers which permit multiple alternatives to a part” [369]. Terry et al [370] also mention that users have the need to “produce variations of elements of an overall solution”. Further, Jennings [175] advocates making partial solutions “first-class entities”, and Smith et al [345] claim that a computational tool that supports multiple ideas should provide means for “fluid composition and decomposition of ideas”, thus again implying manipulation of constituent components of designs. Finally, Jones [179], within his three-step “divergence-transformation-convergence” model of designing, champions the recombination of partial solutions in the transformation step.

In CAMBRIA 1 expert study feedback, SUBSTITUTE methods could also potentially solve the problem of “small iterations” mentioned by some CAMBRIA 1 study participants [P1 59:45][P2 50:22][P2 55:04], leading to overly similar designs. Namely, the higher up one goes in the
hierarchy of parts in an alternative, a part substitution would lead to a more significant change in the overall design. One participant stated that the user should be able to zoom into an alternative and “work at various levels of detail” [P5 51:26], i.e., be able to modify specific parts of the model as well. Further, a participant mentioned the capability to manage parts (e.g., “wall”, “roof”) of models as well [P6 1:28:18], and furthermore used the term “components” as he was describing the drawing [P6 1:29:12]. Another participant expressed the desire to be able to see alternatives of parts (and not only whole designs) as well [P3 1:42:04]. Thus the prototype should be able to break up designs into their constituent parts, and then explore alternatives only for those parts [P3 1:42:04], [P3 1:44:55], [P3 1:46:16].

**Emblematic multi-state functionality.** To illustrate the EDIT-MODIFY-SUBSTITUTE class of methods, and for readers’ convenience, I will recall the example first given in Figure 5.12 on page 106, and include a slightly modified version in Figure 10.47 below.

![Torso armour alternatives](image)

**Figure 10.47:** Substituting a component of the model (torso armour) in-place, with three different component-level alternatives. Images credit [386].

Figure 10.47 shows three steps in modifying just a certain part (or region) of the overall design model: the torso armour part. Upon being presented with corresponding options, the user can alternate among different design or model alternatives for just one specific region of the overall model.

While this example already implies a set of alternatives (for the armour part), the operation of replacing a part is still being executed within the context of a single state (i.e., the model of the avatar). However, one might easily imagine parallelized versions of EDIT-MODIFY-SUBSTITUTE methods, in which all the linked items or parts in all the active alternatives are being substituted in parallel.
10.2. DESIGN GUIDELINES FOR ALTERNATIVES-ENABLED PCAD TOOLS

Figure 10.48: A mockup of substitute-part functionality, based on pop-up “coverflow” lists of alternatives for a part.

Figure 10.48 shows a parallelized version of SUBSTITUTE. Upon selecting item 4 and choosing the “Substitute variable” command, three separate but synchronized views are spawned, showing a number of design sub-alternatives for this particular item (in this example, using the “lifestream” [122] or “Coverflow” arrangement [17] for part alternatives).

Snap (EDIT-MODIFY-SNAP)

**DG-25.** Enable the parallel snapping technique, whereby the points attach automatically to other objects based on their names, not geometric positions.

Snapping is a common feature in many CAD and vector drawing tools, whereby an end point \( P \) (for instance, the end point of a curve being dragged on screen) is attached (or “snapped”) onto a position on another 1-D curve. In all subsequent deformations of the geometric model, this snapping is preserved. **Subjunctive parametric snapping** extends this to multiple design alternatives.

**Design rationale.** The ability to snap a point into an exact location pre-defined by the tool (e.g., onto another point, or along a curve for some parameterization value \( t \in [0, 1] \)) serves as a means to decrease the complexity of designing, and as a means to impose a certain order or regularity to the model.

As for *subjunctive* snapping, the justification for this particular, novel pCAD interaction technique is, again, imposing a certain order onto the model (since the points snap onto “same” locations in all active design alternatives; only that the meaning of “same” here differs from the conventional, purely geometric sense). There is also a certain element of “chance taking” inherent to this technique, since one would possibly anticipate or expect a serendipitous discovery of a new design or shape, based on this special meaning of “sameness” of locations.
In CAMBRIA 1 study, a participant requested being able to control points snapping to other points [P6 33:07], and the ability to enable control points snapping to corresponding control points in multiple alternatives ("subjunctive snaps") [P6 34:34] [P6 37:37] [P6 38:19] [P6 46:55]. Another participant also suggested snaps [P3 24:04] [P3 28:20] [P3 29:33]. Also, as mentioned in Chapter 9, the feature of “subjunctive snapping” was suggested by my academic advisors as a significant novel feature, or novel parametric CAD ability, that has not been reported in the literature and that should be implemented in the next high-fidelity prototype, CAMBRIA 2. As I presented elsewhere in this dissertation ( Chapters 3 and 4), expert knowledge and opinion is considered to be a valid way to justify beliefs [283]; thus, it can be understood as valid design rationale as well.

**Emblematic multi-state functionality.** In example shown on Figure 10.49, the user starts dragging a control point (indicated by the red crosshair pointer) in the upper left alternative; the target curve automatically turns blue when the tool senses that a snap operation is possible.

![Simultaneous (parallel) point dragging (with Alt key depressed)](image1)

![Subjunctive snapping (final state)](image2)

**Figure 10.49:** Point snapping in parallel across multiple alternatives in CAMBRIA 2.

Even though the dragged end point instances are much farther from the corresponding curves in the remaining three design alternatives, these instances still snap onto the same parametric distance locations \( t = 0.14905 \) in all four alternatives.

**Associate/Link (EDIT-MODIFY-ASSOCIATE)**

**DG-26.** **Allow the user to associate (link) objects in multiple, flexible ways.**

Association or linking is what makes the pass and brush techniques possible. For example, an alternative can only be passed into linked collections. Also, an item can only be passed into linked alternatives; likewise, properties of items can only be passed into linked items.

**Design rationale.** The justification for associating items are three-fold: (1) to help the user understand which items are supposed to be cognate, (2) to understand which items are linked and can thus receive a subset of property values, irrespective of whether those items are cognate...
or not, and (3) once linked, to facilitate transfer or propagation of properties from source items to target items. The first justification helps the user to understand the computational model(s) in question, thus facilitating design decision making; the second one to understand the current associations; while the third one enables novel, effective modes of editing that have not been previously reported in the literature to the best of my knowledge.

In CAMBRIA 1 expert study feedback, almost all participants found the passing and linking mechanisms “interesting” [P1 1:23:30] [P2 50:22] [P3 1:11:01] [P4 28:23] [P5 35:46] [P5 36:32]. Also, it has to be easy to “link” and “delink” items [P1 49:07], as well as link arbitrary items, not only cognate ones [P5 30:11] [P5 35:07] [P5 35:46] [P5 36:32] [P5 39:25] [P6 26:52] [P6 27:11] [P6 27:30] [P6 1:37:09].

In both versions of CAMBRIA, linking is achieved implicitly, through duplicating items or alternatives, which preserves identifiers and thus maintains the linkage amongst duplicated items. However, a richer functionality related to linking is missing. In CAMBRIA 2, as I was working with the prototype, just like the participants in the CAMBRIA 1 study I occasionally felt the need for more varied and powerful linking functionality, in particular the ability to easily link and delink, as well as link arbitrary (and not only cognate) items. Clearly, this represents another potentially fruitful area of future research.

As for design tradeoffs, from the CAMBRIA 1 expert study, linking and de-linking might not work for designs that are “tightly woven” and the linkage would eventually break down [P1 52:23][P1 53:34]. Furthermore, a study participant claimed that the ease of passing items and their values would lead to “small design iterations only” thus negatively affecting the end results of designing in terms of their variety [P2 50:22][P2 55:04].

**Emblematic multi-state functionality.** No specific linkage-editing techniques have been implemented in any of the research prototypes developed in this work. However, one might envision techniques similar to constraint editing techniques as implemented in CAMBRIA 2 (see page 188). The user might (1) select an item (say, a curve $C$) (2) raise its contextual menu and select the “Propagate properties to...” option, and then (3) select a number of items in multiple design alternatives to which the curve $C$ will now be propagating values for a set of chosen properties of $C$. Finally, as another feature related to linking, Figure 10.36 on page 261 demonstrates *brushing* in CAMBRIA 2, thus enabling the user to examine linkages among items.

**Undo / Redo (EDIT-MODIFY-UNDO and -REDO)**

**DG-27.** Allow undo and redo capabilities, with separate undo stacks for each alternative.

Undo/redo technique allows the user to cancel the effect of previous commands. In most cases, all the actions are saved into a linear structure called “undo stack”. Within this work, one
may distinguish among local undo stacks (i.e., those belonging to one particular design alternative) and global undo stacks (i.e., the one registering any and all command carried out in the tool). Furthermore, and as discussed in Chapter 5, one can distinguish among partial or regional undo capabilities [184], featuring associated set of actions that can be undone or redone for the items contained just within that region. The same considerations as for the two-way (local-global) division of undo stacks apply, with the exception that instead of just two levels, one might have a number of intermediate undo levels.

**Design rationale.** Undo/redo is a standard, requisite functionality in just about any computational tool, primarily because it allows the user to quickly recover from unintended errors or mistakes. In addition to error recovery, undo is also used for conducting rapid, near-term, “what-if” experiments [369], for conducting longer-term experimentation (e.g., by reverting to a command/state that is lower in the undo stack), as well as compare two alternatives by performing rapid undo-redo sequences [368, 370, 367].

Within the context of multi-state tools, the availability of separate undo stacks per design alternative facilitates and enables error recovery and rapid experimentation when executing editing commands in parallel, thus affecting multiple states at once.

In the CAMBRIA 1 expert study, undo/redo was requested by some participants [P1 57:19] [P2 27:57]. In CAMBRIA 2, full undo mechanism has been implemented per alternative i.e., each alternative has its own undo stack thus significantly improving the prototype, in my experience, along the dimensions described above (error recovery, and “what-if” experimenting).

**Emblematic multi-state functionality.** There are many ways to explore the question of undo and redo within the context of multi-state system, and I consider this a potentially fruitful area of future research. The first step, obviously, is adding a separate undo stack to each design alternative, instead of maintaining one global stack for all the alternatives in the collection or design project (see e.g. Figure 9.1 on page 174).

However, other variants of undo can easily be imagined for multi-state systems. For example, separate parts of the model might maintain their own, local undo stacks. This holds for all kinds of items, simple or composite; for example, as a new group is created from a set of simpler items, a new undo stack could be created for this new group object as well. As a result, one might achieve full error-recovery and “what-if” capability at all levels of the decomposition of the model.

10.2.6.4 Annotate (EDIT-ANNOTATE)

**DG-28. Offer rich annotation capabilities.**

Refers to adding graphical, textual, or audio annotations to alternatives, their parts, or collections. In a sense, annotations belong to a design state, yet can be considered to be separate from
it. In other words, annotations are a part of the associated design state *stricto sensu* yet not part of it *lato sensu*. Annotations are considered to be *metadata* [209, 377] that provides additional information that in most cases cannot be extracted from the design state itself; such metadata might include but is not limited to [209]:

1. “Biographical” or lineage-related information. This might include e.g., the author, parent revisions (i.e., antecedents in the design history of the state), time and place of creation, who has accessed or modified the state since then, and so on. CAMBRIA 2 is able to show lineage data through its Design History arrangement mode.

2. Subject attributes. Information of and about the design state; for example, a model of a skyscraper might be about the novel developments in the area of sustainable design, within the context of high-rises. In most cases, subject attributes include rich, textual description of the state.

3. Relationship attributes. These are associations and links to related design states; for example, a pCAD model might be linked to a set of related models, or to a collection of raster images.

**Design rationale.** The purpose of annotation is to facilitate understanding, accessing, and subsequent retrieval of associated design states, and as such can be regarded as an essential ingredient of any computational tool or system, single-state or multi-state alike. In addition to the above, within the Cognitive Dimensions framework [139, 138, 41, 140], the secondary notation dimension assesses the extent of availability of secondary means to express meaning.

In CAMBRIA 1 study, one participant expressed the need for commenting or annotation of each design move by audio-recording spoken rationale for the move [P5 36:32]; in the participant’s opinion, such a mode of annotation would be much more convenient than written or typed annotation [P5 42:44], since it would not disrupt psychological “flow” [89].

### 10.3 Design Principles for Alternatives-Enabled pCAD Tools

Just like design guidelines, design “principles” can also be considered to be high-level suggestions or recommendations which help interactive system designers to approach, frame, and solve complex system design problems, as well as heuristically evaluate proposed design solutions or candidates [333, p. 40]. However, Shneiderman and Plaisant [333, p. 62] explain the distinction between the two, and suggest that “guidelines are narrowly focused”, while “principles tend to be more fundamental, widely applicable, and enduring”. Guidelines are therefore more specific and lower-level; design principles are higher-level, more general, and frequently derived from design guidelines themselves.
As the first example of design principles, Shneiderman and Plaisant [333, p. 70] suggests the following set of eight “Golden Rules”, or HCI design principles, for interface design:

1. Strive for consistency. 5. Prevent errors.
2. Cater to universal usability. 6. Permit easy reversal of actions.
3. Offer informative feedback. 7. Support internal locus of control.

**Figure 10.50:** Interface design principles (“Golden Rules”) by Shneiderman and Plaisant [333, p. 70].

As can be seen from this example, design “principles” are of wider and more general nature than design “guidelines”, several examples of which were given in Section 10.2.

As another example of design principles, Jennings [175] proposes a set of ten “guiding principles” for designing software tools that support creative problem solving in expert domains, arrived at by conducting literature review and interactive prototype development:

1. Make partial solutions first-class entities.
2. Support problem and solution matching.
3. Allow subjectivity and ambiguity.
4. Prefer general, flexible actions and representations.
5. Engage multiple ways of doing and thinking.
7. Encourage parallel exploration of breadth and depth.
8. Provide rich history mechanism.
9. Assist the construction and use of repertoires.
10. Create an effective environment.
Smith et al [345] provides seven “lessons” that can be applied to the design of computational tools supporting multiple “ideas”, however due to the high-level nature of said lessons they should more properly be understood as a set of design principles:

1. Make it really easy to switch between ideas.
2. Provide an efficient way to view multiple ideas at once.
3. Allow users to adapt the interface to their preferences.
4. Provide ways to identify ideas pictorially and textually.
5. Provide multiple ways to group and classify ideas.
6. Provide an explicit means for capturing the situation.
7. Support fluid composition and decomposition of ideas.

As another example of design principles, Nielsen [269] proposes a set of ten “usability heuristics” for user interface design (Figure 10.53).

1. Make system status visible, so that the users knows what is going on at all times.
2. Express the user interface in terms of user’s problem domain, instead of in terms of system concepts.
3. Support user control and freedom through features such as undo, redo, and leave unwanted state.
4. Make the interface consistent, both throughout the tool, and with platform conventions.
5. Prevent errors through careful interface design instead of letting them occur.
6. Minimize the users’ memory load by making objects, actions, and options visible, instead of forcing them to recall information.
7. Allow interaction modalities for all levels of users (novices, average users, experts), by offering e.g., menus for novices and keyboard shortcut accelerators for experts.

8. Keep interface design aesthetically pleasing and minimalist.

9. Help users recognize, diagnose, and recover from errors, for instance by offering error messages which use plain language, indicate the problem, and suggest a solution.

10. An interface should be sufficiently intuitive to be used without documentation, however it may be necessary to provide help and documentation which should be easy to search, focused on the user’s task, list concrete steps to be carried out, and not be too large.

**Figure 10.53:** Ten “usability heuristics” for user interface design by Nielsen [269].

Based on the proposed set of design guidelines (Section 10.2), and from personal experience acquired while conducting the totality of work complete so far, I now propose the following set of design principles for specifying and heuristically evaluating alternatives-enabled pCAD tools:

### Design principles for multi-state pCAD systems:

- **DP-1.** Pay Utmost Attention to Methods for Visual Arrangement.
- **DP-2.** Constantly Seek Opportunities to Parallelize.
- **DP-3.** Think “Parts”.
- **DP-4.** Honour and Integrate Existing (Single-State) IxD Knowledge.
- **DP-5.** Enable Gentle Transition to Multi-State Systems.
- **DP-6.** Think “Large Displays”.
- **DP-7.** Make Ample Use of Animated State Transitions.

**Figure 10.54:** High-level design principles for alternatives-enabled pCAD tools, as proposed in this thesis.

I will now detail each of these design principles in turn.
10.3. DESIGN PRINCIPLES FOR ALTERNATIVES-ENABLED PCAD TOOLS

10.3.1 DP-1: Pay Utmost Attention to Methods for Visual Arrangement

Evaluation studies conducted in this thesis, as well as related work, indicate that the ability to arrange multiple objects on screen is, by far, the most important functionality of tools supporting alternatives-enabled work. This fact simply cannot be overstated, and am therefore dedicating it the very first spot in the list of design principles, in order to clearly indicate its overall importance for this class of tools. To the arrangement ability, I hereby also add the related express abilities, since they likewise deal with a visual representation of an object.

For example, in CAMBRIA 1 study, the largest proportion of the study participants’ feedback was due to the ability to easily and quickly arrange thumbnails of alternatives in the “arrangement” view; in addition, this feedback has overwhelmingly been a positive one. Much of related work (Chapter 5), likewise, corroborates the users’ requirement or need to be able to view and arrange multiple objects at once. Such need appears to be universal, and thus fundamental.

In specific terms, multi-state systems must offer powerful, effective, varied means to visually arrange, lay out, or present multiple computational objects on screen at once i.e., simultaneously. While I presented a number of arrangement methods suitable for multi-state tools in Section 10.2.3, undoubtedly many more are possible, and in fact desirable due to the numerous benefits of visual comparison described in the literature and that takes advantage of the highly parallel human visual system.

10.3.2 DP-2: Constantly Seek Opportunities to Parallelize

This design principle is narrowly related to the previous one (§ 10.3.1), but now extended in a wider, more fundamental sense. To start with, arrange for viewing already implies a degree of parallelism, since the user views and works with multiple design objects in parallel i.e., simultaneously. However, one can and should go further than that. The ability to view multiple objects at once is just one task that can be “parallelized”, in a sense; I thus here make a stronger claim, and say that other tasks, entities and processes can and should be parallelized as well.

In this dissertation, we have seen a number of parallelized approaches. For instance, the techniques for searching (§ 10.2.1.1), previewing (§ 10.2.1.2), and selecting items (§ 10.2.6.1) have been "parallelized" for use in multi-state systems; likewise, the navigational techniques of panning (§ 10.2.4.1), zooming § 10.2.4.2, examining (§ 10.2.5), and many editing/modification techniques (§ 10.2.6.3). I would argue, however, that this merely scratches the surface, and that many more opportunities to “parallelize” various interaction techniques would present themselves would one continue with this research further.

The opportunities for parallelization are not limited to interaction techniques only, and they extend to the underlying data structures, as well as algorithms that operate on such data structures. This aspect of multi-state tools, therefore, pertains to architectural (i.e., implementation-related)
issues. In this work, I did not pursue opportunities to implement parallel data structures nor parallel data algorithms. However, based on my previous experience as well as study of parallel computing, possible approaches include but are not limited to: multiple computational processes being executed in parallel; distributed computational based on message passing mechanisms; shared and distributed memories; different parallel computing paradigms that provide lesser or greater degrees of parallelism (e.g., cluster, grid, and cloud computing); new general-purpose programming languages suitable for developing highly-concurrent and scalable applications (e.g., Erlang, Scala); specialized API libraries for established programming languages (e.g., AMP, MPI or Boost.Thread for C++); and GPGPU (General-purpose Graphics Processing Unit) programming languages such as OpenCL and CUDA taking advantage of massive parallelism afforded by the large number of computing “cores” (also called “shaders” or “stream processors”) available on modern computer graphics cards\(^7\).

To conclude, then, the purpose of this particular design principle is to compel the interaction designer (or system architect) to decisively shift his or her mindset from the well-established ways and best practices for developing single-state tools, to a mindset where the designer looks at every potential opportunity to parallelize any and all aspects of said tools, to the most reasonable or possible extent possible. This applies but is not limited to: visualization techniques; interaction techniques; meaningful and careful integration of both visualization and interaction techniques in order to come up with optimal ways to work with multiple alternatives; and effective implementation methods which in most cases implies a resolute, final orientation towards parallel computing approaches.

**10.3.3 DP-3: Think “Parts”**

Designs are rarely if ever monolithic or “flat” (i.e., non-hierarchical or thoroughly uniform); in most instances, a design possesses a certain inner structure, characterized in terms of modules or parts of said designs, which are then composed and combined together into a hierarchy or assembly of said parts or modules. For example, within the context of parametric design, the global DAG can be considered to be composed of (and to naturally induce the set of) its constituent sub-graphs; in addition, many commercial pCAD packages allow their users to build a model using various kinds of components which are actually self-contained parametric graphs in and by themselves.

The importance of being able to work with parts or modules of a solution is significant in

\(^7\)For instance, one of the most powerful commercially available graphics processing units (or graphics cards) at the time of this writing, features 4096 stream processors, resulting in the total of 8.2 TFLOPs of single-precision compute power. To illustrate this order of magnitude, if one were to perform the simple task of adding two numbers every second by hand, one would need approximately 260,000 years for what a 8.2 TFLOPs computing device can achieve in just one second. This computing power translates directly into the capability to maintain multiple computational models (i.e., design alternatives) in a consistent state, and at interactive rates, for model sizes and model complexities much greater than the ones typically used in the past.
10.3. DESIGN PRINCIPLES FOR ALTERNATIVES-ENABLED PCAD TOOLS

Design exploration; see e.g., [221, 367, 416]. In addition, several experts in the CAMBRIA 1 formative study mentioned the need to manipulate parts of design solutions as well: see Section 8.2.1.1 describing the participants’ feedback on this specific, parts- or modularity-related aspect of design work in general, and interaction with CAMBRIA 1 in particular. While CAMBRIA 2 does not offer part-related capabilities, the exposition on page 272 suggests one way to work with design alternatives at the part level, with a “coverflow” visualization strip offering to the user to substitute parts “in-place”, possibly for multiple design alternatives in parallel.

In other words, this design principle serves to warn about, and direct the interaction designer towards addressing this obviously very important need and requirement for working with parts of a design model. This in turn opens up another, rather important “lateral” dimension to alternatives, and which can be considered orthogonal to all the remaining aspects of designing. In concrete terms, this means that alternatives should be available to individual parts of the design as well, and not only to the global (“master”) design or model.

10.3.4 DP-4: Honour and Integrate Existing (Single-State) IxD Knowledge

Despite the drive towards multi-state tools and systems, as espoused and championed by this dissertation as well as by some of the related work, the legacy of single-state systems must still be honoured. Related to the previous principle of “think parallel” (§ 10.3.5), since I consider that multi-state systems by definition also include single-state systems (i.e., the special case of design multiplicity $m = 1$), it follows that all the existing design guidelines, principles, best practices, and theories developed for single-state systems, must by induction be honoured in multi-state systems as well.

In practical terms, this means that for example design guidelines for attracting user’s attention [403][333, p. 60], design guidelines for the use of colour in user interfaces [333, p. 469], graphical interface design principles (“Golden Rules”) by Shneiderman and Plaisant [333, p. 70], system design principles for tools that support “creative” problem solving [175], and the ten “usability heuristics” for user interface design by Nielsen [269] must still be observed when designing, specifying, and evaluating multi-state tools as well. In this thesis, of course, I went a step further, and proposed a wider, more general set of design guidelines and principles that hold for tools for which the number of design states $m$ can be greater than one. Yet, the body of IxD knowledge pertaining to single-state systems, verified and validated over several decades of meticulous research, must still be taken into account when designing alternatives-enabled systems. The interaction designer must familiarize himself with this knowledge, assess how it fits within the general context of designing, implementing, and evaluating multi-state systems, and then proceed accordingly.

---

8See the discussion presented in Section 4.3.1, Chapter 4.
10.3.5 **DP-5: Enable Gentle Transition to Multi-State Systems**

This design principle is narrowly related to the principle of honouring existing, single-state IxD base of knowledge (§ 10.3.4), only now viewed from the perspective of end users, instead of from that of designers-researchers. Single-state systems, namely, have a long history of successful everyday use and have been, by all accounts, tremendously successful, useful and effective in practice.

This immense pool of accumulated knowledge, expertise, and tried-and-tested workflows associated with single-state systems, and acquired over decades of intensive use, cannot be ignored or cast aside. In other words, users are likely to have strong expectations of how computational tools should “work” and behave, justified by decades of successful use. A novel tool that suddenly goes against these expectations and standards, and places completely new and unusual demands on the user, would in most probability be condemned to failure. This is, in the nutshell, undesirable, since as I have already explained in § 3.2.4 on page 58, the ultimate evidence that an interaction design is really successful is when and if it has achieved wide acceptance and adoption by its intended user community. While in academic research one is considered to be able and sometimes even expected to conduct “pure” research just for the sake of research, the interaction designer-researcher should still aim for at least some degree of target user community acceptance of the tool, or class of tools, that he is designing.

Accordingly, within the context of multi-state system, this in particular means that the tool should probably adopt as many single-state conventions and standards as possible, in order to avoid unpleasant surprises or excessive demands on the user, of course considering their overall suitability for multi-state work. In addition, this means that novel concepts and interaction techniques, no matter how effective, should be introduced gradually and with a good rationale for their intended use. Finally, as already noted by some researchers of multi-state systems, the users should not be distracted from their core work activity.

In practical terms, this means that users of multi-state tools should be able to get up and running quickly, perhaps initially only using features familiar from single-state tools, and then gradually transition to, and adopt, multi-state workflows. For instance, the user should be able to maximize and work on a single alternative, while initially using the alternatives-related features (such as e.g., collections) as a kind of “improved file management” or “revision control” only, and then gradually transition to full-on use of collections as envisioned by the approaches described in this work. As an illustrative example, evidence from the CAMBRIA 1 formative study (see Chapter 8, Section 8.2.1.4) shows that three out of six participants preferred to embark on episodes of solo-edit-solo-edit... sequences involving just one alternative at a time, within a collection, thus clearly following a workflow familiar from conventional, single-state systems.

---

9However, this design principle is still intended for interaction designers.
To summarize, this design principle should compel the interaction designer to integrate existing knowledge, expectations, and standards into novel multi-state design to the maximum extent possible, thus making the transition to multi-state systems as painless for end users as possible.

10.3.6 DP-6: Think “Large Displays”

Multi-state computational tools lend themselves naturally to viewing, and working with, multiple design states at once. Hence, such tools will naturally find their best use with large display surfaces, allowing the user to work with multiple states without having to perform excessive navigation operations. In other words, navigation techniques such as scrolling, panning, zooming and detail+context techniques, even when adjusted for multi-state use (I enumerate such techniques in § 10.2.4), will surely lead to longer task completion times, and to worse visibility and viscosity scores along the tool’s cognitive dimensions (see § 9.3), as compared to being able to view and work with many more design states on a larger display, thus eschewing such excessive navigation and search activities.

Therefore, large display surfaces should be considered at all times during the interaction design of multi-state systems. I already mentioned one such large display surface, Surface Hub by Microsoft Inc., at the beginning of this dissertation (Figure 1.4 on page 4). Other large display technologies and approaches are of course being constantly refined and made better as we speak. At the time of this writing, early adopters of advanced displays technologies are slowly setting their sights on so-called “4K” computer monitors, which typically offer display sizes of 28-32” along the diagonal (instead of currently common 20-24”), as well as display resolutions in the order of 4000×2000 pixels (instead of currently prevalent ≈2000×1000 pixels), thus being able to view up to four times more information per monitor. In addition, some users nowadays also opt for multi-display configurations, such as 2×1, 3×1, or 3×2 configurations. Figure 10.55 shows such a 3×2 multi-monitor setup, built using a set of six 24” monitors.

Considering the relentless and continuous technological progress in the area of computer displays, and if I were to extrapolate a bit, one can expect very large (wall-sized), flexible, and ubiquitous display areas in not so distant future. Certainly, this represents the perfect and ideal scenario for the class of computational tools developed in this work.

Large screen displays, however, give rise to an entirely separate area of research upon which I have not touched upon in this thesis; exploring the ramifications of designing and using multi-state tools in this context certainly represents a potentially fruitful area of future search\(^\text{10}\). To conclude, this design principle compels the interaction designer-researcher to investigate, keep track of, and incorporate latest developments in the area of large displays, in order to inform the interaction

\(^{10}\)I also mention large displays, and list some ideas, in the concluding chapter of this dissertation, see § 12.4 on page 315 describing future research.
10.3.7 DP-7: Make Ample Use of Animated State Transitions

By their very nature, multi-state tools help users to work efficiently with *multiple* objects or states. One can make an even stronger claim, and state that multi-state tools can potentially help users to work with *many*, and not just *multiple*, objects simultaneously visible on the screen, possibly ranging into hundreds or even thousands. In this scenario, however, one action could potentially result in multiple alterations scattered across multiple currently visible design alternatives, leading to a situation where the user loses track of what has just transpired on screen, thus possibly leading to other detrimental effects such as the loss of psychological “flow” [89].

As an illustrative example, performing an “subjunctive snap” operation (p. 275) sometimes leads to surprising and unexpected outcomes, since that operation is dependent on the identities of items, not their geometric locations. Similarly, “passing items” or “passing values” sometimes lead to outcomes which are difficult to envision initially i.e., lead to designs which differ significantly and unexpectedly, from their originating design states. Likewise, in the CAMBRIA 1 study, one participant opined that parallel editing is cognitively very demanding and that even induces “vertigo” [P4 35:13][P4 39:52], thus indicating the degree of difficulty to effectively track the changes occurring simultaneously in several alternatives at once; two other participants likewise stated that maintaining linkage among items, either manually or automatically, would probably be too demanding [P1 52:23][P1 53:34][P4 28:23][P4 31:01]. In the CAMBRIA 2 case study, my dissatisfaction...
with abrupt changes, as I was alternating between different arrangements of alternatives, led to my experimenting with animated transitions among different arrangements, and to corresponding implementations of the same.

Taking these considerations into account, I thus posit that to avoid abrupt, unexpected state transitions which interfere with the user’s understanding and apprehension of what has just transpired on screen, the interaction designer should use animated state transitions wherever possible. The benefits of animation are well known from the literature. In the first place, and most relevant to this work, it has been determined that animation can aid users to better understand a state change [34, 382]. Furthermore, animations can (1) attract user’s attention and therefore help him direct his focus to some point of interest, (2) naturally convey transformations of an object, (3) help communicate cause-and-effect relationships, and finally (4) be emotionally engaging and thus lead to higher satisfaction rates associated with the tool [153]. And in fact, these points are corroborated by my own experience while building and evaluating CAMBRIA 2: I was able to better understand arrangement transitions; they were emotionally engaging; and led to my own higher satisfaction with the tool.

To conclude, this design principle compels the interaction designer to use animation wherever possible, since multi-state tools are by their nature replete with frequent, visually complex, and cognitively demanding state transitions, in order to help the user understand how a change came about, and thus prevent his disorientation and loss of context.

10.4 Chapter Summary

This chapter presented a consolidation of all the previous stages of research, i.e., integration of all the previously generated bodies of knowledge (mostly arrived at through RtD activities) into several higher-level bodies of knowledge that I arrived at through the cognitive processes of introspection and reflection.

Following Frayling [121], the entirety of work described in this chapter can therefore be considered an instance of research-for-design (RfD) research, as distinguished from research-through-design (RtD) research conducted in Phases A–D. The goal of RfD is to generate new knowledge that can help designers reframe (and thus help them solve) the problems they commonly face in their work.

Accordingly, the higher-level, consolidated bodies of RfD knowledge presented in this chapter can help interaction design researchers and practitioners to design, specify, and heuristically evaluate future alternatives-enabled pCAD tools, and include: (1) a classification of interaction primitives, (2) a set of design guidelines, and (3) a set of design principles.
Chapter 11

Epilogue: The Process

As a form of epilogue, in this chapter I describe the _IxD research process_ that I followed in order to generate and evaluate all the research artifacts produced in this work. I noticed, experienced, and observed many different aspects of the research process. I feel, however, that the most important ones include:

1. **Knowledge aspects (§ 11.1).** Any instance of research process is geared towards the creation of new, and updating of existing IxD knowledge, thus advancing the so-called “state-of-art” in the field. Important areas (or scopes, types) of knowledge produced in IxD research process include: schools of thought in design inquiry; methodological IxD knowledge; the hierarchy of various bodies of IxD knowledge; and IxD rationale knowledge.

2. **Timeline aspects (§ 11.2).** Any research process necessarily proceeds (i.e., unfolds, develops, and evolves) along the chronological dimension, and is in addition a time-delimited endeavour. Timeline aspects include: preparing the IxD student or research-novice before the onset of research activities; common IxD research phases, and their sequential arrangement; and since interactive tools are such a prominent part of IxD research, software development lifecycles.

3. **Cognition aspects (§ 11.3).** The researcher involved in, or going through a particular IxD research process, necessarily exercises various _cognitive activities_. These include: cognitively traversing the “space” of IxD designs; updating and improving one’s IxD expertise (or bodies of IxD knowledge, or personal theories-in-use); conversing reflectively with the “problematic” situation of developing and evaluating feasible IxD designs; decision making; and learning activities.

4. **Personality aspects of the researcher-designer (§ 11.4).** Any researcher or student going through an IxD research process is characterized by a large number of personal characteristics, which influence how he or she will perform in the process. Personality aspects include:
11.1 KNOWLEDGE-RELATED ASPECTS OF THE RESEARCH PROCESS

the individual’s motivation, self-motivation, and morale; the individual’s learning styles and natural inclinations; personality traits (i.e., personal profile) in general; and private (i.e., non-work) circumstances of the individual that likewise can influence the execution of the IxD research process.

(5) Organizational (people, teamwork) aspects (§ 11.5). Finally, no IxD research process takes place in isolation, and is in fact embedded into a wider organizational context, consisting of other individuals belonging to an organization and various socio-technical systems. Important organizational aspects include: communication among the members of the organization; knowledge sharing among the members of the organization; organizational learning and organizational culture; the student-supervisor relationship; team roles; and workplace conditions.

11.1 Knowledge-Related Aspects of the Research Process

I adopt the position that as the project moves forward, the knowledge that the researcher “knows” gets developed, refined, built-up, increased in scope, and increasingly more differentiated in terms of details and structure. Thus, in a way, the researcher will obtain “better expertise” as he or she proceeds with the research.

11.1.1 IxD Research “Expertise”

In the following, I interchangeably use the following terms to denote the totality of IxD knowledge that the researcher/PhD student accrues along the way: expertise, mental maps, mental models, and theories-in-action. Expertise denotes the set of skills that the student will possess at the end of the research project. Likewise, mental maps and mental models denote bodies of knowledge that help the student navigate the complex activities of design and research. Argyris and Schön [24, 20, 23, 22, 21] write about multiple theories-of-action that any individual holds in his mind, and which direct his behaviour in different areas of life and work, and are thus to be understood as a type of expertise as well.

I depict such bodies of knowledge acquired by the researcher in an intuitive fashion as a “pyramid” of various bodies of IxD knowledge, as previously explained in Chapter 3, as well as as explained in some detail in the next section, Section 11.1.2. As shown in Figure 11.1 (left), at the beginning of doctoral studies, the student’s knowledge is scant and undifferentiated, as well as marred by errors or misconceptions [292, 346, 185]. Fuzzy objects in the pyramid denote, in an intuitive fashion, the existence of vague and undifferentiated bodies of knowledge at the outset of the doctoral process. To indicate errors and misconceptions, I use red squares with the inscribed letter “E”. At the end of doctoral studies, student’s acquired bodies of IxD knowledge should be well-differentiated, detailed, and with misconceptions trimmed to a minimum.
11.1.2 The Hierarchy of IxD Knowledge

In this thesis, I claim that bodies of knowledge typically generated in IxD research can be classified in terms of their hierarchical rank, order of derivation, and taxonomic inclusion. I thus synthesize and provide a model, or “mental map” of aforementioned bodies of IxD knowledge. Chapter 3 already provided a full justification for this model; in this section, I detail the model further.

Figure 11.2: The three-level hierarchical model of design knowledge: a methodological “mental map”.

Figure 11.2 provides another view of this hierarchical model of IxD design knowledge first mentioned in Chapter 3 on page 45, at a somewhat higher degree of abstraction. Roughly, design knowledge can be categorized into three hierarchical levels: (1) low-level design knowledge (such as e.g., concepts from the domain vocabulary), (2) mid-level design knowledge (design artifacts developed through RtD approach), and (3) high-level design knowledge (such as design guidelines, and design patterns).

Figure 11.3 below, on the other hand, provides a concrete “instantiation” of this abstract model relative to this thesis, i.e., showing some specific bodies of knowledge generated in this work...
11.1. KNOWLEDGE-RELATED ASPECTS OF THE RESEARCH PROCESS

(such as CAMBRIA 1 and 2, design guidelines, and design principles), as well as a number of other pertinent details.

According to the figure, in this thesis I developed a number of bodies of knowledge in the bottom-up fashion: low-level bodies such as e.g., the domain vocabulary; mid-level ones such as various design artifacts; and high-level ones such as the classification of interaction primitives, the set of design guidelines, the set of design principles, and the description of the research process. Additionally, Figure 11.3 shows in an intuitive fashion how those bodies of knowledge influenced each other. While I will go into some detail on how the IxD researcher goes about generating and modifying those bodies of knowledge in Section 11.3 (page 296), the figure also clearly demonstrates that:

1. Bottom-up direction implies *emergence* or *construction* of design knowledge (the left arrow in...
(2) Top-bottom direction implies *corrections* of design knowledge (right arrow in Figure 11.3).

Looking at this model of IxD knowledge, one can easily discern among the artifacts typically produced in a design research project, along with their hierarchical levels, lineage, mutual influence, and relationship of taxonomic inclusion\(^1\). It is my claim that such a map of IxD knowledge, as one major benefit of the model, will greatly reduce *methodological disorientation* for any incoming graduate IxD student, or interaction design researcher-novice.

### 11.1.3 Methodological IxD Knowledge

However, in IxD research, beyond the project-specific knowledge described in § 11.1.2, other important bodies of IxD knowledge exist as well. One such body of knowledge is the *methodological IxD knowledge*\(^2\).

Any research is undertaken by adopting a certain *research methodology*, *research strategy*, or *research approach* \([173, 88, 327]\). I described the related issues, as well as the methodological approach adopted for this thesis, in Chapter 3. Methodological issues are therefore intimately concerned with how to elaborate a systematic, precise “plan of action” while trying to solve any given IxD research problem; such a plan of action can be considered to belong to the “know-how” or procedural type of knowledge (see Section 3.1 on the types of knowledge).

In the field of IxD, a bewildering number of valid methodological approaches exist, reflecting the history of HCI and related fields, as well as the wide variety of professions involved throughout the history of HCI, interaction design research, and most recently “interaction science” \([73, 285]\). For example, due to this profusion of possible approaches and perspectives, Fallman and Stolterman \([114]\) write about the current “disciplinary anxiety” in the field of IxD research. They acknowledge the “substantial latitude in terms of acknowledged research methods and approaches, ranging from quantitative empirical lab studies and qualitative observational studies to research-through-design and critical design experiments”. This richness and sheer number of possible approaches contributes to the overall uncertainty, field “anxiety”, and fragmentation into partisan communities favouring only a certain set of methods. A researcher, especially a junior one, thus might not understand the purpose of research, or “see the forest for the trees”, if not in the possession of the mental map of this particular type of knowledge.

---

\(^1\)Figure 3.8 on page 48 (Chapter 3) provides an even clearer depiction of the middle layer, showing some meronymic relationships among typical bodies of design knowledge.

\(^2\)The “process description” body of knowledge shown in Figure 11.3 represents one specific, per-project instance of methodological IxD knowledge.
11.1.4 IxD Rationale Knowledge

Another important type of knowledge generated in interaction design research is that of design rationale knowledge, or the total corpus of knowledge related to the interaction design research topic at hand. In this work, IxD rationale knowledge includes the knowledge required to justify interaction design decisions in the context of multi-state tools; for example, each of the design guidelines that I detailed in Section 10.2 is accompanied by a corresponding “design rationale” section, thus explicating this type of knowledge, per family of interaction methods.

Thus, in interaction design, design decisions should be justified in terms of how well the associated implementations (i.e., implemented features) fulfil a set of design requirements [257, 258, 247, 327]. Such justifications can be provided only by an an individual who has acquired substantial expertise (most commonly over a number of years) in interaction design: what works, and why. Novice researchers (including entering doctoral students) lack this knowledge by definition.

11.2 Timeline-Related Aspects of the Research Process

Any research project is by definition a time-limited i.e., temporary endeavour, consisting of events arranged in chronological order, and whose goal is to create research outcomes. As the first important chronological aspect, before the onset of research, the student should be well prepared for design-based research (DBR), because without becoming familiar with design-based research and its many facets, the student might experience disorientation due to the inability to “make sense” [316, 187] of research. Once the student has been adequately prepared, he or she can commence with research understood in the strict sense.

In this dissertation, the overall process that I followed reflects the order of chapters given in this dissertation: pre-design → conceptual design → CAMBRIA 1 → CAMBRIA 2 → consolidation. While these stages were arranged in a linear sequence, this does not mean that I followed them strictly. In fact, I was alternating pragmatically among different activities updating different bodies of knowledge. This more realistic model, reflecting my experience, is shown in Figure 11.4.

For instance, as I was engaged in the development and evaluation of CAMBRIA prototypes, I was occasionally also reflecting, or had a sudden insight or idea, on how to amend the conceptual model. The process shown in Figure 11.4 resembles the model by Sedlmair et al [327], where the interaction designer-researcher likewise proceeds sequentially, but also alternates pragmatically, between the nine activities of learn, winnow, cast, discover, design, implement, deploy, reflect and write, which are further grouped into:

(1) Personal validation activities: learn, winnow, cast,
11.3 Cognition-Related Aspects of the Research Process

Another important perspective of the IxD research process include various aspects related to cognitive activities that I exerted during the process.

As the first cognition-related aspect, that of “traversing” the space of IxD designs, and as previously put forward in Section 2.5, designing can be viewed as the activity of a designer-researcher traversing (i.e., discovering, re-visiting) IxD states in the corresponding “IxD space”, as shown in Figure 11.5(a). This view is compatible with the symbolic information processing (SIP) approach, as famously espoused by Newell and Simon [264], and in which “symbolic structures” representing design states stream through, and are modified within, individual designers’ minds, while using external representations. This activity of “traversing design states” by a designer-researcher then leads to the creation of the global design history, which can be best understood as the set of linked design states, as depicted in Figure 11.5(b). Accordingly, in this work, I traversed the design space of multi-state pCAD tools, as evidenced by the development of multiple design artifacts such as e.g., interactive research prototypes, and the set of design requirements on such tools.

Another important aspect is the chronological maturation of researcher’s IxD knowledge, narrowly related to the bodies of knowledge that I described in § 11.1. One way to view designing is as the process of “construction of design representations” [391], wherein the designer-researcher
11.3. COGNITION-RELATED ASPECTS OF THE RESEARCH PROCESS

Figure 11.5: Designer-researcher “traverses” design states in the IxD design space (a), thus generating design history (b).

pragmatically alternates among the “knowledge updating” activities associated with different bodies of knowledge (or “theories-in-action” [24, 22]), and which I have structured into the hierarchy of IxD knowledge similar to existing, well-established hierarchical models of knowledge [265, 3].

Figure 11.6: Pragmatic alternating of the activities of updating various bodies of design knowledge.

Figure 11.6 depicts yet another view of the hierarchical model of design knowledge, with different bodies of design knowledge, and the associated “knowledge updating” feedback loops (indicated by small, red arrows self-referencing each body) representing a designer-researcher’s knowledge-updating cognitive activity.

According to this model, the researcher-designer pragmatically and contingently alternates among different bodies of knowledge, at a different hierarchical level. In my experience, that is
how I proceeded in this research. As an illustrative example, while designing and coding CAMBRIA 1, I sometimes suddenly realized that some feature $F$ (say, a very specific interaction technique) should definitely be described in the set of design guidelines. I then temporarily stopped developing CAMBRIA 1, and commenced with reflecting on, and writing down, an argumentation of why the feature $F$ should be mentioned in the set of design guidelines. This “interim” episode that disrupted (at that time) the “main” RtD research activity, the one of designing and developing CAMBRIA 1, was therefore temporarily put on hold in order to elaborate and write down new ideas about feature $F$ and its role in a higher-level body of knowledge, in this case the set of design guidelines. Once I finished writing this up, I either returned to my previous activity of developing CAMBRIA 1, or perhaps some other activity (for example, updating the “literature review” research artifact, if I discovered a new, pertinent article). Therefore, to again stress this fact, in my experience the designer-researcher alternates pragmatically among different bodies of design knowledge, placed at different hierarchical levels.

The fact that I embarked on updating the set of design guidelines with the information about feature $F$, although my ideas about the structure and elements of said set of guidelines was at the time unknown to me, points to another important aspect of knowledge-generating cognitive activities: namely, that bodies of knowledge “become clearer” with time. This is echoed, for example, by K. Weick [401], who writes that one important body of knowledge (that of theories), comes into being by a process of theory building in which said theories gradually become “clearer” as if “bringing binoculars into sharper focus”. Figure 11.7 depicts this idea in an intuitive fashion, featuring a clockwork mechanism with increasingly better defined structural relationships linking individual parts of the clockwork, as well as increasingly “sharper” details for each part of the clockwork. Likewise, for any body of knowledge (even non-conventional ones such as models and prototypes, understood as “embodied knowledge” in Frayling’s sense), with passage of time said body becomes increasingly more elaborated, both in terms of its structure, as well as in terms of the richness and completeness of its details.

According to this model, therefore, any body of knowledge, even including the totality of all the knowledge produced in this work, are always at the beginning of any research project just an undifferentiated and fuzzy “blob”, as shown in Figure 11.7(a). Thus, at the outset, we do not know anything about that body of knowledge, what is its structure, and what are the details of its parts; we only know that “it’s there” i.e., that it exists. Only through the extended activities of research, does such a “blob” become clearer, finally reaching an adequate level of clarity, as intuitively shown in Figure 11.7(d), with the clockwork mechanism in acute, sharp focus, and with clearly discernible details and structure.

3 I chose a clockwork mechanism due to its overall coherency, purposefulness, delicate inner structure, and richness of detail; all properties of any good body of knowledge.

4 And, if we go to the extreme, the humanity’s total historical research output.
Figure 11.7: A visual metaphor depicting a progressively “clearer” (i.e., better structured, understood, and comprehended) body of knowledge, closely following K. Weick [401]. Note that both the structure as well as the details of this clockwork become progressively more distinguished and discernible.

In hierarchical models, since upper layers naturally emerge from lower levels of knowledge, this means that lower levels are always, by definition, clearer than upper levels; or, using the opposite direction, higher levels are always more “blurry” (i.e., undifferentiated) than the lower levels. Figure 11.8 shows a sequence of progressively sharper and more elaborated hierarchy of design knowledge, per phases of research listed in § 11.2.

Figure 11.8: Progressive build-up and clarification (“sharpening of focus”) of hierarchical bodies of IxD knowledge developed in this thesis.

Figure 11.8(e) shows a “finished” pyramid of knowledge, i.e., a pyramid containing all the bodies of knowledge required at the outset of a research project; in addition, as shown in the figure, all bodies contained therein are sufficiently “clear”\(^5\). To further illustrate this gradual development of bodies of knowledge, per hierarchical level and per phases of research:

(a) Pre-design. In the pre-design phase, at the very outset of this doctoral research project, I knew that I had to study and compile references of related work, as well as define important, relevant terms in the domain vocabulary (low-level knowledge layer). I also knew that some prototype development activities will be involved as well (mid-level layer).

And indeed, Figure 11.8(a) shows a number of points in the low-level layer, whose exact

\(^5\)Since this dissertation is nearing its end, logically the pyramid in Figure 11.8(e) realistically shows the state of affairs for this research, perhaps maybe with the exception of the high-level body of knowledge “IxD method” that I am currently describing in this very section. Thus, only the top circle in Figure 11.8(e) should be slightly blurry, but will become clear, and completely differentiated in terms of its inner structure and level of detail, at the end of this chapter.
number and scope were still being worked out, and are thus relatively “blurry”. The mid-level layer is even blurrier, due to the fact that it still wasn’t clear to me which design artifacts will have to be developed and how many; in addition, another important design artifact (the set of design requirements) was yet to be declared as one of the major contributions of this work in subsequent phases.

Thus, any ideas about high-level knowledge artifacts weren’t even being planned at that point, which is again reflected in Figure 11.8(a), showing no indication of high-level bodies of knowledge.

(b) **Conceptual design.** This phase further clarified low-level and mid-level bodies of knowledge, as well as provided first hints (and awareness) of the need to work on at least two high-level bodies of knowledge (design guidelines, and the custom IxD research method followed), as shown in Figure 11.8(b).

(c), (d) **CAMBRIA 1 and 2.** Marked by a rather detailed (but still incomplete) low-level knowledge, and a number of well-defined points in the mid-level knowledge (GUI mockups, throwaway prototypes), as well as further differentiation and expansion of high-level knowledge (in particular, the growing set of design guidelines), as shown in Figures 11.8(c) and (d).

(e) **Consolidation.** Realization that one principal high-level body of knowledge (the one of design guidelines) should be further differentiated into the following three: the classification of interaction primitives, the design guidelines themselves, and the high-level design principles. This realization fed back into a low-level body of knowledge, the one of literature review, because I realized that I have to add another section on *interaction taxonomies* into the corresponding chapter, in order to come up with my own classification of interaction primitives for alternatives-enabled pCAD tools, onto which I scaffolded my set of design guidelines.

From knowledge-generating cognitive activities, I now move on to the “reflective practice” activities proposed by D. Schön, as a significant off-shoot of SIT, or *situated design* [360]. Reacting to the positivist doctrine underlying the Simon’s “science of design” initiative [342, 343], and elaborated in his own book “*The reflective practitioner: How professionals think in action*” [323], Schön claims that professional practice (including the practice of design) cannot be best regarded as an objective, detached search in some “problem space”, but as a constructivist, reflective, intuitive and artistic “*conversation with the materials of a situation*”\(^6\).

---

\(^6\)It should be noted however that some proponents of positivist approaches, including H. Simon himself with his *symbolic information processing* (SIP) approach, believed that SIT can be interpreted in SIP terms, see for example [390].
This reflective “conversation” with (the materials of) situation can be depicted diagrammatically as shown in Figure 11.9, whereby the designer loops through the following cognitive activities:

1. **Make a move** within a situation (which then influences and/or shapes that same situation),

2. **Analyze the move**, and then, based on this analysis,

3. **Reframe/set the problematic situation** by continuously:
   
   (a) **Naming** things present in the situation, and
   
   (b) **Imposing** the discipline (i.e., a frame) on those things.

Each design move can be regarded as a hypothesis or as a “local experiment which contributes to the global experiment of reframing the problem” [323, p.94]. The designer makes the move, and then evaluates the consequences of that move. These consequences can be desirable or not, and can conform to (or violate) expectations created by previous moves. In addition, a move can created new opportunities (or potentials, or problems) in the on-going design work [323, p.101]. Furthermore, the designer-researcher thinks, or reflects, as the designing takes place, thus engaging in “reflection-in-action”. A designer-researcher also sometimes stops reflecting-in-action, and observes the effects and consequences of his previous moves, thus engaging in “reflection-on-action”. Likewise, in this thesis, I frequently recurred to reflection-in-action, as well as in reflection-on-action. For example, the process of developing interactive prototypes proceeded in an organic fashion, sometimes resulting in unexpected computational behaviours (either in the positive or
negative sense), further influencing subsequent development, and compelling me to stop for a while and “make sense” of said computational behaviours, and thus reframe the problem.

Another important cognition-related aspect of the process that I followed, is that of decision-making activities undertaken during the research project. For this analysis, and in order to interpret my experience, I decided to lean heavily on the recent decision-making model proposed by John Boyd [47], according to which some aspects of human cognition can be modelled by the following four processes: (i) observe, (ii) orient, (iii) decide, and (iv) act. The initial letters of these four cognitive activities thus lend its name to the model; the “OODA” decision making loop. Figure 11.10 shows a concise diagrammatic depiction of the cognitive activities present in an OODA loop.

![Figure 11.10: OODA loop with its four main cognitive processes: observe, orient, decide, and act. Figure created after [383].](image)

According to this model, a decision-maker first observes an ever-changing situation, and then tries to make sense or understand his observations through the “orient” activity. Once this sense-making or orienting activity is complete, the decision-maker decides what to do next, which can lead to many different outcomes including: (1) reverting to the “observe” activity, (2) perform additional “orienting” or sense-making of collected observations, or (3) act, thus affecting the situation and all subsequent observations of the situation. Finally, from exerting an action, the decision-maker goes back to the “observe” activity.

The model has been applied with success to many different activities, such as winning aerial battles, product development, and design. In the words of David G. Ullman, a renowned design researcher, OODA is a “succinct representation of the natural decision cycle seen in every context: war, business, product development, or life” [383]. One single OODA loop describes
decision-making for a single task, and for a single decision-maker. Since projects are in general undertaken by a team consisting of many members, and since there is a multitude of different tasks to be executed, a project generally has many interacting and nested OODA loops. The four main cognitive activities in the OODA loop are:

1. **Observe.** Through this activity a decision-maker collects observations of the evolving, “problematic” situation. The set of collected observations is always evolving, inconsistent, uncertain, incomplete, and decision-maker dependent.8

2. **Orient.** Arguably the pivotal, most important activity in the OODA loop, that might also be called “understand”, or “make sense” of the problematic situation.

   The output of this activity is the situational understanding i.e., a set of oriented observations. As per Boyd [47], this situational understanding is dependent on, and strongly influenced by, decision-maker’s (1) past experiences, (2) cultural heritage, (3) genetic heritage, (4) skills and abilities to analyze and synthesize information, and (5) new information coming into this activity at the current iteration of the OODA loop.

3. **Decide.** Once the decision-maker has achieved a situational understanding in the previous, orient step, he or she must now decide on the next course of action to be undertaken by the decision-maker. The decision-maker might decide to either (1) observe more (i.e., return to the initial step observe), (2) orient further (i.e., return to the previous step of orient), (3) take a concrete action (see the next stage, the act activity), (4) involve more people, (5) turn attention to other OODA loops, or (6) make other decisions, as needed.

4. **Act.** Denotes undertaking the action decided on in the preceding activity decide. An action affects the evolving situation and therefore all future observations of it.

   Here I point out to the reader the extraordinary similarity of OODA model with the Schön’s model of reflective practice. And indeed, with its emphasis on observing the ever-developing, “problematic” situation, OODA bears a certain similarity to the “situated action” perspective on design. And vice versa, within the problem-solving paradigm, one can of course also view OODA loop as a decision-making cycle which leads the problem-solver from the current, “problematic” situation (i.e., state in the world) to a desired, “goal” situation in the world, thus honouring all three working definitions of designing (see § 2.1) given in this thesis.

---

7 Another advantage of the OODA iterative model is the fact that it easily pinpoints and explains various hindrances or potential pitfalls that might slow down or even completely stall the decision-making process, thus leading to a delay in the execution of the design research project.

8 That is, dependent on who is doing the observing [383]. In general, the observer and the decision maker might be one person, or two persons. In the latter case, the observer reports to the decision maker.
Finally, as the last cognition-related aspect, any research project, especially projects such as the one described in this thesis (i.e., research conducted as a partial pre-requisite for obtaining an PhD academic degree), is also characterized by a significant educational or learning dimension, and takes place in an educational setting. One influential model of personal learning, the so-called “experiential learning cycle” (ELC) model [195] by D. A. Kolb, describes learning as a process of acquiring experiences.

Figure 11.11: Cognitive processes within the Kolb’s “experiential learning cycle” model for learning [195].

Figure 11.11 depicts ELC diagramatically, with its four main cognitive activities experience, reflect, conceptualize, and experiment, arranged in continuous, uninterrupted loop similar to Schôn’s loop shown in Figure 11.9.

(1) Experience (verb). Denotes having a concrete learning experience, such as attending a class, reading a book, or browsing online materials.

(2) Reflect. Denotes the act of taking time to reflect on, and review, the educational experience, including reflecting alone and in privacy, or discussing the matter with a colleague or mentor.

(3) Conceptualize. Based on previous reflection, denotes the activity of coming up with specific conclusions about the theoretical abstracts related to the learning experienced.

(4) Experiment. Denotes the activity of actively trying out what has been learned in new situations (i.e., test concepts), and then going back to having another learning experience.

Table 11.1 summarizes these four typical learner types, along with the characteristic question asked by each type, as well as their preferred activities.

Any given individual will in general have a tendency towards any of the four cognitive activities (experience, reflect, conceptualize, and experiment). As a rule, then, any individual will possess a different effectiveness in any of the four activities, with certain repercussions and implications on the role(s) that individuals should assume within a research project.
11.4 Personal Aspects of the Research Process

While all process-related aspects are in some way or the other “personal” to any researcher involved in a research project, in this section I enumerate some aspects that apply to, and describe, the researcher’s personality in the strict sense.

Firstly, different individuals are characterized by different (general) personal attributes (behavioural, mental, emotional, and temperamental), which in turn will again affect their performance within a research project.

An individual is well advised to build upon, and improve, his or her particular strengths. In his notable paper “Managing Oneself” [104], P. Drucker, a renowned and influential expert in the subject field of management (personal and organizational), argues that an individual can never be successful by improving his or her weaknesses, but only by improving one’s strengths. A full treatment of personality traits is beyond the scope of this thesis, however one influential and well-known “personality traits model” includes the “Big Five” model, which has been found through empirical studies to be sound [101, 168, 180, 134]. Within this model, individuals can be distinguished among the following five orthogonal dimensions: openness to new experiences; conscientiousness; extraversion; agreeableness; and neuroticism. Other personality aspects include the individual’s age, intelligence, ethnic background, and cultural background. Taken as a whole, then, the personality profile i.e., the set of all personal traits for an individual, and its suitability for certain tasks and organizational cultures is studied within a separate discipline of organizational fit [277, 198].

As another personality aspect, motivation has been shown to be “the single greatest influence on how well people perform” [245], and can be defined as “the feeling of wanting to do something, especially something that involves hard work and effort” [278]; alternatively, as the condition that

<table>
<thead>
<tr>
<th>Learner type:</th>
<th>Typical question:</th>
<th>Preferred activities:</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Experencer”</td>
<td>What if?</td>
<td>synthesizing, summarizing, integrating, evaluating, verifying, explaining</td>
</tr>
<tr>
<td>“Reflector”</td>
<td>Why?</td>
<td>interacting with people, speaking, listening, questioning, observing, visualizing, imagining, inferring, diverging, brainstorming</td>
</tr>
<tr>
<td>Theorist</td>
<td>What?</td>
<td>finding patterns, analyzing, seeing relationships and inter-relationships, identifying parts, ordering, prioritizing, classifying, comparing, organizing</td>
</tr>
</tbody>
</table>
| Experimenter | How?             | tweaking, tinkering, experimenting, exploring, inquiring, problem solving, seeing, predicting, making things work ]

Table 11.1: Learning preferences and inclinations [195, 115].
makes people “moved to do something” [318]. The related concept of self-motivation can be defined as “the quality of being capable of hard work and effort without the need for encouragement” [279]. Motivation has been shown to exert more influence on the work productivity than any other factor, for example, in the fields of software engineering [45, 245, 35], where undermined motivation leads to expensive project failures [215, 373]. The related concept of work morale [245, p. 254] is similar yet still different from motivation, and denotes the current desire to work at a certain job. Morale can be high while the motivation is low, and vice versa.

Finally, as the last yet nonetheless important personal aspect, an individual assumes many personal roles that are not related to the research project underway, but might affect it nonetheless. These private roles and resulting circumstances might be related, for example, to the individual's family and personal relationships, and other circumstances.

11.5 Organizational Aspects of the Research Process

As the last aspect type of the process that I experienced while conducting research described in this dissertation, most research projects, including this doctoral research, are embedded into a wider organizational context, i.e., within the context of groups of individuals who collaborate in order to achieve some common objective. Organizations exist at different scales of complexity and size: institutions, research groups, or just a few people (e.g., the interaction and working as well as personal relationships between a doctoral student and his academic supervisors). I limit this section to an overview of the most pertinent concepts and issues.

The important concept of “organizational learning” refers to the fact that any organization, just like separate individuals, must be able to learn i.e., be able to detect and correct errors taking place within the organization (i.e., within its members), in order to ensure its survival [55]. As shown in Figure 11.12, knowledge is shared among members of an organization through the process of communication. The process of communication is, therefore, of utmost importance for the survival and performance of an organization; it serves to propagate sound knowledge, and decrease defects in knowledge or expertise inherent to each member of the organization.

Argyris and Schöen [23], in their large body of work exploring organizational issues, define organizational learning as a process that “occurs when members of the organization act as learning agents for the organization, by detecting and correcting errors in organizational theory-in-use, and embedding the results of their inquiry in private images and shared maps of organization.” The definition above references an important concept, that of theory-in-use, which can be individual.

---

9The academic discipline of communication studies is fairly extensive and interdisciplinary in scope. Concise surveys and introductions include e.g., [79, 58].

10Again, Figure 11.12 shows this in an intuitive manner, with the knowledge pyramid representing such “images” and “maps”, and red E boxes representing errors or misconceptions in one’s expertise.
(i.e., pertaining to a person), or organizational (shared). The learning of individuals within organizations is either facilitated or inhibited by various governing variables defining the organizational learning culture [19]. The related concept is that of organizational culture, that Denison [97] defines as “the internal social psychological environment of organizations and the relationship of that environment to individual meaning and organizational adaptation” [97, 241]. Simon [340] notes that organizational learning takes place “inside individual human heads” only; an organization can therefore learn only through (1) acquisition of new knowledge (i.e., learning) by its current members, and/or (2) by accepting new members with specialized knowledge that does not currently exist in the organization. This must be qualified by the fact that many organizations nowadays utilize various knowledge management platforms [94], thus allowing the organizational knowledge to reside in various forms of external memory as well.

11.6 Chapter Summary

In this chapter I described the research and development process that I followed, or went through, in order to generate all the major research contributions of this thesis. While any instance of such processes can be characterized by many different and distinct aspects or attributes, the main important ones include:

- Knowledge aspects, describing how bodies of knowledge (or expertise) get generated and validated along the way,
• **Timeline (chronological)** aspects, which describe how research and development evolve along the temporal axis,

• **Cognition** aspects, which cover cognitive activities taking place within an individual's mind, such as learning and traversing of design states,

• **Personality** aspects related to the researcher-designer or graduate student, and

• **Organizational** aspects which likewise impact the process to a significant degree.
Chapter 12

Conclusions

This thesis addresses one specific weakness of most contemporary computational design tools: despite the evidence that designers, when using design media other than computational ones, frequently work with multiple design variants in parallel, most computational tools allow their users to work with just one design model at a time. This shortcoming thus prevents designers from engaging in work processes which have been shown to be typical of, and prevalent in design. To alleviate this shortcoming, in this thesis I contribute new interaction techniques, prototype systems, design guidelines, and design principles for specifying, building, and evaluating computational pCAD tools that enable the user to work with multiple design models in parallel.

12.1 Review of Research Contributions

The research contributions of this thesis are as follows:

(i) A well-documented case arguing for the use of multiple design alternatives. To strengthen the case for multiple design alternatives, as well as to provide a part of the motivation for this work, I provided illustrative examples, cited empirical studies, and described expert opinion lending support to the claim on alternatives-rich design work in Chapter 4. At the end of that chapter, I also proposed a simple conceptual model of designing wherein designers, in general, work with $m$ alternatives at a time. This number $m$, or multiplicity factor, fluctuates during a design project, and ranges from $m = 1$ indicating that the user engages in single-state design work, up to some number $m > 1$ of design alternatives that the user works on in parallel. This simple model helps situate and frame alternatives-enabled computational tools.

(ii) Interactive prototypes of novel alternatives-enabled pCAD tools (Chapters 6, 8, and 9). Having built a good case for multiple design alternatives, I then proceeded to engage in the research-through-design, i.e., developing and evaluating a number of design artifacts, such as GUI mockups, early throwaway prototypes, and higher-fidelity prototypes (CAMBRIA 1
and 2). The initial literature review, and pre-design activities (Chapter 6) provided me with first indications of what might constitute effective approaches to viewing, editing, and managing multiple design alternatives at once in a computational tool, which in turn led me to the development of the conceptual model (Chapter 7) of such tools, followed by the development of CAMBRIA 1 and 2.

(a) CAMBRIA 1 (Chapter 8). Based on the analysis of past approaches, a number of GUI mockups and throwaway interactive prototype, as well as on the conceptual model, I designed, developed, and evaluated the first higher-level prototype CAMBRIA 1. The contributions of CAMBRIA 1 include various novel interaction techniques, such as the ability to group and view multiple designs, as well as to perform several parallelized drawing operations: parallel drawing of vector lines and curves, parallel import of SVG drawings, parallel brushing, parallel select, and parallel modification of graphical items. I conducted a formative study of CAMBRIA 1 based on expert feedback, which resulted in the empirical characterization of might constitute effective alternatives-enabled pCAD tools.

(b) CAMBRIA 2 (Chapter 9). The accumulated experience has subsequently allowed me to design, implement, and evaluate the second high-fidelity interactive prototype, CAMBRIA 2. In particular, this version enables the user to manipulate, arrange and edit drawings directly in the arrangement view. Also, the prototype implements and demonstrates the specific interaction technique of subjunctive snapping, whereby the user can attach (or “snap”) a control point onto a line or point in parallel, for all currently active design alternatives. I evaluated CAMBRIA 2 using the cognitive dimensions framework, which provided additional input into what might constitute effective alternatives-enabled pCAD tools.

(iii) A classification of interaction primitives for alternatives-enabled pCAD tools. I synthesized and presented a classification of interaction primitives (tasks, and interaction methods) for alternatives-enabled pCAD tools in Section 10.1, inspired by several existing taxonomies and typologies of interaction techniques.

(iv) Design guidelines for alternatives-enabled pCAD tools. Building upon the classification of interaction primitives, I then synthesized the set of design guidelines that support specifying, building, and evaluating alternatives-enabled pCAD tools (Section 10.2).

(v) Design principles for alternatives-enabled pCAD tools. Reflecting on the totality of work completed so far, I further proposed a set of higher-level design principles that can support specifying, building, and evaluating alternatives-enabled pCAD tools (see Section 10.3).
A hierarchical model of design knowledge. I synthesized and presented a hierarchical model of design knowledge, i.e., of knowledge produced both in design practice and design research, similar to existing hierarchical models such as the one by Newell [265] and Ackoff [3]. To the best of my knowledge, no similar model has been proposed yet in the design research literature. Figures in this dissertation that show some variant of this model, at different levels of abstraction, include: Fig. 3.5 (p. 45); Fig. 3.8 (p. 48); Fig. 11.1 cast as “expertise” (p. 292); Fig. 11.2 (p. 292); Fig. 11.3 (p. 293); Fig. 11.6 (p. 297); and Fig. 11.8 (p. 299).

![Diagram of hierarchical model of design knowledge]

Figure 12.1: The hierarchical model of design knowledge proposed in this work.

In this model, parts of design knowledge are structured in terms of their stratification level, lineage, mutual influence, and relationship of taxonomic inclusion. I posit that such a hierarchical model of design knowledge will be useful to researchers and practitioners alike in different areas of design, helping them to better situate and understand the bodies of knowledge created in both design practice and research, as well as inform future research related to the epistemological and methodological foundations of design practice and research. Figure 12.1 shows one variant of the model, with design knowledge classified into three hierarchical levels.

(vii) Research and development process followed. I provided a description of the research and development process that I followed in this work, and which builds upon the aforementioned hierarchical model of design knowledge, in Chapter 11.

Among the most salient characteristics of the process, (1) the bodies of design knowledge are constructed in the bottom-up fashion (i.e., one layer of knowledge emerges from the lower one), (2) said bodies of design knowledge become increasingly more detailed with passage of time, both in terms of their inner structure as well as richness of detail (Figure 12.2).

Other aspects of the research process that I covered include: knowledge aspects (§ 11.1),
CHAPTER 12. CONCLUSIONS

Figure 12.2: Progressive build-up and clarification of hierarchical bodies of design knowledge.

timeline (chronology) aspects (§ 11.2), cognition (§ 11.3), personality of the researcher-designer (§ 11.4), and organizational aspects (§ 11.5).

The contributions of this dissertation enumerated above, demonstrate and corroborate high-level bodies of knowledge needed to conceptualize, implement, and evaluate effective computational (pCAD) tools for working with multiple design alternatives in parallel.

12.2 Answering the Research Questions

Closing the full circle, the research contributions listed above answer the research questions posed at the very beginning of this dissertation, as follows:

RQ1 What are the essential system features of novel multi-state pCAD tools that can support design work?

According to the findings of both the expert think-along study, and the case study, a subset of features appears to support designing, especially the ability to view and compare multiple designs at once, as well as arrange them in different ways.

RQ2 What are the interaction design guidelines and principles for specifying, building and evaluating multi-state pCAD tools?

The interaction design guidelines were presented in Section 10.2, and design principles in Section 10.3.

RQ3 What methodological approach can be followed that can help system designers build multi-state pCAD tools?

The methodological approach that I followed was described in Chapters 3 and 11.

12.3 Judging the Success of This Research

As for fulfilling the criteria of “successful” HCI design research adapted to evaluate this work (see Section 3.2.4), I argue that said criteria have been met, with the following rationale:
(1) **Significant invention.** The totality of work described in this thesis represents a significant step towards conceptualizing highly-interactive multi-state pCAD tools. In addition, interactive visualization approaches presented in this work were (1) justified in terms of how well they fulfill the design requirements, while (2) referring to the first (i.e., perceptual) principles.

(2) **Design process description.** I give a full description of the design process that I followed in this work in order to generate all the major research outcomes of this thesis (Chapter 11).

(3) **High relevance.** I claim that the research presented in this thesis is highly-relevant, due to the following: (1) I demonstrate how one can build effective alternatives-enabled pCAD tools, thus potentially opening the path for similar implementations in real-world tools and practices, (2) I provide a set of design guidelines and design principles, thus showing the “preferred state” that the corresponding HCI or interaction designs should ultimately converge to, and (3) I provide a well-reasoned and justified case as of why CAD tools should adopt alternatives-enabled workflows (see Chapter 4).

(4) **High extensibility.** I claim that the set of design guidelines and design principles presented in this work are highly extensible and generalizable to other domains. I support this claim by providing additional details in the section on future work, see Section 12.4.

### 12.4 Limitations and Future Work

Any instance of research suffers from various *limitations* that may or may not have a negative impact on its quality, validity, and applicability, and thus lead to future work that would remove or ameliorate such limitations. In addition, any instance of research naturally introduces new problems, and opens up new vistas as well as directions for future work.

**Limited Number of Implementations**

The number of research artifacts that a single individual can develop within a set time frame is limited. Hence, the completeness of research-through-development is necessarily limited as well, which in turn affects how thoroughly and completely the design space has been explored by the researcher or interaction designer. Despite the dozens of low-fidelity prototypes (see Chapter 6) as well as several high-fidelity prototypes that I developed (Chapters 8 and 9), in ideal case one would develop additional and more feature-complete interactive tools, perhaps through a multi-member development team effort. A more complete tool would likely reveal shortcomings in the current set of design guidelines, as well as enable me to release the tool to a much wider audience, thus potentially collect and analyze much richer and more extensive user feedback.
Incompleteness of Design Guidelines and/or Principles

As per the definition of design problems (Chapter 2) and the definition of design-based research (Chapter 3), design problems are in general never conclusively “done” or categorically “solved”. Thus, most research artifacts produced in design-based research can always be improved on, iterated on, refined, and elaborated further. The sets of design guidelines and principles proposed in this thesis are no exception to this fact, i.e., they can always be improved on, refined, extended, made “better”, more complete, and more fit for a purpose, especially considering the aforementioned limited number of implementations.

It is my hope, therefore, that this thesis will provide inspiration as well as open new lines of inquiry, thus potentially leading to future work detailing and elaborating richer sets of design guidelines and principles in the field of pCAD, as well as any other area of application which could benefit from working with multiple alternatives in parallel.

The Issue of “Small Iterations”

According to CAMBRIA 1 study, the approach to working with multiple design alternatives that I presented in this thesis might lend itself well only to design solutions which require “small iterations”, “refinement”, and “tweaking” only. This particularly applies, if one were to extend this work to other domains, to modifying parts of designs that require tremendous amounts of re-work and tweaking, such as code and prose.

Therefore, the open problem here is how to achieve “large” or “substantial”, yet still valid variations of designs. One way forward could be the substitute-in-place editing technique described on page 272. Future work could investigate the questions of the suitability of the part that is replacing the current part (i.e., an objective measure of compatibility and/or suitability), as well as ways to integrate the new part (for instance, by automatically “repairing” all the constraints relating the new part and the host design), and then automatically validate the newly obtained overall design, thus ensuring its soundness.

Another potentially fruitful approach to achieving more significant design changes would be the application of random perturbation to a part of the design, thus possibly leading to a serendipitous discovery [369] of acceptable designs. Some of the throwaway prototypes developed in this work showcase random perturbation techniques: the 2D font designer (§ 6.2.3.2) and CAMBRIA 1 (Chapter 8).

Modularity of Designs

Related to the issue of small iterations mentioned above, expert feedback from CAMBRIA 1 study indicated the need for working with parts of designs as well. I tried to tackle this issue by proposing
the substitute-in-place technique described above, however investigating how the multi-alternative approach would apply to parts themselves, would probably merit additional, focused research.

Existing literature seems to corroborate the need for this capability. For instance, in [221] Luzner recommends recombining parts of solutions, in order to come up with better hybrid solutions. Terry recommends “standalone alternatives at any level of granularity”[367, p. 162] as well as to treat parts of models as “containers which permit multiple alternatives to a part”[369]. Computational tools supporting multiple “ideas” should allow for “fluid composition and decomposition of ideas”[345]. In the domain of pCAD, this would imply imposing, inducing, or inferring a modular structure, thus allowing the users to work with individual sub-graphs of the global parametric DAG.

**Parallel Editing: Useful or Not?**

While the research described in this dissertation confirmed the high value of viewing and comparing multiple designs at once, further research is needed in order to determine the real value (efficiency, effectiveness) of modifying or editing multiple designs in parallel. The CAMBRIA 1 study provided a mixed feedback on the value of parallel editing, with experts rating it either positively (“it’s interesting”) or negatively (“can induce vertigo”). Indeed, this seems to be corroborated by other researchers’ findings; for example, Terry [367, p. 157] states in the conclusion of his thesis:

> From our study data, the ability to manipulate sets of alternatives simultaneously (i.e., apply an operation to all of them at the same time) seems less vital. Further research may demonstrate when and how this capability may prove most useful.

In this thesis, some of the CAMBRIA 1 study participants engaged in the workflows editing just one design variant at a time. The recent paper by Zaman et al [416] confirms this new pattern of use, with participants requesting that “sandboxing” (i.e., working with just one design at a time) be “the default work mode”, as well as that “parallel editing only be available on demand”, as opposed to the default mode in which participants edit multiple design alternatives in parallel.

In summary, additional studies are required to find out the real utility of parallel editing, and under which conditions. At the moment, however, evidence tentatively points to a limited value of parallel editing, probably due to the higher cognitive investment needed to track all the changes occurring in real time from such operations.

**Large Displays**

As I already presented in some detail in § 10.3.6 on page 287, the interaction designer-research should investigate, and keep track of, the latest developments in the area of large displays:
both technological developments (i.e., the availability or large monitors, wall-sized displays, multi-monitor configurations, and so on), as well as developments in how best to interact with systems featuring such large displays.

Yet, large screen displays bring about a completely fresh set of associated issues and research problems, especially in the context of this work (multi-state tools and systems). For example, how does one interact with a wall-sized display, when users have to locomote i.e., change their location physically in order to reach another part of the display? How does collaboration look like on such large display surfaces?

As just one illustrative example, when they exceed a certain size, displays become “qualitatively different” [363], and interaction design (IxD) transforms into “environmental design”. At least three different configurations are possible for large displays: distant-contiguous, desktop-contiguous, and non-contiguous [363]. Each of these three configurations would represent and induce another set of challenges to multi-state interaction.

In summary, issues such as the ones above and related issues represent a fertile ground for future research, investigating how display size and ubiquitousness affect multi-state interaction.

**Evaluating the Quality of Design Guidelines “In the Field”**

As stated in Chapter 3, the ultimate measure and test of the success of an HCI design is its wide adoption by the intended user community. Yet, since this process can take years, a suitable proxy measure for determining the quality of any given design artifact can be found in the form of a further, deeper, and more extensive evaluation of said artifact, done while involving real users performing typical tasks. In other words, the tool should be tested in the field, also sometimes called “testing in the wild”. Such a test could be done e.g., by conducting a workshop involving a number of expert pCAD users during a conference in the general field of CAD or HCI, using a more elaborate or more complete version of any of the interactive prototypes developed in this work.

**Conducting Quantitative Evaluations**

As mentioned in Chapter 3, § 3.3, much of the work described in this dissertation can be classified as exploratory HCI design, because merely augmenting and enhancing current (single-state) computational tools cannot lead to a successful resolution of the research problem at hand. Thus, in early-stage, exploratory HCI design research, quantitative evaluation methods are in general unsuitable and even counterproductive due to the lack of mature HCI designs, as well as due to the danger of eliminating promising branches of design early on in the process.

However, once such mature design artifacts (and especially interactive prototypes and even
fully-featured, release-quality applications) start becoming available, quantitative evaluation methods and techniques become suitable and even desirable for assessing said artifacts. Such evaluations might include e.g., controlled quantitative studies, logging and analysis of low-level system events and user actions (e.g., mouse events, eye and gaze tracking, recording backtracking steps and other types of design history traversals, and so on), task performance measurements (speed, accuracy), user satisfaction levels, various types of questionnaires, as well as the application of sophisticated statistical methods to analyze such sets of quantitative data. The interactive prototypes developed in this work represent a significant step towards employing quantitative studies of future alternatives-enabled tools.

**Generalizing This Research to Other Domains**

It is my contention that the work described in this thesis can readily be applied to 3D modeling, CAD and pCAD alike. For example, the conceptual model presented in Chapter 7 is sufficiently general to be extended to 3D models; the same holds for interaction tasks and methods that I presented in Chapter 10. Potential issues might include e.g., more complex navigation tasks, occlusion issues, or higher computational demands due to larger size and complexity of parametric 3D models.

Likewise, I argue that the design guidelines and design principles can be relatively easily be transferred to the domain of text editing as well, whether plain text (i.e., source code), or rich text editing. Again, the conceptual model presented in Chapter 7 is sufficiently general to be extended to concepts such as “words”, “paragraphs”, and “sections”. Next, any text can be assumed to be globally constrained in the top-to-bottom direction, for any writing system. For instance, chapters, sections, sub-sections, paragraphs, and sentences always follow each other in a sequential manner, from top to the bottom of the page in a document. In addition, individual words within paragraphs are constrained in most languages to follow the left-to-right direction, or in some cases the right-to-left direction (such as e.g., Arabic and Hebrew scripts). Thus, both direction senses (top-to-bottom, and LTR or RTL) can be regarded as two special types of one-way data-flow constraints, within the type of parametric models that I heavily used in this work. Different alternatives of a text document, then, might have some sections, paragraphs, or sentences reshuffled to different locations within the document, thus leading to different variants or alternatives of the text document.

**Generalizing Cyclic Models of Cognition**

As another suggestion of future work, it appears that the different cognitive models (Figure 12.3) I mentioned in this work possess a number of similarities and shared attributes. I am not aware of any research that explores these similarities, and then attempts to come up with an unified model.
For instance, all of the following models: Schön’s “reflective practice” model (page 301), Boyd’s “observe-orient-decide-act” decision-making model (page 302), and Kolb’s “experiential learning cycle” model (page 304) feature the following:

1. All models derive their input from the environment.
2. All models represent a continuous loop or cycle of activities.
3. All models feature four principal cognitive activities, that can in generally be classified as sense, reflect, frame, and act.

I believe that these and other similarities merit a closer look, which might perhaps result in a more general and inclusive cyclical model of cognition. The model that I proposed in Section 10.1.1 (Figure 10.1 on page 219) represents one step towards such a generalized cyclic model of cognition.

12.5 Closing Thoughts

The sustained pace of scientific and technological progress makes it now possible to imagine an artistic studio, workspace, or office of the future, where the designer can work with multiple design variants displayed across a number of large, high-quality display surfaces. Suddenly, the digital design models that have been lying dormant and inaccessible due to the inherent limitations of single-state tools, lack of screen real estate, lack of processing power, or proper design history mechanisms, can now at long last “break free”, and become readily available in their full richness and variety for their immediate visual evaluation, comparison, reflection, discussion, collaborative modification, recombination, and aggregation. It is my hope that this thesis is a small but important step towards that vision.
Appendix A: Interviews

This appendix contains transcripts of interviews conducted with the participants of the CAMBRIA 1 “think aloud” formative evaluation study (see Chapter 8, Section 8.2 on page 153).
Participant P1

38:18 Participant  (Tests Solo and Mute on four alternatives in the parallel editor.) I am not sure why Mute... I am not sure I would ever want... I see, so this helps me if got all three and I only want to... But I still wonder, why would I ever want to turn off all of them at once? See, I think, now I want to bring up everything... Oh, that's what it did. Hmm. That is kind of confusing.

38:49 Interviewer  Yeah, this is still a little bit buggy. But the idea is, when you press Solo, you are focused on just this alternative, right.

38:56 Participant  Yeah, that part makes perfect sense. I am just wondering why... (continues trying out Solo and Mute buttons.)

39:02 Interviewer  Yeah, this is supposed to go back to the previous state. This is still a little bit buggy.

39:08 Participant  OK, that would make everything... That's what I would expect. OK. And, so now... (continues editing in silence for a minute or two) OK, we only send it (passes some items from one alternative to others) to the ones that are active? So what sort of questions would you have. I am not very good with line drawings, I am not sure how much further I can take this. So, what sort of questions do you...

41:07 Interviewer  Yeah, just your general thinking, what do you think, is this useful, or ...

41:16 Participant  (Discovers the button on top of parallel editors which perturbs selected graphical elements in random fashion.) Sorry, does this actually work? Oh, that's interesting. I am not sure, if I were doing real (work), if there were actually distinct relationships that these things had, random, like, this might be too random. Right now, I am just looking for interesting blobs on the page, and this suggests different ways of doing it. (Continues parallel editing.)

42:47 Participant  Can you Undo? No? That's (Undo) is something that I would really want, because now I lost where I was.

43:08 Interviewer  So, did you try here to draw something, like in the previous version? What are you trying to achive here?

43:17 Participant  I am just playing around, I just thought this was an interesting thing (an interesting alternative), and now I thought, OK, what would what would make it more (interesting)... This is an interesting starting point, that was different from what we had. I like that shape (a circle),
I like the way that shape (a curve) is intersecting it. Now I just want to play around with. (Experiments with the design by dragging the circle continuously around the alternative’s canvas.) OK, that gives it more symmetry, that is sort of ... that’s kind of interesting, you can do that... Tangents can be interesting, but non-tangents can be interesting too.

43:49 Participant So, how do I create a new alternative in this one (in the Arrangement view)? OK, here. And if I want to have them both available, I just use rubber-band dragging. So how do I duplicate that in here? (Wonders how to duplicate all the elements from one alternative to the newly created one.)

44:16 Interviewer (Investigator shows how to duplicate an alternative in the Juxtaposition parallel editor by passing items from one alternative to another.) In this case, we are calling it Replicate item. So, is this something that you would like (to have)?

45:06 Participant Well, having... Having... So, now unfortunately I cannot back to what I had before (because Undo is missing), but whatever that interesting thing was, I’d like to start branching off alternatives from this (points to an alternative in the juxtaposed parallel editor). Like, OK, start with that, keep that around, or whatever it used to be, and now give me another one where I, you know, compare, something like this (moves a circle around in one alternative), or I think that it how it was originally. So I can keep that, bring another one maybe like that (drags the circle to another position), bring a third one (drags the circle to yet another position), a fourth one like that (again drags the circle), and then something like that. And have them starting to branch out (participant gestures with his hands, broadly waving at or around the juxtaposed parallel editor), and then I could say: what I am doing? Now, I am just playing around with lines at this point, I am not trying to design something specific.

45:59 Interviewer Yeah, so this is one way to do it (i.e. how to duplicate an alternative). (Investigator shows how to duplicate an alternative by clicking on the Duplicate menu button in the icon list view / Thumbnails view in Alternatives widget.)

46:30 Participant So now, are they (replicated graphical items) still linked, or is it only...? No, they are not. Yeah, that’s a sort of thing.

46:39 Interviewer Alright, they are supposed to be linked. This is just an very early prototype. They are supposed, or going to be linked (in an improved prototype).

46:50 Participant Right. The way I would want to do, is just solo this (alternative), move it (the graphical item) around, then solo another one, move it (the linked item) around, create much (sic) alternatives, and then start playing around.
Interviewer: But you would like to see all of them at once?

Participant: Yeah, for visual elements, all at once. Just to be able to compare them. This could well be different for people who have a lot more training in visual things, but it’s really really hard for me to remember visual arrangements. So if I had to look at each one of these, one at a time... (selects different alternatives in icon list view in Alternatives widget). I mean, I can sort of remember, imagine I can’t see that (blocks Alternatives widget with his hand, and clicks on different alternatives one by one thus showing in the parallel editor) OK, then it’s this one, then it’s this one... It’s like: what is he doing again? Seeing them up there, something like this (points at icon list view), or something like that (points at arrangement view, and zooms into an arrangement of five alternatives).

Interviewer: So you find this (ability to compare many alternatives) useful then?

Participant: Yeah, absolutely.

Interviewer: You can also increase the size of thumbnails as well in the thumbnails view (investigator points at the icons list view in Arrangement view).

Participant: Yeah, so now I can see OK what do I like about these various things (i.e. different alternatives), but that’s different from textual-based things, where looking at the text doesn’t tell me... The difference is, the (text-based) alternatives are in my head, because they are much larger actual decisions, and you can’t represent these little bits, characters, on the page, so I don’t.... I only want to look at one version of the text.

Interviewer: What about if you had a number of different outlines on the screen like this. Different outlines, like you have one draft with Introduction, Methodology, Results. Another one with Introduction, Background, and Our Approach. (And then compare these two outlines?)

Participant: Yeah, you see, looking at them, that I can keep in my mind. I’m thinking: should we put there a typical Research Question? Should I put the Related Work at the beginning, and does it drives the Research Question? Or should I just present the Research Question at the beginning, and then add the Related Work at the end. That is a common question. And I can think about that. I don’t need to see them. I am thinking: if I do this, if I do that... Another thing is that it is so much work to actually create those alternatives for text-based things, like prose and code. And, that could, necessary for these graphical things to work, it just has to be for an experienced user, not necessarily first-time user, has to be so easy to draw, to create, to link, to delink. You really, when you are skilled at it, you are focusing on the alternatives, and you are not focusing at
all on using the software, or trying to get to do what you need to do. When you have that in the visual domain, it is much better suited to have multiple alternatives presented at once.

50:23 Interviewer Are you familiar with the Hartmann’s paper? (2008 Hartmann’s paper - “Design as exploration - creating interface alternatives through parallel authoring and runtime tuning” - as an illustration of a system [Juxtapose parallel editor] that is capable of managing multiple alternatives in the domain of text.)

51:30 Participant (Skimms the paper by Hartmann and comments on it.) Yeah, I haven’t... Again, I think this is more about visual.... So long as you don’t have to do too much code to, er...

52:08 Interviewer He (Hartmann) is actually preserving identities among paragraphs of the source code, elements of your code. So if you change something in one tab (points at the Juxtapose parallel editor), or one version of the code, is automatically changed in other versions.

52:23 Participant For interface things, like changing the color, or certain modest changes to the interface, that’s going to work well. But code is so tightly woven to what came before and what came after and everything else. So much of programming is figuring out where to couple and where to decouple and where to separate, and where to blend, and once you made that decisions, it is a hellacious amount of work to change them often. So, sure, I think that’s fine for interfaces, I don’t think for... and modest changes to interfaces, if you set it to zooming interfaces. Think about your two interfaces. That would be very hard to have one code-based that supports both of those, that you are working on simultaneously.

53:30 Interviewer So you would say that you are a little bit skeptical?

53:34 Participant I think, maybe... I haven’t had the chance to see where he is doing this, but for certain aspects of user interface design, it makes sense, because it could be better than just sketching things up on a whiteboard, because you can actually try different things and see quickly what they look like. I am more cautious about not this (points at Fig 1 in the paper, showing parallel tuning of runtimes in Juxtapose), this makes sense, but this (points at two source code alternatives in the Juxtapose parallel code editor), the way to keep parallel copies of code and link it, and make one change in one relevant to the other, all in almost unlimited ways. For example, changing this variable from zero to fifty five, or something, fine! But, substandard changes in the code, it’s gonna break everything else.

54:27 Interviewer Yeah, the question is, how do you deal with structural changes. But, I could see him for example using a parse tree, and then just maintaining correspondences between multiple parse trees.
Participant: Yeah, well... (Gives an off-hand remark about doing something similar and using the same controller in Jenning's PhD work.) But, the thing is, it is not so much the technical capacity to track changes, it's just, when you get conflicts, the computer can't resolve them, and it's waaay too much work for a human being.

Interviewer: Yeah, that's typically the case with software versioning systems.

Participant: Yeah. If they (versioning systems) could do that, then you would have automatic programming, you would not need programmers, large parts of programming would be done automatically, and that has not happened yet.

Interviewer: Ideas started to appear, in a sense. They are still scattered. We are trying to come up with a comprehensive bibliography for this field. Computing, design, ...

Participant: So, what's your schedule? I mean, at some point, you have to finish up and graduate. So, it's take a lot of work to produce these two prototypes, I'm sure.

Interviewer: Yeah, it's a lot of work, but the last one(). So, as you progress with these prototypes, you learn more and more about the problem, and the inner structure of such an application. I think I am on a good path. Once we evaluate this what we have right now, we will have another instance of code writing, to improve on this one. I will take the last one, and then I will just add features that are still missing from the first one, like curves, and things like that.

Participant: I think the modest undo is also just really important. Oh, I did not mean to move that thing. Ooops. The problem is, when people try to support alternatives with undo, undo is really awkward and inadequate. But for ooops I just moved something, I literally undo, that was a bad choice, I did not intend to do that, undo is the way to go. People are so used to it. Curves are going to be necessary, for people, if you want to get real designers and play around with it. And then Undo.

Interviewer: Once we have this, again, we will have a final round of evaluations, we are still thinking about what we are doing to do, probably "cognitive dimensions of computing artifacts". (a 14-minute discussion on the merits of CDF framework follows. I leave out this from the transcripts because it's not directly relevant to this study.)

Interviewer: I think we can complete the final part of this study, and this is the post-interview, what do you think, and what are your opinions about the prototype. What do you think are the most interesting features of these prototypes?
1:12:30 Participant  The most interesting features would definitely be the linked items, and the capacity to, being able to simultaneously affect the same item in every alternative, but also be able to focus on one alternative only. That is very interesting. Being able to pass properties, and so forth, and see all these alternatives at once, that is the single most interesting thing.

1:13:57 Participant  And the arrangement view, because from my long experience I know that fixed grids like this are great for something, and are getting in my way for other things. There will definitely be times where I think, actually, I want to consider these two, and I want to consider these three, you know. Arrangement in this context is physical proximity for ready comparison, and that’s not something I can do with I can do with this (points to the 2x2 grid in parallel editor). I suppose, you can probably move those around (tries to move alternatives in parallel editor). No, you can’t. But you can imagine moving them around, like tiles, or their boxes around. But this is limited. This (points to arrangement view) gives you the flexibility.

1:15:02 Interviewer  Do you still find having a separate editor is useful, or would you like to edit them in-place (in the arrangement view)?

1:15:09 Participant  I would bet that I would tend to spend most time in the arrangement view. This view would be expanded (tries to stretch arrangement view over most of the screen).

1:15:18 Interviewer  Would you like to edit them (in arrangement view) as well?

1:15:25 Participant  Basically, I would like to be able to do this (points at juxtaposition parallel editor) in that (points at arrangement view). Now, for this, you need to have some sort of way of doing the mute. Have you seen Chris’es thesis? (Chris Jennings’ PhD thesis at SFU.) You hover over thumbnails, and the operations would appear (halo menus). That would be a way to put a Mute button on it. When you got alternatives, you do not want buttons around everything, because this just clutters the field. But if you know, oh yeah, if I want to mute this, I just hover over it, and the mute button is right there.

1:16:05 Interviewer  Interesting. Should it (the halo menu) follow the cursor?

1:16:07 Participant  No, it (a halo menu) is around around an alternative’s thumbnail. You move over this, and they appear around the thumbnail.

1:16:15 Interviewer  It could be a little bit problematic, if you zoom in completely (into an alternative’s thumbnail), make it like this big (the size of the screen), then you can’t have it (i.e. the halo menu) around (the alternative).
1:16:23 Participant  Yeah, OK, so there is issues. You know, maybe you shouldn't be able to zoom in that far. If it's filling the screen, you would not want to mute it. Well, you might. Most of the time, when you do something like that on this, when you play with mute and unmute, you will have multiple alternatives on screen, because that's the only reason for wanting it here.

1:16:55 Interviewer  Do you think this would be actually feasible to have these little images like this (zooms out of a collection, making thumbnails small), and still be able to edit them? I mean, visually. You can also drag this (parallel editor) to another screen. (The investigator drags the parallel editing widget into another, second physical screen.) Let's make it like this (increase its size). I can have this on another screen. Do you think this would be useful, if you have another screen?

1:17:35 Participant  Yeah, absolutely. I think, there is times when I want to to edit here (in the arrangement view), and there is times when I want much greater control that (parallel editor) offers. If it came down to it, like I only had one, edit there, and see here (in the arrangement view). The way it works now. If it were just too much work, then I would say, make that the (), and ....

1:18:00 Interviewer  But, in let's say a perfect version of such a system, you would definitely like to be able to edit them here (arrangement view), right?

1:18:09 Participant  Yeah. I won't be able to zoom in, but if I wanted to do precise, careful editing, I would do it here, in the parallel editor. But, once I want to move that one, just over a bit, and see how it compares with other ones... And I would also be able to select one of these (in the arrangement view), and duplicate them, create another one. There was a button for this, right? I see, in the Thumbnails (icon list) view. Being able to do the same operation here (in the arrangement view).

1:19:10 Participant  Also, I can give you forty-person years of programming effort in five minutes, so, when duplicating, it should appear near and not overlapping, everything else should move away. The existing alternatives should move out and the new one should be between them and somewhere near the original. Because, the first thing I will want to do, is to compare how they look, and if they overlap, that makes it impossible.

1:19:59 Interviewer  What do you think are the least interesting features of these prototypes?

1:20:03 Participant  Collections widget. I expect to... Basically, I would have everything (in the application). When I have multiple projects (then yes, I would like to have a Collections widget), but within a project I would have exactly one collection.

1:20:20 Interviewer  OK, but you can have just some collections open here, and the rest closed.
1:20:27 Participant  (Dealing with many collections is) too much work. I rarely want, for things like this, I rarely want folders and folders and folders. I want to distinguish projects, so if I am doing an interface design here, or if I am doing birthday cards here, those would be completely distinct projects, stored in different places in the system. But, within a given project, I’ll put all the alternatives (into just one collection). Especially, if I got that kind of view (points at arrangement view), I would just shove some of the worst versions off to one side.

1:21:00 Interviewer  So, what you are saying, you would have your whole project in just this window, and that’s it?

1:21:03 Participant  That’s right. I would not use collections. Well, I don’t think I would use collections.

1:21:09 Interviewer  I see. Interesting. I guess, here (points at the arrangement view) you can cluster them, but once you get back to the datagrid view (in Alternatives widget) it’s all in one list.

1:21:19 Participant  Sure. The only time I can see myself using the datagrid is look at the dates. That’s useful. I would probably want these to display “five minutes ago”, “two hours ago”, “yesterday”, “last week”, “last month”. This (i.e. current view) doesn’t differentiate an alternative I worked with a week ago from an alternative I considered five minutes ago.

1:21:55 Participant  By the way, that’s another thing, useful in these views, for me to say: make the older versions more transparent. So I just look at it, and go, what are the more recent ones, I come back after a month, what was the last one I worked on.

1:22:24 Interviewer  Envision this in a much more developed and polished version. How would you work with it in your practice?

1:22:35 Participant  I don’t do a lot of graphical work. If I did, I would love to have something like this. If I were doing photo editing, photo montage kind of stuff, this kind of thing would be useful. Any sort of visual editing, this would be useful. For prose and code editing, not so much. Not very useful.

1:23:06 Interviewer  What barriers does is present to work?

1:23:23 Participant  I think, clearly lots of functionality is missing, like in Photoshop and similar. Other than than, I don’t see that it has any barriers particularly.

1:23:53 Interviewer  How would you change it?
1:23:55 Participant I think I have given you my suggestions. Get rid of collections, allow me to edit in arrangement view, give me curves, shapes, undo.

1:24:32 Interviewer Do you have anything else to add?

1:24:36 Participant A really interesting project! And a huge amount of work, I am sure. Each prototype.
Figure 13.4: Event categories for participant P1.
Participant P2

22:41 Interviewer  This was a quick walk-through the prototype, and, in the third part of this study, I would just like to ask you to play a little bit with the application.

22:55 Participant  OK. There is no specific task to do?

22:57 Interviewer  No specific task, but you can choose one, perhaps create trees, or whatever you find interesting. (Meanwhile, the participant tries to open a collection by double-clicking on its icon in the Collections widget.) This is for renaming (a collection). If you want to open it, you just go to the little book (icon) at the top.

23:16 Participant  Oh, if I want to do (), I can name it here? (Clicks onto the text field at the top of Collections widget.)

23:20 Interviewer  This is filter (field), so if you have lots of collections, you can filter them in this text field. (The participant again double-clicks on a collection icon in the Collection widget.) This is Rename, but it should actually be "opening this collection".

23:31 Participant  Yeah, that’s what I was... yeah, the standard Windows metaphor. So I can call it (the collection) whatever I want, right? (Renames a collection by typing in a new name.) But to open it... (Clicks on the 'change view' icon, featuring a double arrow, at the top right corner of the Collections widget.) No, that switches the view. This arrow (in the top-right corner) suggests that it would put it into (sic) this way (points at the Alternatives widget), like, it suggests that it’s moving (a collection) back and forth (i.e. from the Collections widget into the Alternatives widget, and vice versa). How do I open it (a collection) again then? On the book, oh, like opening a book. I see. OK, if I wanted to do... Can I move across idea clusters (tries to move some selected alternatives in the Alternatives widget into an icon in the Collections widget). No.

24:05 Interviewer  Oh, you would like to drag it (an alternative), right? OK. So this is missing, but this will be... You could actually drag it, just switch to the MDI view, this one, yeah. Just drag this one (alternative) to this one (i.e. MDI window containing a collection of alternatives) to this one (i.e. another MDI window containing a collection). Sorry, you just have to switch to the list view, for the time being. Yeah, so drag is working with these modalities.

24:38 Participant  (Participant switches to icon list view in both MDI windows, and drags and drops four alternatives from the first MDI window to the second one). Oh, okay. Okay. All right. (Drags and drops some additional alternatives.) So this is my empty one (i.e. empty collection). Oh wait, but... Oh this is seeing all of them (collections). But in the middle here (i.e. Alternatives widget) I’m seeing all three of these clusters? (i.e. all the alternatives in a collection.) Is that true?
25:01 Interviewer  Yeah. This is for... So you just have to switch to the graphic view (arrangement view in the Alternatives widget). Yeah, this (collection icon) still has to be refreshed (in the Collections widget). This is C-2, if you just select some of them (i.e. alternatives in C-2), just draw something... (The participant draws some elements in one selected alternative). It (i.e. the miniature of the alternatives in C-2) should appear here (C-2 icon in the Collections widget), but... It's not appearing. OK. It just takes some time, it's still a little bit buggy, but...

25:31 Participant  Yeah. I guess what I was referring to, is the only () here that is selected is this (points to an icon in the Collections widget), yet I am seeing all three (in the Alternatives widget), right? Is that right?

25:41 Interviewer  So, you think it's a little bit confusing, right?

25:42 Participant  Yeah, because normally I would only expect to see all three (MDI windows) if I have selected all three (thumbnails) on the side (i.e. in the Collections widget). Let me try and do some drawing. So this is how I can get this back to just the one that I've selected? (The participant consecutively selects icons in the Collections widget and expects the corresponding collection MDI window to be shown and focused in the Alternatives MDI area.) Oh wait, it isn't. But I have five in it? How come I have five in it? It says zero... Or it's just not updating. (The participant is puzzled by the fact that the icon shows "0 contained alternatives" while there are five alternatives in the Alternatives widget. This is due to a refresh bug in the prototype.)

26:08 Interviewer  Sorry... Collection 0. No, you have to open it actually.

26:14 Participant  It's still not open. So there is four (alternatives) in it? In this one it says zero, but there is actually four? OK, this is a little bit buggy. OK, that's fine. OK. So, I can call it (an alternative) whatever I want, right? (Double-clicks on an alternative in Alternatives widget and renames it.) OK. So let me draw a house. (The participant edits an alternative.) OK, I can erase lines easily. (Continues editing.) OK, so I am drawing my house... Let's say I want to make another one (i.e. alternative). How can I do one, is with the plus over here? How can I start with the one that I have?

27:54 Interviewer  You select both of them. (i.e. the two thumbnails for both alternatives in the Alternatives widget: one originating alternative, and an empty one which will receive all the elements from the originating alternative.)

27:57 Participant  (Tries to duplicate an alternative to an empty alternative, and for that selects both alternatives in the Alternatives widget.) Oh, I have to pass the items. (Tries to select items in the originating alternative, but fails and draws a line instead, because the program is currently in
CHAPTER 12. CONCLUSIONS

drawing mode.) Ooops. Undo is not working. I can’t select it unless I click on the Select tool. OK. So I got my second one (alternative), I can select them all (i.e. elements), I can pass the items to it. I see. OK. I kinda expected to be able to just say, from this (right-clicks within an alternative in the parallel editor), ”create new”. Pass items to new. I expect it more in this context (i.e. within parallel editing area) maybe because, if I’m working in here (in the parallel editing area), this is where I am working, and now I am working over here, and if I wanted to pass all these to a new (alternative), and I am going way over here (i.e. to the Alternatives widget) to do it, seems like I am going out of my way. Or, yeah. I would expect them both, actually. Okay. (Selects four alternatives in the Alternatives widgets and starts editing all four of them in the juxtaposition editor. Passes some items from one alternative to the remaining three.)

29:04 Interviewer So, actually, delete works in parallel as well. So if you select all these again... (The participant selects and deletes several graphical elements in parallel.) Delete is actually parallelized as well.

29:17 Participant I see, OK. (Continues working with four alternatives in the juxtaposition parallel editor.) And if want to see this big? (Tries to maximize an alternative editing pane by double-clicking on its title bar.)

29:26 Interviewer You just, with the mouse wheel, you can scroll or zoom into it.

29:31 Participant Oh, I guess, I meant fill the whole screen. (The investigator suggests several ways to increase the alternative editing window.) Oh, sorry, I mean take this whole window, and make it big. Just select the window, with a rubber band, zoom into it. I don’t want to see four copies of it, I just want to see A-4. (The investigator suggests to the participant to select the thumbnail belonging to A-4 in the Alternatives widget.) Er... Okay. I was expecting like, er, standard Windows interface. Like, double-click on the title bar (points to the title bar of the editing window for A-4), maximizes the window. Instead of, because I have to go... I have to go back to this UI, right? (Points to the Alternatives widget.) I just select the single one. When I’m working over here (points at the parallel editing area), I want to stay working in here, not jump at this window (Alternatives widget), which is actually taking up a lot of space for the small amount of stuff that it is showing me. Can I also make this smaller? (Tries to make Collections widget smaller.) I could. I would like to see a rectangle around this (points to a thumbnail in the Collections widget), so I know what it all contains, because the label isn’t necessarily visually associated. For this () C-0 looks like it could be that too.

31:05 Participant (continues working in parallel with two active alternatives) All right. OK. (Mutes an alternative, and edits another one.) Let’s say I forgot that I didn’t want it in the second one...
(Deletes a part of the drawing from the active alternative.) OK. (Now tries to open another collection in the Collections widget.) Yeah, I have to go to the book. ... Okay. I clicked this open it. It’s opened here (in the Alternatives widget), right?

**32:08 Interviewer**  This means it’s actually already open, because we have an MDI interface, we have to go Window (menu) and select this one (window) that was opened, right?

**32:22 Participant**  OK, I’m totally confused here.

**32:24 Interviewer**  Yeah, but I think that they actually (get) focus if you double-click (on the thumbnails in the Collections widget). Where are we now? And if you click on this one? All right, it’s not switching the focus. All right, so this is missing then. (...) OK, so this is a little bit confusing then.

**32:47 Participant**  I think. Yeah, just the way you move between windows. OK.

**32:54 Interviewer**  If it’s easier for you, you just leave this mode, and do this, we can actually switch to the tab view. This is perhaps more intuitive.

**33:07 Participant**  I see. Yeah. It was confusing, yeah, the other mode. Especially if I didn’t know what was in it. I don’t remember...

**33:19 Interviewer**  Do you think this (Collections widget) is actually useful, or necessary, having this part here? Or, would you say this (Alternatives widget) would be enough?

**33:27 Participant**  Oh, I can see that, tabs. Actually I didn’t notice tabs until just now. I don’t know, they do seem like they are replicating functionality. But... I guess why not just work in this (Collections widget), why, yeah. I don’t know if it’s really necessary in case you kill off this (points to a thumbnail in Collections widget), you also kill off this (points to a tab in Alternatives widget containing several alternatives), because I could seemingly work in just this space (Collections widget). If I could make this bigger (tries to increase the size of the icon in Collections widget by dragging its corner).

**34:04 Interviewer**  You could actually switch to the graphical view here (in the Alternatives widget), so... just close this, and switch to graphical view. Perhaps edit in-place here?

**34:21 Participant**  Yeah! And you know what, going even further, why is this (points to the parallel editing area, showing one alternative) not in here? Right? You could have a single view, maybe I can select and zoom into whichever ones I want to. Maybe make it bigger. (Tries to make a thumbnail in Arrangement view bigger by dragging its corner.)
34:42 **Interviewer** So the idea was to have thumbnails here (in the Arrangement view), or, let's say, small multiples, right, to have an overview of your project, but, you think that, having an editor here in-place, would be a useful thing to have?

34:56 **Participant** Yeah, I probably would. What I am finding is that there are navigation issues between the panes. It's hard to get a mental picture, a model of where I am, because each is disconnected. Like, if I was able to understand where I am here (points to an icon in Arrangement view) in relation to this (points to a thumbnail in Collections widget), it would be different, but there is no connection right now. Like, for example, if I saw in here an rectangle (points to an thumbnail in Collections widget) that showed me I am currently doing this part here (pointing to the Arrangement view), something that ties it together. There are some InfoViz projects that I've seen that do that quite elegantly. Like Detail + Context interfaces. The tie between these views is confusing, that's why I am having these navigation issues.

35:58 **Interviewer** Do you find this parallel edit part is useful, or...? Because it kinda arranges all the alternatives into a grid?

36:12 **Participant** I think it (parallel editing) is nice, but it is hard to know without doing a real task, like it is hard to know, because most of the time we spent toy-playing around, it is hard to know, if I really had to create let's say a drawing that depicts somebody's design, how it is actually going to work. And I would expect that I would be doing this over a time period, where I would have to come back and forth from the ideas, and would it make sense to me to understand which idea is which.

36:53 **Participant** This is quite nice though that we can pull up and see (in the Arrangement view), but I almost want to in addition to slide them and see them overlaid, see a version that visually depicts the change between them. It's funny right now, like if was to put up these two, it's almost like that kid's game, right, like, find five things that are different between those images and you don't know.

37:19 **Interviewer** Actually, I can show you, we have a superposed editor. We still haven’t integrated this into a newer version, but in the older version, we have this. (Shows the older prototype.) So we have this juxtaposition, just like in the newest version, and the superposition. So we create a collection, a number of alternatives, and if I edit these, again I can add here a line, and this is now superposed (points at the superposition parallel editor, and demonstrates further for the next several minutes how it works).

38:34 **Participant** The funny thing, though, like even though it's doing it, it's mind-boggling, because you are seeing like ten... eleven different views open, I want just to see one view that has
all the ideas in it. Because it's mentally taxing for me to figure out which view is which, and just seeing ten things and trying to understand ten things is confusing, whereas I think if there was one superposed view, it would be easier. So is this a collection of all the views (i.e. alternatives) together, in here? Oh. Yet it (the superposition editor) takes up so little space on the screen, and the space is being hogged up by this stuff (pointing at the juxtaposition editor), it's more confusing.

**39:17 Interviewer** You can close this one (closes the juxtaposition editor, thus freeing up more space for the superposition editor).

**39:20 Participant** Okay. Yeah, that's definitely better.

**39:22 Interviewer** You can select three of them, or let's say, just these two. So in this case you have just two alternatives superposed on top of each other.

**39:32 Participant** Yeah, see, this (superposition) makes more sense to me (than juxtaposition).

**39:35 Interviewer** But, the idea is to have also this capability to have both of them, or N alternatives active, at the same time, select all of it, and move, you know, in bulk across the screen.

**39:53 Participant** This makes sense, I can see some situations where that might be useful. I do think though, it's hard to know with toy tasks and play tasks it's on thing, but I'd curious what it would be like in a real situation, which I actually find valuable, in a real world.

**40:10 Interviewer** What is your hunch on this? Do you think people might find this useful, or...?

**40:17 Participant** I don't know. Like, I'm sure a person would probably spend at least 80% of time in one view only, and which one of these (juxtaposition or superposition) it is, I don't know. I can imagine in my head, seeing separate is nice in some situations, seeing it like this is nice in some situations, but I don’t know in which I would be in. And why I would be in it, and what situations I would be in there. Yeah, it's really curious, to know. (Works in the superposition editor.)

**40:49 Participant** (The participant works with the superposition parallel editor.) So this is the other view (i.e. alternative), but which view (alternative)? This gray one in the background, is from which view (alternative), and how do I know it? Somehow, I want almost mouse-over, and see A-0, and know something about it. Or, I can even imagine, these are actually, you could tell they are stacked cards, in flipped order, like the layers in Photoshop. If I could actually physically reorder these layers somehow, that might be a nice way. If I could tilt it, and then I could see actual layers. Like index cards, or something, pieces of paper stacked on one another, transparencies, shuffled around. Because, I think the curious thing would be, what if decided to combine two ideas, and
wanted to overlay them, and overlay certain parts, like transparencies, I may want to turn some sideways, to shuffle the order, and then put them back in place.

42:00 Interviewer Would you like to sketch it?

42:04 Participant Sure. (Sketches the idea, of how to stack alternatives in layers.) So I can imagine like right now... This is from the topmost view, and kind of laddering, if I was to rotate this in 3D space, I could then see that this is actually comprised of this alternative and this underneath, and this top one has my... I could shuffle them around like this... Move them up and down to get different orderings... I could pull them out, if I wanted to. Ooh, I could have two piles. Then I could have multiple superimposed piles on top of each other. Gosh, I could even do more piles. I could turn off some, don’t show me these ones. Show me those two. So I could see what the combination of different elements would be. I don’t know, I might even throw in a special, black piece of paper, or maybe it’s like a partial paper, mask out this half of that piece of paper, don’t show me this side of it. Different ways of picking different superimposed objects.

43:46 Interviewer Like blocking. (i.e. selective blocking of just some parts of an alternative)

43:50 Participant Yeah. One of the things I like in the new version, what I like about this (Arrangement view), this really flexible kind of style. I can put these wherever I want, I can make piles, I can group them spatially, and they are almost just like real-world pieces of paper. As though this is my desk, and I can just move stuff around, I can cluster however I want to. This is what I really like. But then, this view (points to parallel editing area) kinda takes away from the elegance of this view (points to Arrangement view), and I almost want to go to do the same thing here (in the Arrangement view) as I do here (in the parallel editor), which suggests to me that the real work’s base is here (Arrangement view), here I can do everything that I can do on the right side.

44:32 Interviewer So you would then definitely prefer to have an editor in this part (Arrangement view) of the screen?

44:36 Participant Yeah, I think so. Yeah, with even more abilities, than I have now and am able to do, yeah. And even if I could then drag, when zoomed out far enough, I’d see this rectangle (points at a thumbnail in Collections), and I could even move that around, because I maybe want to spatially cluster these collections of ideas.

45:02 Interviewer Like a nested model, perhaps?

45:04 Participant Yeah, nested somehow.

45:10 Interviewer So what would you expect on double-click?
12.5. CLOSING THOUGHTS

45:14 Participant I guess it did, I expected it to open.

45:19 Interviewer Editing in-place on double-click, that’s what you think what be the most natural choice. Because if we take a look at Google Maps, for example, double-click is zoom in.

45:30 Participant Yeah. I don’t think that’s a good interaction on Google.

45:38 Interviewer You could, let’s say, switch to edit mode for example here, and still use the mouse wheel for zooming in and out.

45:46 Participant I think so, yeah. So there is move around mode, and there is edit mode. Yeah, I can see having the menu for edit mode, and also for move mode, and I can switch between the two.

46:08 Interviewer So in that case you would actually have a viewing mode, where you can’t move anything around, edit mode where you can move those icons around, and a third in-edit mode for actually changing the elements inside the alternative.

46:26 Participant I think so, but it's something I think a person had to play with to see... You might have hunches but I don’t know for sure. Okay. Cool. (Tries dragging an element from an alternative in parallel editor to an alternative in Arrangement view.) I’m just curious, yeah, I am not sure how useful it would be, but...

47:11 Interviewer Do you think that this (parallel editing area) will still be of value, if we implement editing here (in the Arrangement view)?

47:18 Participant No, I don’t think so. If they are together in the same view, I don’t think you need to do that. Yeah. I really like the real-world metaphor, as if they are pieces of paper.

47:38 Interviewer Also, yeah, less work with switching context from one pane to another.

47:45 Participant Yeah, exactly. Although, you know, it’s funny, this kind of real-world metaphor is used much less in software design right now.

47:58 Interviewer Yeah, there is a certain move away from skeumorphic interfaces.

48:04 Participant (Works with alternatives, including changing the color of lines.) Oh, I have to select the line first. Oh, that’s kind of odd. I expected it to be like Paint, where you... just define the pen, and use it afterwards. (...) And this switches to mute. Okay. I am not sure I have many more thoughts.
That was certainly pretty valuable, I heard a number of interesting ideas from you today. OK, so that was the third part, let's move to the final part. Just a short post-interview, your overall impressions on the prototype. So my first question to you would be, what do you think are the most interesting features of this prototype?

It's interesting how you can move ideas, I can move elements of different ideas around very easily. Which I think could be valuable, but in a drawing context, I am not sure how often I would do that. Because, I guess, the only time when I am really drawing stuff, is when I am sketching new user interface ideas. And there, I try and come up with as many different ideas that I can, where ideas are very different. So I am not making replications of an existing idea, and I wonder if by being able to so easily transfer elements it would cause people to be more likely to have small iterations of their ideas rather than big iterations. You know what I mean? It would be valuable certainly, but I am just wondering what the long-term practice might become. So, what I think about the original examples, you were showing me for example the guy who made the actual prototype of different buildings. To make each of those, it took him probably a long time, and it wasn't just a few seconds, and he has a duplicate that he transformed. The fact that it is so difficult to create that second version, means, well I would think that he would be more likely to create something completely different, rather than a small modification of his original idea. So if you wanted to get many divergent ideas, it's almost beneficial to make it hard to duplicate. I am not sure this is actually encountered in practice, but it's something to think through. Maybe it's neat to transfer your ideas, make people make divergent ideas, without making people explore alternatives.

So you are saying actually that this mechanism of passing items is limited in the sense of not helping much with structurally different ideas?

Yeah, like it's encouraging me to have N+1 ideas, where they are only slightly modified from my original idea. It's curious, because it's trying to make idea generation phase faster, and more efficient, but are we taking away something that is actually quite important, that difficulty to make new ideas? I don't know, it is like this philosophical conundrum, this copy-paste sort of benefit, like a detriment I guess to our ideas. I think comparing ideas is probably most compelling thing, so you can go back and can look and can see, and if you could more easily do it with superposition somehow (picks up mouse and points at arrangement view), as opposed to this thing (i.e. differences game for kids), and if you could more easily do it with superposition somehow, how far the things have changed.
Perhaps make it transparent? (Referring to make design alternatives / drawing transparent in the Arrangement view.)

Like if I drag this over (drags an alternative’s thumbnail over another alternative’s thumbnail in Arrangement view), it starts to turn red all the things that match, as they go across or something. Some visual indicator of what has changed. Some of them are stacked, then I might know to see that. Yeah, I think comparison is the key thing. But in my brain, this is in the context of designing the user interface. I don’t know how it would be like in other design areas, like architecture... Maybe they do lots of iterations of an idea that are small multiples of it.

So would you say, you know, the potential use for drawing and sketching interfaces is limited?

Well, maybe it’s just the way I work, I sit down, I want to come up with a new design idea, for an interface, or something, to do X. I want my sketches to look as different as they can, as opposed to as similar as they can.

Interesting. So, what you are actually saying, this pass items (i.e. pass variable, pass value and clone alternative) feature is useful, but to a certain degree only, let’s say for small, incremental changes.

Yeah, I think it would encourage incremental changes to an idea. In initial phases of my design, it might be not useful. But maybe that once I refined my ideas, and am trying to tweak, OK, this is of fifty ideas the one I will focus on, now the ability to pass properties across might be valuable. Idea tweaking, that is extremely valuable.

What do you think are the least interesting features?

I guess, it tries to adhere to standard Windows principles, of almost a file explorer. You open, and then you open and see more details, and then you open and see more details. When I think it would be interesting to see what it would be like, if you break out of the bounds, doing things like incorporate all views into one. I would be curious to see how that would play out.

I am really a big fan of allowing users to spatially organize things themselves, to create their own understanding. That’s what really nice about this middle view (Arrangement view). And, like this right view (parallel editor), I can see these two (alternatives) together, but if I want to move this bottom one right next..., so I can look at these, so that I can zoom into this line here. Like for an example, these two here. I can zoom in, and if I wanted to compare the side line.
What if I could move this line right next up to it, so that I can see it sidewise, like, overlap them, like this. I can’t. That’s a thing I would like to do.

57:20 Interviewer Would it perhaps be useful, again, select these two here, and be able to do this superposition in this window, like dragging this one up, while still maintaining the layout here?

57:33 Participant Yeah, maybe. Some sort of detail + context view. Yeah, I think that might be quite useful.

57:45 Interviewer Or, you know, if you remember, the superposition editor is actually missing from this newest version. So having something like this...

57:56 Participant Yeah, I think that would be quite nice.

58:05 Interviewer If you envision this prototype in a much more developed and polished version, how would you work with it?

58:12 Participant Hmm, it’s a good question. All the stuff I draw out is in a sketchbook by hand on paper. I guess you could do wireframing with this for UI design. In that context, it would probably be quite valuable. But I don’t actually do a lot of wireframing myself now, because I have grad students for that. But my role in UI design is very early on, so sketching by hand is very important. As opposed for refinement, and I think this works well for refinement. It would be really nice though if I could take a sketch and scan it in somehow, then I’d able to manipulate it, because then I would be working in refinement.

59:05 Interviewer So, I personally use Balsamiq for wireframe GUI sketching. Sometimes it so happens, I have a screen with a number of elements, and I have to copy it into another screen, meaning I take the screen, delete half of the elements, and just change some of the elements, while all the copied elements still preserve their IDs, just like here. This is just a personal observation, perhaps it could be useful for this purpose.

59:40 Interviewer What do you think, what barriers does it present?

59:45 Participant I think, certainly this ability to easily compare the parts I want to compare (represents a barrier), because right now it is at a distance. And the users have to do a lot of visual mapping themselves to figure out what is different, if they wanted to. I would say that is the biggest barrier.

1:00:17 Interviewer Would brushing perhaps help? (i.e. help with visual mapping between parts of alternatives?)
1:00:26 Participant  Yeah, that would certainly help. It's something like this, right. Like hover over, and this shows me the related lines. That certainly helps. And to see the differences, like those kids games, "the five things are not the same in these two pictures". Yeah. Or "spot the differences".

1:01:00 Interviewer  How would you change it?

1:01:03 Participant  I am not sure there is much more beyond what I've said, like I totally think collapsing everything into a single view would be quite intriguing to see, and allowing the user huge amount of flexibility over how they are able to compare. And if I could put it (an alternative) over the top (of another alternative), and if can choose the order which one is on top. And removing standard windows barriers, like sections (i.e. splitting the window into many different sections). I'd say, do it all in one spot.

1:01:52 Interviewer  Do you have anything else to add?

1:02:01 Participant  It's a very cool idea. I'd love to see it in different forms, like for (academic) papers, or for web page designs, like almost anything where you have these different ideas that you try to compare, or come up with, beyond drawings like this (i.e simple drawings shown using this prototype). I'd love to see it, in other contexts.

1:02:26 Interviewer  How would you do it for text editing?

1:02:30 Participant  Good question. So I think, I can imagine this is a paper, and I can see the paper, and this is another version of the paper, I could pull them up side by side, or same way I can just drag them on top of each other, and if I align them nicely, it would show me what is different between the two of them, turn it red or something. I can zoom into multiple ones. And then I could spatially group the papers, like these are two versions of the paper that were very early on, these are both camera-ready versions of the paper. Right now, I use the naming of the file to help me cluster and group them. But if I could visually do it, that would be far better.

1:03:17 Interviewer  And differences. All right. Okay. So, that was it!

1:03:27 Participant  Okay. Cool! Good stuff!
CHAPTER 12. CONCLUSIONS

Figure 13.5: Event categories for participant P2.

[Diagram showing event categories for participant P2]
Participant P3

21:40 Participant  That’s a map view, we can drag, zoom, OK, I see that. Oh now I select I see that in the selection over at the other side. Here I see already alternatives, so I am in a collection, oh these are the collections, OK. I have to get used to how it is called.

22:10 Interviewer  Yeah. So this is, we call it Arrangement view. Because you can arrange your alternatives around, perhaps in clusters that you would like, or consider relevant for your project. Now, this is a little bit different from collections. Now you are currently in collection C-1, and right now, in this project, we have three collections.

22:49 Participant  And I see that the alternatives are consecutively numbered, but distributed among the collections, so... (Accidentally closes a tab containing the Arrangement view of all the alternatives in a collection.) Oh, I closed one collection accidently, by clicking on a (“Close Tab” button located on the tab)... So let’s see... Let’s get the collection back. (Tries to reopen that tab / collection somehow.)

23:08 Interviewer  If you would like to re-open it, you actually right-click on the toolbar and open the (widget) “Collections of Alternatives”.

23:22 Participant  (opens Collections widget) OK, I got that. So I get this window, one to three. (The participant now has three tabs open, with each tab containing an Arrangement view for its associated collection.)

23:30 Interviewer  So C-1 (is now visible in the Collections widget), and there is a little booklet (i.e. a button with a booklet icon) on top (to open its tab in the Arrangement view). You can actually rename it if you wish as well, your collection. C-1. You just double-click (on the thumbnail designating C-1 in the Collections widget). (Participant renames a collection, and attempts to open it in Arrangement view.) Yeah, we are still working on the double-click (to open the collection), that would be the most logical operation here (to open a collection’s tab in the Arrangement view, but which is not implemented in the prototype), but you can click on the booklet (i.e. a "Open Collection" button with the booklet icon) on top to open this collection.

24:04 Participant  Yeah, because double-click really doesn’t really open that (collection). So I have to... OK, go there. OK, I see. OK. So then... you want me to do some alternatives, I guess? So I just start with alternative A-9 (selects A-9 in the Arrangement view), I will just start drawing something, and let’s see. (Starts drawing a house in the selected alternative, A-9.) Because we are really happy with these mouse drawings. No grid, I guess.
24:39 Interviewer  (Investigator suggests how to make the drawing bigger on screen.) You can zoom into this (alternative) with mouse-wheel operation. (The participant zooms into the alternative with a mouse-wheel.)

24:48 Participant  Oh. It’s opposite to the Google Earth, or... (Participant finds that the sense of mouse-wheel rotation direction is opposite to the sense used in standard applications.)

24:52 Interviewer  Yeah, it’s opposite, that’s correct.

24:55 Participant  (Tries to pan the drawing, but doesn’t know how to do it and draws a line by mistake instead.) Oh, that was of course... If I want to use just pan... Oh, so, it’s right-click. OK.

25:07 Participant  (Now tries to delete the erroneously drawn line, followed by an attempt.) So I can select this (line). Oh, I have to select the Selection tool. (Participant clicks on Select mode button, selects the line by rubber-dragging the mouse pointer, and pressing the Delete keyboard button.) OK.

25:16 Participant  (Now tries to manipulate another line in the drawing, by dragging its end point.) So, can I manipulate just an end (of the line)?

25:22 Interviewer  Yeah, you just have to go to that second selection mode (Control Points Select Mode). You just click one more time, this is a little bit buggy still. If you just click on this button again.

25:33 Participant  Ah, there we go. OK. So, better than, of course, if you get into the areas of CAD systems... Er... So I don’t want to criticize too much on that base, drawing and those types of conventions, usually... I’m a Bentley user, so I want to right-click and get out of the thing. But probably Escape (tries Esc key to get out of the Control Point Selection mode), no Escape doesn’t get me also out of the mode... So there should be some way to escape these modes. But that’s more in the interaction with the drawing, then... Let’s figure out something with these alternatives here. So, pen, I can do... (Tries to thicken a line by clicking on “Increase Line Thickness” button.)

26:27 Interviewer  You actually have to go to Select mode again. (Participant is confused by three different modes.) And select the element, this is also still a little bit primitive.

26:42 Participant  There we go. Something like that. (Draws a building in elevation plan.) OK, fine. But, just... This one (line) I want to delete. I thought I already deleted that (line) already.

27:03 Interviewer  Hm, could be. Yeah. It’s, perhaps, still a little bit buggy. If you could try to delete again. (Tries and successfully deletes the line.)
27:18 Participant  OK. So this is now of course one alternative (points to it in the Arrangement view), so, and the other one, I don’t see anything, and this one I see something. So, I wonder of course how I can start populating multiple alternatives. So if I want to have this (points to one selected alternative, in the Juxtaposed Parallel Editor, with the building elevation plan drawn so far), and develop an alternative of it?

27:44 Interviewer  Yeah. So, one suggestion would be to use Ctrl-Click and select two or three or four alternatives (in the Arrangement view), and then, one way to do it, you just select all the elements in the first alternative, and right-click on these selected elements, and then (choose) "Pass Items". (Participant does this, and successfully passes all the items to three additional, empty alternatives.)

28:20 Participant  OK. So, that makes sense. So, then I can mute an alternative. (Mutes an alternative in the juxtaposed parallel editor, and leaves three of them active.) And go in here. Right now you don’t have copy/paste (capability) yet, right?

28:35 Interviewer  Copy like... ? Copy and paste? No. We are still working on that.

28:43 Participant  OK. So I just have to draw multiples here. (Continues working in the juxtaposed 2x2 parallel editor. Tries to zoom in but zooms out instead.) Oh, scrolling is the wrong way, of course. Yeah, I’m very used to the scroll in Google Earth, and Google Maps, and so on. (Now tries to pan in the parallel editor but again erroneously draws a line instead.) And now I want to pan. Sorry. Oh, it’s the right click (participant now successfully pans all four drawings, and continues editing three alternatives in parallel) So, something like that.

29:33 Participant  Then, of course, you know, later on, one would expect some grids, and snaps, and those types of things, but this is of course...

29:44 Participant  (Tries to delete a line.) OK, so, I have first to select, then right-click (to pick "Delete" option). OK. Now I want to switch off another alternative here, let’s mute this one. I’m not necessarily very systematic about it because I am just familiarizing myself with how it works. (Continues editing two active drawings, with other two being muted.) OK. Something like this here. Of course, the lines are a little bit off, but I guess that is to be expected.

30:44 Participant  Can I multi-select (these lines) with Ctrl-clicks? (Selects lines constituting a grid in what appears to be a door or curtain wall in the drawing.) OK, there you go, and I will make those (lines) not quite that thin. (Makes those lines thinner, in parallel, for two active alternatives.) And, the same with those. (Selects another set of lines for the two active alternatives, and makes them thinner as well.) There you go.
31:02 Participant  Now, OK, now I want to select those (selects the first set of lines modified), and say, what do I have here, I want to pass properties. (Selects "Pass All Properties" contextual menu for the selected items, but nothing happens because the other two alternatives are still muted.) And, of course, I want to pass it over to that one. (De-mutes the alternative, and passes "Pen Width" property.) OK, that works. Cool.

31:35 Participant  Cool. So now I have those alternatives. In this view (juxtaposed parallel editor), can I rearrange the windows, if I am not happy with how they came up?

31:45 Interviewer  For the time being, you can’t. But if you describe to me how would you like to arrange (them), that would be useful.

31:52 Participant  Well, right now, I would just try to drag them by the header. Of course, it’s an ambiguous operation, all the other windows should react to that.

32:05 Interviewer  I see. You would just like to change their relative position in this grid, right?

32:11 Participant  Right. I just want to get this one up there, this one over here. That one down there. Maybe, I don’t know. Maybe this one over to the right. I’ll probably look at it, just to the right up. The two other ones, because they right now identical, down here, and then I would go on, and mock around with those. But this one, yeah, this one I want to have over to the left here. Right now, can I just deselect them, and then if I select this one... OK, so now it has the same arrangement as it has in here. No, it doesn’t have the same arrangement.

33:13 Interviewer  So you would like to have a certain mapping from this view, to the parallel editing area?

33:19 Participant  No, I’m just trying to figure out how I can control on this side (juxtaposed parallel editor), what the grid looks like. And, I am not entirely sure, when this one ends up here, other then... No, it’s not even...

33:33 Interviewer  Yeah, this is still rather random, I would say. So I am still not sorting or anything here.

33:42 Participant  OK, so yeah... I would expect, if I have something like this, yeah, I am not sure why this one comes in up to the right there. So let’s try this again. See, now, if I select the fourth one, I would expect that one to fill the empty slot, right? But instead, I am not sure, I guess it goes up here. So is that the rule? It gets always added to the top there? Not really.

34:22 Interviewer  Yeah, this is still quite random.
**34:28 Participant**  There should be something so that I can arrange the view in here. In my opinion, it shouldn't necessarily be the duplication of what is the arrangement over here (in the Arrangement view), because I might want some different sorting and ordering here, because I might have more alternatives in this collection, but I am only looking at a subset of those alternatives in this view, so I want to be able to arrange them in this view (juxtaposed parallel editor), in a way that makes sense beyond the arrangement in this collection. I might have a grouping here and a grouping further down, I might select two of those and two of those, and I might want to be able to arrange them in a specific orientation to each other in this view. And then do manipulations here, which map differently into the collections back.

**35:25 Participant**  So, OK, so that's cool, so here I could go and... Right now, A-1 is active, and I'd say, if I do. And this is just active, I could work on this one. I am getting this button wrong, for the pen. (Moves a line in one alternative.) And I could say, here, OK, let's see... Can I just do this. Here you go. Something like this. Ooops.

**36:09 Participant**  I think for selection it would be good to look into what Adobe Illustrator and those tools do. You know, where everything is just one selection cursor or two, and it should really be dependent on where you go with your cursor. So if I select here, that means I select this line, if I click at the end it should mean that I select a point or something. It would be good to look at how some modeling tools do that. Way back, when I thought from Z-Ed had a quite interesting solution to that. But they had also topological modifiers. You want to get away a lot from these modifiers, or have some under right click. Because you don't want people to interact with stuff, to be down here, and then have to go up here to change the mode just to do the same stuff. I am still just trying to edit this thing, right. So I really don't want to go back and forth between the toolbox and what I am doing down there. That could be either, if you don't get around on modes, it could be either on the right click, or... I am just saying, I am selecting, it means the whole thing. And if I select this (points to a line), well, I should just get the handles, if I can the end point handles, then I can grab the handle and do the end-point manipulation. And if stay on the line, then I just get the whole element. So I think, you could base it on inference from where the user selects stuff. So if I select this, clearly it's a line, but then would be good to get handles for the end points, instead of having to switch the mode to get them. I would just get them for the line that I select. And then I could grab that end point, and move that end point. So I would do everything down here, I wouldn't have to go back up, down up, anything. But that is more about CAD system side, instead of the alternatives-side.

**38:47 Participant**  (Edits a line.) So I want to, because I figure out, this (line) should really be thick. I really want to... But for that, if I do pass properties, pen widths... (Tries to pass property.) It doesn't pass to anything because also I'm not on... (Participant states here that properties cannot
be passed over to muted alternatives.) So if I do... What do I have to do here? Ah, I have to click on this (button) up here. (Clicks twice on Mute button, activates all the alternatives, and thickens the line in parallel for all four alternatives, by clicking on “Thicken Line” button on top.) Ah, and then it’s already thick, I don’t even need to pass it. There you go. And so now I can do this, and say make it a little bit thicker, like so. Oh, and then, it’s already thick. I don’t even need to pass it. Ahh, this is interesting. So this line is that line.... (Participant discovers and thinks aloud about linked lines.)

39:32 Interviewer  Yeah, this is one property that I think is crucial here, that we would like you to comment on. This property of preserving object identities across states.

39:51 Participant  At the first instance, it’s surprising but then at the second instance it’s very interesting because coincidentally I reused the line in that other alternative with the same meaning, right? I could have messed up so to speak and put this line up here when I drew this alternative, right? So, accidentally, I have to admit, I guess, we simulate other things here. It wouldn’t have been logical, I guess. Because this is that side-wall, make this shorter and move the other over. So I think, in this action, it probably would have been the logical outcome, that it happened this way. But otherwise, if these two (lines) are reversed, you could see how that could become confusing.

41:01 Interviewer  Yeah... Well that’s because we are using lines here as atomic elements, but were this a 3D object modeler then a wall would be a wall.

41:23 Participant  It’s just something, I guess, that people would get used to, and understand the meaning. I mean, the highlight (on mouse-over) is pretty obvious. So it’s not like it’s something that suddenly pops up and you reactivated the view. So even working in the singleton view, I should have been alerted to that fact that I really need to continue using them in the same kind of usage here. (The expression “same kind of usage” denotes that the selected element must receive the same operation in all alternatives.)

41:53 Participant  Something else popped up to mind. When I was talking. What was that, I am not entirely sure. Something else related to this. Anyway, right now, probably in a way I should have known from the demonstration but it’s still something that pops suddenly up. It’s that these things, those are the alternatives of the same item, right? So it’s not like four independent copies, but it’s four versions there, of specific items that are related to each other. Some might be only in some alternatives, and some items are in all alternatives, depending on how all this was put together. So, yeah... That makes sense. Can probably get really confusing in the long run. For models of big complexity, I am wondering, people are working on an alternative and maybe get so immersed in working on that alternative that they forget that it should be applied
to others. Of course you have that capability of seeing, once it's selected, pass items and kind of, foresee update across alternatives. In a sketch like this this is pretty obvious, right, because we have a limited amount of artifacts here and none is hiding any other, right. But if you have a really complicated 3D model, even if it’s a wireframe, wireframe overlaps and may really hide what you’ve done. Elements that you might have to update but you don’t see them as needing that update because it’s kind of hidden in the complexity of the view. It’s a general problem, I guess, in all complex things, it’s not specific to what you are trying to do with the alternatives here. But it could be potentially magnified. If the people take and try to control it all, and suddenly they realize they don’t want... But that’s I guess is a challenge, whichever way they maintain variance, right. So probably even be able to see them side by side, and see the relationships, might mitigate that problem. It’s definitely easier to keep track here than saying, now I have model variant 5-B that I open up and then I don’t see the model variant 5-A anymore, and I have to manipulate my model variant 5-B without any awareness of model variant 5-A. And that is definitely worse. So here, at least people will have a fighting chance to take a look and say, oh, these are variants, and I can see them side by side. This is definitely a strength. That the things are related and get shown as related makes it also I guess more possible.

45:43 Participant I think then later the bigger questions come up that any system will have to deal with, which is how do you manage model complexity and all these types of things. Which is not specific to this system.

45:57 Interviewer I can see one way that it could help with complex models. So just imagine this scenario where you are working with this model, and you have a whole set of rooms making up a configuration, and then you work on let’s say three or four alternatives, you close too, forget about them, and then after some time you reopen them. So whatever you apply to this complex of rooms, or set of rooms, would still apply to the complex of rooms that you just opened, in those old alternatives. This, you know, perhaps there is some value to that, these ids are preserved, so whatever you do, again, in your model, will also be reflected to the elements that have the same id in other, let’s call them old alternatives.

46:53 Participant But the changes wouldn’t be enforced unless I enforce them actively, right?

47:00 Interviewer Well, we here follow the philosophy of propagating deltas. So if you reopen an old model, and you do something with the new model, then whatever you do with the new model, the deltas are being propagated, so it still affects the old model.

47:17 Participant Right, but only if I have it actively open, and active, right? So, I would have to open it up, and it’s active, and then I do something here, and it changes that as a model. Even if it was an old one, and lacks a lot of the additional changes, and so one. Yeah, that makes sense.
47:40 Participant  I was wondering now, OK so I am working in this collection here, and I want to take one of those (points at several alternatives in the Arrangement view), and start a new collection with that.

47:52 Interviewer  All right. Here we should have an option that would be named, let's say "Clone Alternative". That would be very useful, am still working on it and will add this. But, let's say, such an operation Clone, all the selected alternatives, that's something that is definitely needed here. But right now, you can just create a new alternative in this collection, and just pass all the items from one alternative to another. So if you click on this green "Plus" button on top... No? Still not working in this view. Just click on this pair of arrows in the corner, just to switch to the icon list view. Here, I think you can create a new alternative. Yeah. So this is an empty one. If you switch again to the arrangement view. It should be there somewhere. No. Y ou just have to close and reopen C-2. Yeah, this is still a little bit buggy. But, now you have C-2, now click on this booklet on top. So, what would you like to do here?

49:38 Participant  So, well, I was really thinking about starting another collection. I'm just thinking about hierarchies of alternatives, where one way of dealing with complexity is by grouping things together, right? One way of dealing with complexity in a view like this is by proximity. You arrange things by proximity, and this becomes immediately clear that I think those four have something to do with each other, and those four have something to do with each other, for whatever reasons. So proximity is clearly one of those things that we use to order stuff. And the other way is, of course, of dealing with complexity is by saying, oh, I make this group into one item. Some type of grouping. Which anyway you do with the collections, right?

50:38 Participant  So that's why the collections probably would have to become more transparent to, you know, getting things, alternatives, back and forth between collections.

50:51 Participant  But the other question is, can I have collections of collections, for example? I think, eventually, people will want to be able to, according to my experience with grouping, and those kinds of things, eventually you always increase complexity, right? And then you try to decrease that complexity again so suddenly you have groups of groups, or assemblies of assemblies, and I suspect, people will have collections of collections, depending on how they think about.

51:27 Participant  Let's say we are working on multiple projects. Do I want all of these things to be just one project, or should I be able to have multiple projects in here, depending on what I am doing, right? So, I probably might want to maintain multiple projects in here depending on how I can transport things from one alternative to another. If one of the alternatives is an assembly, I might want to be able to reuse that assembly in another collection, or another project so to speak.
But then, a project might have a different hierarchy in terms of, you know, how I think about it. What you intended with what collection means, might not mean the same to somebody who is working in this. I’d group alternatives within collection but then eventually I might want to think of collections as groups of alternatives, but I need to group them as, well, early designs, versus later designs, you know, stuff like that. Or, if it is a campus, I might have different parts of the campus where some things repeat and other things do not repeat. And I might want to be able to arrange those in an hierarchical fashion. And I’m not necessarily certain of how deep this hierarchy is. I think, theoretically, whenever you start having those types of groupings and collections, I think inherently the capability should be there that it’s hierarchically nested to whatever the users want. Because you cannot really predict how people are starting to think about these types of organizational arrangements.

53:20 Interviewer  All right. So it should be more flexible, perhaps a folder structure to the left. A typical hierarchical structure for collections?

53:28 Participant  Well, that could be the way to think of it. Right now you have thumbnails, you are seeing thumbnails here, that’s fine, it could be an expandable list. Either way, I mean, that’s already what’s available. And again, I look at stuff like this, people are familiar with this type of organization. And that familiarity is what people often then usually conceive as intuitive. So they see something like that, and say, oh, I understand how that works. So that might be something to think about. And you draw on familiar examples, here is the job that you want to do. If I can’t do what I want to do, then you have to think about new things, right? But, if they do what you want to do, and it’s already there, as an example, then there is a pattern... It’s more or less a pattern, these types of expandable lists, then suddenly you reuse them, because people would say, oh I know how to use this.

54:40 Participant  Drag and drop would be another thing. It’s something that people learned to do, dragging and dropping stuff.

54:49 Interviewer  We have some basic drag and drop capability. So if you just expand a little bit this middle area... As you can see we have three collections, so if you go to Window menu on top, and switch to subwindow mode. This is now a typical MDI interface. And now, drag and drop works for the time being in the icon mode. If you again click on those double arrows on top, and you stay here, and also the second one in icon mode, so you can drag now.

55:37 Participant  (Tries out drag and drop in the iconlist mode in Alternatives widget.) I move that (alternative) over here (i.e. a collection), but then I lose it over there (in the originating collection). Can I Control-click and copy? Yeah, that’s kind of, you know, people get the feel of it, it’s
something they understand as well. This is something they understand. That’s good, great. So in this case, I will be able to say, I take those two (alternatives), I select all of those. (Selects all the items in an alternative, in the parallel editor.) Can I get them (i.e. all the items in one alternative) up there somehow?

56:19 **Interviewer**  You can, we have this “Pass items” (operation).

56:23 **Participant**  (Passes all the items from one alternative to another.) OK, there we go. So, now A-12 is the clone of A-8, and I can drag it over... (Drag A-12 into another collection in the MDI interface.) Now I have A-12 in this collection, now I could start doing stuff here. Are the relationships still maintained between the collections?

56:43 **Interviewer**  Yes, they are. Ids are still preserved, but both of them are now in different collections. So right now you can’t see alternatives from different collections in this parallel editor. But would you like to have that capability? Would you find this useful?

57:10 **Participant**  Well, I am not sure.... Right now, I would say, I am not sure whether there is a use case for that. I’m just pushing the limits here. I am not really sure that it would be useful. It might be useful to be able to see two alternatives from different collections. Yes, I think so. Unless... I think about...

57:48 **Participant**  I don’t know how you conceive of collections. I thought, I see collections as a way to order my stuff a little bit. I still see those things as related to each other. So in this case I might have a way of thinking, one group of alternatives versus another group of alternatives, while you may have thought about that as simply being a way of how people order them in here, right? And this essentially tells you what that group of alternatives is, by having them appear up here and down here. But right now, then, you probably need some type of grouping, some mechanism in here so that you can collapse and expand.

58:41 **Participant**  But I thought of that (i.e. some grouping mechanism) as the collections, so the collections (i.e. the tabs in the Alternatives widget, where each tab provides one Arrangement view per collection) provide that kind of capability right now so I would use them like that, grouping some alternatives in one collection, and other alternatives of the same design into another collection. You know, by saying I have maybe two major branches or three major branches that I pursue, and within those branches I have various alternatives that I investigate. And then, thinking about that in there.

59:16 **Participant**  Now, one thing that comes here to mind is, from the way I am currently working with parametric design, some of the alternatives are implicitly embedded in the range, and
discretization of parameters, right? So, if I look at these alternatives, you know, this could be just the expression of the same parametric design, with different parameters. But I still might want to visualize them to see, OK, this is a parameter set to 3, and that’s the same parameter set to 5, and those are the two alternatives that I get by changing that parameter value. Right now, I’ve drawn this here manually, but you know, usually I would do that parameterically and say, let’s explore that, and maybe I have something that tells me what that alignment is, you know all those types of things, that’s parameterically driven but I might have said as a solution over here, change these things, and all of those things, and I want to look at those side by side. So one alternative... Let’s clone this alternative. So if I look at those two, and say, OK, suddenly I increase that... (Increases the density of a grid in the facade.) So of course I would want to set this parametrically, so that I don’t have to draw and delete lines all the time. Now of course, the same would have to happen down here.

1:01:37 Interviewer  So this means that objects embedded in these models should be completely parametric.

1:01:47 Participant  Possibly. But I want to be able to say, OK, here, whatever that parameter is, instead of having it at 120, I have it now at 60. Who knows what it is. In my mind, this is not a room, this is a facade. You mentioned rooms, but initially they could have looked like some laid-out rooms. Yeah, this is funny, it’s a nice little house. (Continues editing.) Here we go. Oh, I selected too many (lines).

1:02:29 Interviewer  You can do this (making lines thinner using the toolbar button), but you can also pass the value (from another alternative), right?

1:02:36 Participant  Well, I didn’t have this (set of items in another alternative)... Those were new (items), right?

1:02:41 Interviewer  Yeah, OK, but you can still select an element in the upper design, and just pass the value of the thickness of the line.

1:02:55 Participant  Oh, this one here, for example. I see what you mean. So, I could do that here? I have to activate it first, right? There we go. I see what you mean. And maybe we should... Ah, that’s interesting now.

1:03:13 Interviewer  Just to mention this, this is supposed to be very general. So right now we have three properties that can be passed, line thickness, line color, and line shape, but the goal is to be able to pass whatever property we have in a object.
1:03:32 Participant  Sure, I understand. I didn’t goof up here, but... Now I want to select this one. I selected one under the wrong end of the relationship here. And this one. Now we have what I wanted. There you go.

1:03:55 Participant  (After tweaking two alternatives; passing the value for line thickness.) So yeah, that would be the thing, that if this is a parametric design, I have alternatives exposing or expressing one parameter value in one alternative and the other parameter value in the other alternative. So I would want to drive some of those alternatives but just saying, OK, this is displaying your parameter value 1.20 and the other one value 2.60 or whatever. And, probably, have multiple parameters that I use at the same time to get even more differentiation. And, really experiment with alternatives along those lines, based on the parametric design that I have, that has, you know, N number of parameters where of course I theoretically get NxN type of grid of alternatives, or more, depending on the resolution of the parameter, and so on. I would obviously only be able to see a limited set of variants but those variants in my mind are not just manually drawn like I’ve done here but are expressions of some parametric system that I’ve set up where I say OK this is a parameter set for this alternative, and this is the parameter set for that alternative. And look at 2, 3, 6, 8 alternatives at the same time.

1:05:33 Interviewer  You know, these ideas actually keep floating around, and if you remember during your first visit here, we actually had a prototype that actually generates, having that so-called master model, a number of different alternatives, sampling the parameter space. So I think we had that model, S letter, and we came up with, like, thirty different designs for the letter S, by clicking on a button.

1:06:05 Participant  Yeah, I think I remember that, yes. And, I’ve seen, I am not sure whether you had you chance to look at VisTrails? VisTrails allows that in a way as well, where you select parameters and their values, and they do a grid of alternatives using that. In a way, that’s how I would think about this. But still, it would be great to have exactly those capabilities that you have here that you see those things are different expressions of the same object. And, understand what those relations are. Which of course, makes it a whole lot more complicated if you think about this really as a parametric system, and you capture those interactions, and apply them to a parametric system. Very interesting. So right now, if I work on this one over here (works on an alternative).

1:07:20 Interviewer  Yeah, it would be kind of hard to predict, I think, if you work with one alternative, how is this going to turn out for other alternatives being modified. So this is, on one hand, it’s an advantage, they will preserve ids, so whatever I did to one, is being applied to other alternatives. But, on the other hand, we don’t know what is going to happen, actually.
1:07:44 Participant  Yeah, that was the other thing that you mentioned. So right now, if I am working on this one over here, which for some reason... (Tries to select two alternatives in two different collections, which was not possible in the prototype.) I am not entirely sure how this one in the other collection will update it.

1:08:12 Interviewer  Right now, drag and drop only works in the icon mode. Now you are in the Arrangement view.

1:08:23 Participant  Right. But I didn’t want to drag and drop this. This is A-12, it’s the same... So A-12 is the same for both collections? Is that intended or is that just something missed up upon? You see, this is A-12, and this one is also A-12.

1:08:46 Interviewer  Yeah. This is because this view is still not being refreshed. So if you just close C-2 (in the Arrangement view), and re-open it, it (A-12) should not be there. (Participant reopens the collection.)

1:09:07 Participant  Yeah. OK. So now it’s... Oh, because I was kind of... I thought I was still working in this collection, and... OK. So I got this as an alternative over here. I go back to this. Let’s go back over there. So, yeah, what else do you want me to do here?

1:09:50 Interviewer  OK, so. Yeah. You performed a number of tests. I think now you have a pretty good idea of what we are trying to do. Again, we are grouping alternatives into groups, or collections, and we have this parallel editing feature here. The crucial feature of parallel editing is for us item passing, or passing items and passing attributes of these items. So, I think, you are ready to go to the fourth part of this study, which is simply a post-interview, your overall impressions about the prototype. Just a couple of questions about what you actually think about this prototype. So, what do you think are the most interesting features of this prototype? Do you see any value in features?

1:11:01 Participant  About most interesting features, and without any prioritization, I think what’s definitely interesting is to be able to design alternatives side by side and have a relationship between those alternatives. You know, having used it now for about thirty minutes or something, I can see the power in it, but I think the tool probably needs to develop a little bit more, in terms of, you know, if you really want to use it as a design tool, in terms of what it can do, and people would want. I would like it to be in 3D and that kind of stuff. But that aside, it’s like a new pair of shoes, in a way, that you have to walk in them for a while, to learn to appreciate them, and I think it’s a bit with this system as well. I mean, with the guidance that you provided, I think I was able to pick this up relatively quickly, but I think I also got the general trajectory of what is possible with it. I think I like it, I like the capability that makes it possible to say, OK, I sketch something, I can take
that sketch and maybe it should be really easier to make a copy of something, and then take that sketch further without losing the initial sketch, and having that opportunity especially if you think about this in terms of parti and so on to go back to that initial sketch and see, how changes in that sketch affect sketches that have further developed.

1:12:50 Participant However, what I suspect there is that I personally have gotten so used to parametric design that I would want to have parametric relationships maintained in there. (Parametric relationships maintained in the models.) So, especially if I think about the parti, and I'd think, OK, I have some axes (i.e. coordinate system) setup, and so on, and I move the axes, on of those axes is rotated, everything that feeds off that axes should then of course automatically update. It shouldn't be like, you know, moving this line here (moves the line on screen). If I move this line here, I would expect all the other things to come along and reredistribute and so on, similarly down here. So I could really explore, say, you know, take this line and play with that, and I would expect that at that point... OK, now I am curious, if I connect this over here, yes, it's easier already, because I can make that change up in the top solution there and it updates down here. At the same time I could also add additional lines. So that's already easier, but if it was, you know, with the dependency system I would have been done after the first move of that line. So that's kind of, you know, where I would expect in the end that that goes, which could be difficult depending on how you establish, what this (a line) is here, is this two representations of the same element or is this two elements that just know about each other. That could be a challenge. If it's two representations of the same element you might be better off. But it might be too difficult to establish this. (...)

1:15:21 Participant But yeah, I like this idea of being able to look at variants that quickly and easily.

1:15:33 Participant Still, what's confusing in this window (parallel editor) is the right-most button and in this view (arrangement view) is the left-most button. (Participant refers to the fact that one needs to right-click-and-drag in the juxtaposed parallel editor to pan, and left-click-and-drag in the Arrangement view.)

1:16:04 Participant So now you cannot see, since I changed this here (points at an alternative), because I hadn’t made a copy before, I lost the previous version, and I might still want to keep that around. It’s really important to say, OK, make this... yeah, that would probably be one thing, you know, if I branch in my.... Essentially, each of those is a kind of a branch, they are related to each other, right? So this is an early version than that, than this, and this is a kind of branch of that. So these may or may not be branches.... It might be something like that if you look at it.
1:16:44 Interviewer Actually, you are correct here. We keep calling these designs, or alternatives, but they are actually head versions of branches. So every alternative in this collection is actually a branch. That’s correct.

1:16:59 Participant Yeah. So a little earlier, before I made this change here, I realigned this (line), right? Probably I would have wanted to say, now I do a branch, a new branch (from an existing alternative, in the Arrangement view).

1:17:18 Interviewer All right, like forking, right? Would you find any value in having built-in version management, or version control, in such an application? Like seeing a tree of derivation of designs?

1:17:38 Participant It could get confusing, right? Having both branching and versioning. But I guess, in a lot of development environments, that’s what’s done, you have branches and you have versions within that.

1:17:58 Interviewer So, might be useful, but could increase the complexity.

1:18:04 Participant Yeah, might be. I think the first thing would be to figure out how this could be developed into something that says, oh, you know, there is a development in terms of... not in history, but saying, this is maybe the trunk, and those are branches. What is parallel to each other, and what is.... This is certainly an early sketch, and this is a further development of that sketch, and this is also a further development of sketch. But this (i.e. another sketch or alternative) is not necessarily a further development of the sketch, right? At least in my opinion I think of this (points to an alternative in the Arrangement view) as one branch and that as another branch. And in fact, I really would have wanted to keep around and I didn’t think about that earlier, that version where I had it like this (modifies an alternative).

1:19:07 Interviewer So, if you had version control, you could go easily back to the old version, or you could just create a new version. That’s the other alternative for those.

1:19:17 Participant But the thing is, I really then wouldn’t want to go back and lose that new version, right? It could be tricky there saying, I want to go back but I want to keep the version that I have already. So it would be different than an undo or... You would essentially reinstate previous version, without losing the current one.

1:19:44 Interviewer Yeah, that would be the case. So if you view this point of view, we are actually dealing here with configuration management. Because we are preserving ids across design states, therefore all these icons that you see here, or alternatives, actually represent configuration
management here. So, you know, the fields kind of start blurring. Version control, configuration management. This is perhaps actually a major point here, we are trying to blend design exploration, with configuration management here.

1:20:25 Participant Yeah. I think it will be a challenge to make that obvious to people. (Continues working on an alternative.) Now I have those copies. Now I want to pass item there. (Keeps editing.) So now I have those two versions. I have to update this one again. There we go. OK. Also, I have these two versions. This was easier than usual. So in fact we have kind of this pair, and this pair over here. And it's interesting, there is a relation between those two, and there is a relation between those two, but there is also a relation between those two here.

1:22:48 Interviewer That's why we keep talking about configuration management here, because, you know, we could just imagine this in an improved version. For example, these are two versions of Windows, an operating system for example. So one version has a server component, that component is modified a little bit... But we are still able to maintain them as a family of products.

1:23:22 Participant OK. I think you are certainly on to something here. It's very interesting way to think of it, and, yeah. So being able to work in those... One thing, you are seeing the selection of alternatives (in the parallel editing area), that's pretty good.

1:23:50 Participant One thing that could become a challenge and we talked about that, as alternatives proliferate, keeping them organized, certainly the graphical arrangement in a collection can help with that. But, you know, if I just think about that, OK, now I develop some of those further, you know, I really get into a big mess I guess eventually. So that could be challenging. Without having the ability to say, ok, maybe I can group one branch and another branch, but I'd still would want to be able to compare across branches in that case.

1:24:41 Participant So I can think of the branches in this way, that these two are in one branch, because they have the bigger wall piece here, or I can think about them like this, because they have the... Well, it's not really a punched window but, your know... They have a window or an opening here, surrounded by wall-pieces while these two go to the edge. So, there is some overlap here in terms of how I think about those as pairs. So those two could be a pair, but those two could also be a pair. Also, those two could be a pair. So, suddenly it becomes a little bit complicated saying, ok, now if I carry one of those further, even if I have grouping, I would still want to be able to compare between groups, individuals between groups, or individuals between collections. Saying, what I have arrived at here (points at an alternative), in this case, and compare that to what I have over here (points to another alternative, in another collection). So I think, the parallel
editor should not be depending on the organization that I have on the other side so to speak, and those collections and within the collections. You know, one gives me how I order my designs but then I want to compare all of those.

1:26:27 Interviewer I see. OK. So, parallel editor should be divorced from this notion of being able to show just alternatives from one single collection.

1:26:37 Participant Right. The parallel editor should afford me to go across everything. And the things are not related to each other. I think parallel editor is one thing but really also being able to just pair any alternative and look at any alternative, it's the other thing. So in a way, it's true, this thing has the power that I can edit stuff in it, but at the first level I just think of it as something that I can look at at the same stuff, at the same time.

1:27:15 Interviewer I see. But, you are still able to see them (the alternatives) like this in the Arrangement view, right? As you drag them around, arrange them as you wish. It's perhaps too small, or it's just the thumbnails. Would you like to have them bigger?

1:27:37 Participant Well, that's probably a matter of how I arrange the windows and stuff. So then I would have to resize them to get some similar sizes. But viewing in the parallel editor I see them, that's the other thing, I group them with a different logic. I am not so sure about the panning and zooming in here (Arrangement view; participant is jocularly refering to non-standard mouse wheel action). So while I have fun arranging things in here, where I can easily look at those two side by side, and those two side by side, if I want to look at this one and this one side by side, I either would have to rearrange them, and that destroys the arrangement that I have based on the logic that I saw in how I arranged them here, because this logic is not the logic of seeing them side by side. This is some logic that I apply (in Arrangement view), in order to understand my system of alternatives. So in my case it is going from the simplest sketch, or simpler versions to more complicated versions, those I see at the same level in the development, just different expressions of some ideas, but I still might want to look at things that are not directly side by side and see them side by side, and have the affordances of this view here (parallel editor) which is how I can zoom in, and so on, and if I want to explore relationships I can get to those as well. And if I need to edit them, I can do that as well. I think, that's great, you are having this view (parallel editor) and then you are able to go and edit in it, that's great. But in order to look at those two alternatives side by side, I don't want to break up the logic of Arrangement view, or the logic of collections, if I want to see something out of this collection, and that collection, side by side. And manipulating those two windows (two MDI arrangement view windows) is infinitely more complex than just selecting here, selecting there, and having it over there (in the parallel editor). I like that. It's so simple and straightforward to say, you know, you just select it, and there it is, and you see it side by side.
1:30:09 Interviewer What do you think about having in-place editing capabilities directly here in the Arrangement view? Like, for example, just double-click on this one, and, you would be able to edit it in-place.

1:30:30 Participant Well, I can see people wanting to be able to do that. They always want to do everything, everywhere. That was one of Rob's arguments, with Maryam programming in the model, everything should work everywhere, because that's what people than expect, artificially limiting where it works and where it doesn't work, because people will get confused. And that type of rigor also makes things less intuitive. Yeah, if I can edit here, why shouldn't I be able to zoom in and edit there. It looks the same, right? And I understand why not, in this point, because you think of those at this point of time as thumbnails, rather than as full editing windows. And certainly, that makes sense, but in the heat of the battle, people will forget that, at some point. Even if it seems easy enough just to select it here, and edit that there. But people always want everything. If you can stomach it, if that's possible, do it. They would want to do multiple selects and get the benefit of parallel editing even though they are not in the parallel editor, potentially as well.

1:32:12 Interviewer OK. Let's move on to the second question. What do you think are the least interesting features of this prototype?

1:32:26 Participant I haven't got much use out of Collections window, rather than opening and closing the collections. So I am not so sure whether this could not be simplified into one, and take up less area. (The experimenter is showing off some capabilities of Collections window - filter by name, etc.) Maybe those are capabilities that we haven't used yet in this session, but still at this point of time as you asked me, ok, really if I have access to those collections, I could live without that, at this point. The filter would work different in that case. Least interesting... Well, right now, I think that because of this minimal implementation that you have everything is kind of important in it, right? (Shows at thumbnails in Collections and small Overview window in the arrangement view.) See, this (small Overview pane in the Arrangement view) and that (thumbnails in the Collections widget) seems to be a duplication as well, you might be able to be more economical about it, at some point. Overview might become interesting once I zoom in. It always shows everything that is in the collection. OK, here I can zoom in further.

1:34:52 Interviewer If you can envision in a much more developed and polished version, how would you work with it in your practice? If I remember well, you are an architect, right?

1:35:08 Participant Yeah, I studied architecture, and worked in an architectural firm, modeled in a technology hall, but I've seen people work and struggle with all kinds of software systems trying to do their job. Now, I think, one of the things that comes immediately in mind is, how this will work
if I have a design team and design team works on variants and variations of a design. Because
I think inadvertently that, well, since I've worked in a larger firm that's always been the case that
there were design teams not just one designer. And I think, that's one of the next items there,
because as I am thinking about these alternatives, I really would also want to communicate and
think with the rest of the design team about those alternatives, and see how those develop and
then, you know, other information flows into this tool and into decision making about what these
alternatives really mean.

1:36:28 Participant  On one hand, it's current simplicity could be its strength, and probably, pre-
serving that as much as possible would be formidable goal by really making, still making everything
accessible. Right now, it's just 2D drawing, but in the end it should be a full 3D parametric model
that I could develop here. That's how I think I would want to work with it, you know, saying, OK,
I have certain parametric behaviors in that model, looking at those alternatives, and put editing
things in those views to see how that would affect different variants of some model, that would be
highly interesting in exploring designs, and potential of designs.

1:37:31 Interviewer  So visually observing how multiple parametric designs change, perhaps in
real time.

1:37:38 Participant  Yeah. Depending on how all this is programmed, if I say this thing gets wider
or higher, whatever it is, there is some logic to both of these (points at two alternatives), then it will
propagate those changes and make those behaviors, and if I throw in other variants that would
be developed over time, changes would be different. Some of that, that could just be the validity
study, if I have some building based on some, let's say some site, and I know that that site is going
to change, I want to apply the same scheme to different sites, and want to keep that potential
open. If I exercise a design only based on only one site, I never get to explore whether it will work
and be valid, at the other site. But if I can do that side by side, I would have that opportunity. And
that's things suddenly can happen here, where I can see where might thought-out behavior that
I thought would stretch across all the possibilities and all the parameters, might in fact not. And,
certainly, I cannot discover it, and this tool won't guarantee that I discover it, but at least for the the
two or three that I am very interested in I can discover it, right? Which before would have been
much more tedious. And that's a big advantage.

1:39:17 Participant  Now the other thing of course that might be interesting in these variants
would be to add additional information, so if I'd say, ok, have an additional dashboard here (in
the parallel editor) that gives me more information of what I have in my model, be it floor area,
whatever. The metrics that can be computed. If I could have those somewhere, with the variants
to understand those effects. Here it would be things like glazing area, for example. How much
glazing do I have in here. How many running meters of steel profiles I have here, whatever I want
to understand about those designs.

1:40:00 Interviewer   Yeah. Something like a property editor, for each selected component.

1:40:04 Participant   Less than property editor. In this case, I really think about the reporting tool,
right. Because that's how people do understand variances, in terms of how they perform, in some...
not just visually, what you see right now, but in some measured quantities. Ultimately they want
to know the cost, energy usage, and so on, but let's just stay with what is interactively possible,
right? Like if I just have simple quantities here, that already could be indicative of something. But
ultimately, I should be able to say, you know, if I think of this as something where I have... if this is
a full fledged parametric system under there, I could compute whatever values I want. I should be
able to have some area here where I say, ok, I want this variable displayed here, or that variable, or
so, right? And those two alternatives, that's suddenly with some additional information, whichever
way I want to display them, color-coded bars, numeric values, whatever I want to do to give me
a brief overview if I look at those alternatives, to see how they compare also in other things that
I cannot see just by the visual. That's definitely that I think would really be tremendously helpful
and I think people will probably have already mentioned this to you.

1:41:46 Interviewer   Yeah, additional ways to display our information. OK. What do you think
what barriers does it present right now?

1:42:04 Participant   Well right now, the reduced toolset is of course a barrier to my work. But
that aside, some of the functionalities that we talked about, like being able to group things hierar-
chically and organize them differently, I think that flexibility should be there because even within
any practice you have different projects where the needs are different, so that might be really
something where you think about the whole project and big sets, what might be collections, but
then it breaks down into subsets... Any of those preconceived ordering systems that are not in-
finity hierarchically nestable, are suspect to me because people may want to think about just one
building as the collection of stuff, but then, some of those alternatives just might deal with details
in that building. (Participant here refers to having a hierarchical decomposition of objects.) And
so, that might be probably also be a capability to think about. Right now we are seeing the whole
thing here, all the time (points to the Arrangement view). Could I just take two of those (points to
two lines in an alternative), and move some forward (i.e. looking into alternatives for just those two
lines)? Probably I could, right? That would be definitely something I think in the long run I would
be looking at. You know, I say, let's see, I selected.... (Works with an alternative.) And now I want
those two (alternatives) to be the basis for something that I build up on them in the parametric
dependency. This is probably not the best example but, you know, if I am really interested in the
detail of this corner up here (points to the corner of the designed house, while panning both alternatives simultaneously) I might want to have a few alternatives of what happens up here (in the corner). This is a bad example because here I really want to... nothing much to happen. I hope you understand what I mean here.

1:44:50 Interviewer Yeah, isolating instability a little bit.

1:44:55 Participant Okay, yes. I’m just saying, I’m exploring specific alternatives just on parts of the building, right, because I don’t want to re-sketch the whole thing all the time, right? I really think about it in the assembly fashion, that could be an interesting way also to say, okay, I have alternative assemblies. It’s a matter of subdivisions. Or, you think about, I don’t know, facade, or let’s say interior, I might different bathroom modules that I want to insert and explore, how they are arranged in a high-rise building, to make a core, you know, all these types of things. People have to go through rather mundane alternatives sometimes, and then they want to take that idea of the core and apply it to a different high-rise building, right? So that type of usability, I think it’s kind of implied in here, but maybe it’s not...

1:46:16 Participant Maybe it’s being back to the idea of patterns, where you think of, let’s say I’m a firm that does high-rises, and thinking of the core in a high-rise building as a pattern, probably would be appropriate. I could think about how to skin the building in various climates, that could be different patterns of facade, elements. Lobby could be a pattern, those types of pieces that I can apply, what type of high-rise, what level of luxury the client wants, and those types of things. Hospital design, similarly, there is a lot of design in how operating suites work. And those could be, you probably want to design alternatives of those, and then move them forward in two bigger aggregates of operating suites, and figure out how those two work with each other, and then be able to change the module grid / column grid and see how those still fit and react to those changes, and so on. It can get quite messy, I think, and what the dependencies are, and how they might propagate, and so on. I am not sure, if you want to think about an underlying parametric system here, whether that’s multidirectional constraint system, or whether it’s something that’s more like a directed acyclic graph, similar to what Grasshopper and GC do, you know, all have implications as to how this could be really useful, and also not just managing alternatives and versions, but also how to manage assemblies and so on.

1:49:44 Interviewer Yeah. So the second-last question is, how would you change it? We touched on this a number of times, but if you have anything to add. So, how would you change it.

1:50:09 Participant (Participant maximally expanded the parallel editor full screen.) I can see why there is probably the synchronized view... (Refers to 2x2 windows in the juxtaposed parallel
editor.) I expanded it to have dual screen here, just to experience it differently I guess, because now I can have a bigger collection on my other screen that you don’t see.

1:51:04 Participant How would I change it? Certainly, being able to select cross-collections, I think. Just thinking about the whole idea of collections, thinking of that different in terms of really just a hierarchical ordering of your alternatives. And, being less rigorous about making that hierarchical ordering something so fundamentally different. (That is, the MDI windows representing collections look much more different than the alternatives.) A collection looks very different than an alternative. But perhaps you can think of all of those as containers of containers and then they are eventually the nodes that contain, essentially the views of alternatives so to speak.

1:52:04 Participant I really think if I am in this view (Arrangement view) this should be it. I just say, OK... What I have over here (in the Arrangement view), I should just be able to drill down and then, you know, be here, so to speak. I should just be able to drill down, and not have so many different views.

1:52:45 Participant Certainly you would also want also to be able to back up and navigate across (collections). I can see that. But I really think about it more in this type of browser fashion, where you have this type of view (opens Windows Explorer), and these collections are essentially those high-level folders. And then you can drill down and eventually this would be the alternative. You wouldn’t have this second pane here. (Refers to the detail pane in Windows Explorer.) And then, if you select this alternative, it would be here (in the alternative). And if you select a second alternative, which you can not do in Windows Explorer of course... But if I could select two of those (folders) and at the same time, then they would be here. But that's what I would envision, that I could select any two or more nodes in here. And if there are no nodes I would probably ignore them, or if I clicked on them expand them immediately. And then populate the, I wouldn’t call it a parallel editor, I don’t know what is a parallel editor right now. This comparison view, whatever it is. Viewer and editor. Populate that from my selection, and really leave it to the user to structure what his hierarchy is. The users are used to do with a lot of stuff, like you know, highly strangely organized file systems. And we all learned how to do that, more or less.

1:54:44 Interviewer Yeah, it is contingent but people like it. And it is flexible.

1:54:48 Participant Liking - that's not necessary, right? But you know how to work with it. That's the thing. And they don’t have to learn a new metaphor, a new view of how to work with stuff. If they can apply existing knowledge that they have, that's what gives you intuitiveness. Because, they say, oh I know this, I know how to work with it. And that comes from recognizing things that they’ve used before. So, drawing stuff from operating systems or what browsers do, is good insofar
as you essentially draw from experiences that users have already made. You want to avoid those
that they bitch about, that’s for sure, but they don’t need to like it. If they feel familiar with it, they
can work with it quickly and they might like it because of that. Because if you want people to like
it, then every person likes something different. And so you lose that edge where you can say, ok,
I am drawing on the experience that probably 95% of my users will have.
Figure 13.6: Event categories for participant P3.
Participant P4

27:35 Participant It’s interesting, you know, element IDs in a DGN model. I’ll just speak in Microstation and GC terms, if that’s OK. It’s interesting, because you have different elements in different models, but in fact what you are actually having, you said, same element ID, same object in multiple alternatives, as opposed to multiple alternatives with multiple objects in it. It’s a single thing that shows up in multiple alternatives, yeah?

28:03 Interviewer Exactly, yeah.

28:05 Participant So we are supposed to have different things with different IDs that you just synchronize with each other, although that has basically the same effect. A different approach to the problem.

28:17 Interviewer OK. So you are saying, you have different objects, and you would like to somehow link them actually.

28:23 Participant No, it’s just a general comment, it’s very interesting how you maintain the identity of an object across multiple alternatives, especially when some of the operations can destabilize the IDs. The real obvious problem is the geometrical operations which could say change which is the start point of a profile. And then you do an extrude operation and now you face IDs of the solid that’s created. It’s all messed up because in two different alternatives you have different numbering sequence that applied to the profile. To the eye they look like are the same but to the computer they look very different. So, how do you get that to be stable from the user’s perspective? It’s an interesting problem, and in a product like Microstation it really does rely upon the IDs of things and the start points of things. But that’s fragile. And in GC, it was thrown away in favor of what you call selector points, right? This whole thing where, where you fill in an edge, you do not pick the edge, in fact you create a point. You place the point in proximity of the edge, and say, fill in the edge closest to the point. So the IDs change, of course, it doesn’t care it’s still picking the same edge. But my only point is, this really does expose the stability of the relationships across the alternatives. If you implemented this that could be the source of tremendous joy or tremendous frustration, you know.

30:09 Interviewer Yeah, definitely. Because, as you said, joy, because these IDs, if you preserve them actually help you, if applied in a operation. Let’s say you have different doors, or door models. So you change the shape of one door in one alternative to, let’s say, another shape, French doors, whatever. But, since the IDs are preserved, whatever you to the doors in one model, is being applied to the door in other models as well. But you said, joy, the downside of this could be that, the results of any operation could actually be unpredictable.
31:01 Participant  Yeah. You’ll hit all kinds of corner cases where it wants to blow up.

31:08 Participant  Thinking about the problem, I am correct in putting myself, you’ve got just lines and arcs stuff here, but I assume I am to imagine that I would be doing both 2D and 3D parametric modeling and that, potentially, these 2D views are 3D models based. Is that the correct assumption?

31:32 Interviewer  That’s correct, yeah. This is just our playpan to play with ideas, actually.

31:40 Participant  So putting myself in that position, I’m trying to imagine what the domain is. It was smart to show me examples from the architecture so I can imagine that’s the central domain of the objects that are being modeled. And maybe, not even, some particular mechanical part. There are other domains with other sets of rules, but I thought about this matters more about the semantics of things, like... I don’t care what the door ID is. Doors have numbers. For example, your office has a number on it. That number in fact is really important because all your mail needs to get there. But if you change that number, so long as you notify the post office it doesn’t actually matter what that number is for as long as the relationship is maintained.

32:29 Participant  So, in going from CAD to an object oriented system, the big thing is differentiation, the semantics of the thing. You don’t deal with a door number, you actually say, this is Sinisa’s office, and office has a door. And so, all the different models I happen to know, no matter what it looks like, no matter what the IDs are, it’s Sinisa’s office. It’s like orthogonal to the geometric model being drawn, but it’s just as important in understanding what the parallel alternatives are. I guess just like the other representations of it, not just the drawing elements but the catalog of objects, the dictionary that accompanies those views, would be just as important in understanding what the alternatives are, and also in terms of making sure that you keep the relationships established among the different alternatives.

33:47 Interviewer  Yeah, I agree with that. (...) I keep using this term "ID", but this is perhaps a little bit too technical. So, the point here is that, the identities are being preserved. As you said, Sinisa’s office, Makai’s office, and this is enough. I replicate an item, if this is Makai’s office, when I replicate it into this alternative, this is again Makai’s office. And once I do that, for example I deactivate these two (investigator mutes two alternatives), and I drag this around (drags a line around the canvas), of course the properties of this element, or object, are being changed, and if I activate this one (alternative), again, I’m dealing with the same element but its properties are now different.

35:13 Participant  Right. So, you moved one of the end points. What happens when you move... I should probably try this on mine (prototype). I have an end point and I move it. Now I deactivate
the bottom. I am moving the others. Now I reactivate it. What happens when I move the end point? Oh, it keeps the relative position! It's now moving it, but it's moving, the two different views had a different start point, because they've taken out of synchronization. Which raises the interesting dilemma, of whether or not you, when you are moving an end point, are you intending it to be relative to its existing location, or did you in fact mean absolute position for that object? So, in all of the worlds, it would have to snap immediately to the... As if you had first re-replicated that object, to refresh it, and then... I guess it would be the same thing as if you took one of them that you wanted to be the winner, and then push its changes back up to all the other alternatives. So you can do that right now in your current prototype but it is a kind of workaround. I guess I stand by my initial reaction which was, it’s interesting, relative versus absolute when you go to edit something and it’s already now different. If you look at how you are coding something, and you push your changes, would they get pushed to either the other people’s branches that have yet to emerge back into yours, you might not even know what you are doing to them. So I am looking at this one but the other ones are still changed to this, and it’s moving in a different way. I would be glancing really quickly between the two of these to see whether or not... I might align this one up but the another one is now out of sync and I misaligned it. Ooops, let me turn that off. Yeah, the interactive (parallel) editing requires like another kind of hand-eye coordination I’ve probably never practiced.

37:33 Interviewer This is interesting. So, what do you think about it?

37:37 Participant I don’t know! I would not try it in a non-trivial domain. I guess I probably wouldn’t ever draw this way. You are working more... data driven. I think I might exceed my human capacity, I really just have to... Like, I am actually editing one alternative at a time. It’s not that I need to edit one at once, it’s that I need to be aware of all the other alternatives simultaneously. It’s like the old Aalto quote, of being simultaneously aware of these things at one scale smaller and one scale larger, and then you work it. It’s Aalto Senior. As a kind of urban planning thought. Like if you are, moving one door from one wall assembly to another, you have to be still be thinking about what are the detailed scale, what the intersection of the door frame and the material is going to make. Because you will have to flash that later. Actually, you are looking at the door higher than the door because you are looking at two adjacent rooms or a building against the outside. So you have to think about all three of them in order to make any decision at all. Which to me is kind of the point of the multiple... superposition. (Points at the superposition parallel editor.) I guess that’s what that is, isn’t it.

39:10 Interviewer Yeah. Parallel editing is a little bit tricky. If you remember, architects, designers... they all have walls stacked with models, with drawings, and stuff. So we are kind of trying to replicate this here, but that thing that you said, that you would actually be interested in editing
one, and then observing how it applies to all the other alternatives, this is interesting.

39:52 Participant You know, when I think about it like tracing paper, and it’s almost comforting to think of tracing paper because it’s like something familiar compared to something like this which, yeah, could induce vertigo. The bottom one (superposition editor) is actually the one that I would probably want to most of my editing in (as distinguished from the juxtaposed editor). If I could, because that’s where I am simultaneously aware. Most of the time I use tracing paper by, you know... OK, you use it to make changes to something iteratively where you copy everything but one little part, and then change that part, and it becomes the new version. The other way that you use it to separate out parts of things, even sometimes so that you can dim them by putting them in back. And for that you actually do keep the other alternatives around, you’d say what I’d have to to do get a bathroom here, what if I do the bathroom over there? Then you get them together, take a look at it and I actually, if I could reach into both of them, and move stuff in them simultaneously I would, and so the first thing I would do with this multiple graph, is to click on the one that is gray, to make it active. I would probably even want to be able to, click on the one that is black and make it gray. So if I could reach into space as it were, reach into alternatives and go back and forth, I think the simultaneous space is where the creativity and productivity is. And the juxtaposed, individual edits are almost like detailed views, they are less powerful but they are more precise.

41:51 Interviewer This is interesting. But for editing, you would prefer the superposed view (i.e. superposed parallel editor)?

41:59 Participant Yeah. And you know, this is formZ’s ghosted objects, and these are the other layers of traced paper. And they are also Photoshop layers when I click, sometime back they introduced the ability to have cursor with the arrows, but then you used to move your selection. It will jump to the selected object, it will activate the layer of which the pixels that you clicked belong. Which is a feature you have to able to to turn on and off because sometimes you turn off layers to protect yourself from picking them. But if your intention was to get to them, how would you pick that layers first before even picking them is a major annoyance. So they had to make them a mode between either one of them to protect myself or give me the power to flip between them. So I would run through these features of Photoshop, because it would get over this, making me go up to the other windows to toggle them on and off.

43:15 Interviewer OK. Just to continue with the features of our prototype here, I will mention this ability of passing elements and their properties. (Investigator explains the concepts of pass item and pass property.) (Technical issues with the teleconference call up to 1:01:28.)

1:01:28 Participant That’s nice. Selection, brushing, that stuff is actually very helpful. GenerativeComponents (GC; a commercial CAD package) relies on parallel selection. Your ability to
12.5. CLOSING THOUGHTS

identify one object in multiple views. Highlight, as the mouse passes over it, is a sufficient cue. So simply moving the mouse around the view, and watching the elements flicker in multiple views, is enough to get one’s bearing, as it were.

1:02:25 Interviewer  Yeah, it definitely is useful, even with two or four alternatives, and especially if you start working with ten or fifteen alternatives.

1:02:41 Participant  Can you do it again? I just want to watch for something. If you can go to one of the views on the right (juxtaposition parallel editor), pick an object there (a line). Can you make one of the views (in the Arrangement view) larger? Now do the same as you had before, now go and pick one of the objects, or mouse-over it in one of the individual (views). Move one, let's say. The power of that feature would need to be made in dynamics, so that the thumbnails on the left aren't actually just image thumbnails that you refreshed upon the completion of dynamics. They would need to done in dynamic, as should be mouse-over, what you just called highlight, not select. Because I think the parallel selection thing is really, it gets at this essential problem of identifying which object is the same object in a bunch of places. And, that's no less true than when you are looking at your whole collection over there on the left and you select one of the things on the right. And you are there too, you know, you are picking it, you are redrawing the image correctly. But I think, that's just absolutely incredible. One time we broke that in GC during our migration from one version to another. What's when I learned that lesson, because we didn't notice for a minute, then we started to use it or show it to someone, they would be like, what's this thing, I can't use this at all. And that was all what was broken about it, was the parallel highlight, that wasn't working properly. Can I talk about something that keeps coming to my mind, as you show this to me, or do you want to try to get through certain amount... Do you want to finish showing me this version, or another one?

1:05:28 Interviewer  No, as I said, any input that you have, any thought that comes to you, feel free to express it.

1:05:37 Participant  So while you were talking, I drew something on my tablet, and you know you do a diagram sometimes, it's a four-square grid. What it's really comprised of, is an x-axis and an y-axis. So, across the x-axis, I wrote Worlds, and across the y-axis, I wrote Representations. So, GC maintains parallel representations of the same world. GC maintains one world, across multiple representations. Your work created single representation, across multiple worlds. And where it would get really interesting is when you have more than one representation across more than one world. So, for instance, you have a tabular view, and I can pick on one of these worlds, but what I don't have, is a view of a single object, that cuts across the worlds. So if you for instance gave me an outline, a tree, some sort of an non-graphical representation of the objects, that would
be the second representation, and that’s maybe the difference between say a graph, flow diagram, and the graph dictionary. Or, a hypergraph, or a hyperlister, as it would be in Maya. It would really be an interesting view of this because the leaves of that tree are your objects, but they would probably proliferate, as one object has like two states, another object has seven states. So I am not suggesting that a tree view is a particularly efficient representation of alternatives, it isn’t, it’s a terrible way to look at it. However, they user looks at it from the semantics of the thing, Sinisa’s office. But I want to go and say, show me Sinisa’s office as a line of text, but now as it’s proper name. And then tell me that there are six alternatives, please open those six and arrange them quickly for me, diff them for me and report the differences. In the end, I am not so concerned about six views, or eight views, or ten views of a bunch of collections. What I am actually trying to do, I am trying to say, Sinisa’s office is either going to have a window, this one got no window, this one is in the basement because he failed his thesis defence and got thrown out of this corner office upstairs and he got bought out by venture capitalists. It’s the object itself, so I’d really like to see the objects abstracted from the alternatives. Like I said, on the axis Y, the objects objects themselves in a non-graphical representation is an alternative representation, next to a geometric view. GC has some tools for, depends on whatever, it has different ways to look at the same objects. So that’s the Y axis. And the X-axes are how many different alternatives there are. So perhaps this is missing - just bring in other representations, the ones that I really need and come to rely upon in a complex data-driven parametric environment. Microstation now has really grown what’s called Explorer which is a tree view of the objects in the model, and it is used to not just the levels, and models, and files, you can search it, you can see how many fittings there are, all that stuff. And it just goes to show that even what was once a conventional CAD platform, has evolved to be a lot like GC where you need a catalog of objects.

1:10:34 Interviewer I see. OK, if I understood you correctly, you are actually referring perhaps to component-based libraries, in such an interface?

1:10:55 Participant My observation is that the object-oriented programming has taken over the world, object-oriented systems design, and really CAD now, with BIM, everybody is beginning to think of them as objects, as parametric objects. At the end of the day, sure I may arrange the lines on a page, and do still have to draw details. And I still do interact with my objects as 3D things, and push them around. And, increasingly, I think it’s gonna be the data on those things that drives the line, like the end point of that line is only there because I placed a parametric constraints and it grouped it there. And it’s really the parameters that I am trying to find the differences of. So, with some sensitivity analysis, changing those parameters. I’d love to see a mashup of what you’ve done with those other parts of the parametric system.
1:12:20 Interviewer  So we have a couple of mockups as well. (Investigator demonstrates several wireframe GUI mockups to the participant.) This is actually branching, and this is interesting. Here we have again a juxtaposed editor, and at some point I’m actually supposed to be able to have these windows (points at the Property Editor) as additional views into the model.

1:13:14 Participant  Branching is interesting. The branching in the graph is interesting, because we mention this all the time in GC as kind of a pipe-dream of something that would be wonderful to be able to do.

1:13:30 Interviewer  Actually, if you look at all these alternatives in one particular collection, of course you can think about them as alternatives, or instances of your current design state, or instantiation of your parametric model. But, you could also think about them as branches that you have in your collection, and what you see, are actually just the heads, or the most current versions of your branches that you have in your collection.

1:14:08 Participant  You know, in the mind of the designer, it’s easy to slip back and forth, and it’s clearly good to erase the differences between taking one thing and changing it over time, versus taking one thing and changing it simultaneously, that is, like I moved it this way then I moved it that way, then I moved it that way. Well, right now you have all these different states, where one lead to the next. They represent both moments in time and simultaneous alternatives of something, they do both of them at the same time. And I think that’s why, you know, early on, (a designer, unintelligible) got ahold of Maya, and started doing, you know, design studies using the animation system. And he is doing the animations, and design alternatives studies at the same time, as you just exercise it through time, and then look at all those states, then freeze one and decide that this represents your thing. You can go concretely to one or the other, but I think maybe to do so is, like, too rigid.

1:15:30 Participant  So, really neat. I look forward to seeing the results of your study with users. And I would love to what to make of this, where you want to go with it, because the end domain in which you applied this will have a lot to do with what kind of the right answer is. Because if you are doing CAD drafting, the way this feels as a little bit of prototype just because right now you are just working in 2D. This has a very different set of what requirements it places on alternatives, what are you using the alternatives for. It’s sure that, for a manufacturer, alternatives are the three different doors that we can produce, the table of variations that I can put in suite’s catalog. These are my alternatives. To an architect, they are, client likes this one, and didn’t like this one, this one is more expensive, the alternatives themselves are evaluated under another set of criteria. The analysis of alternatives is naturally optioneering it. You know, alternatives to a factory, this is the way we will configure it, and this is the way we configure it that way if the price of this stock changes, and
I need to reconfigure it. What I need out of alternatives, is even specific, within ABC, across the different application domains, you would get a very different system, if you try to apply it to these particular use cases.

1:17:35 Interviewer Yeah, I agree with that. For the time being, we are trying to keep it simple, perhaps plug it in later on into a 3D modeler. We are just hoping that we can get our ideas across somehow, perhaps in the simplest ways.

1:18:01 Participant I can imagine that you can misuse the Microstation platform quite nicely to throw up a bunch of different models, and synchronize them. You could sort of play the same game that GC plays, and try doing it with data objects in a few different domains. It would make a really nice proof of concept.

1:18:35 Interviewer This is mostly it. Perhaps I should just mention here the novelty in this prototype, these little buttons that you can see (Solo/Mute buttons). Are you familiar with audio mixing consoles? (Investigator explains Solo/Mute buttons.)

1:19:37 Participant That's nice, I like the metaphor, it's something everyone can understand, they get it really quickly if they don’t have any experience with that. I'd keep the metaphor. Watching you do that, and thinking about a band, and the fact that a band is a bunch of people. I am trying to imagine that what you got here is the song that they are all playing. I thought that each window might maybe be controlled by a different person and I kind of wondered... There is the collaboration aspect. Because, you like everybody else, have remained trapped in a one-user, one-model system. So that is one admonishment. The other one is, that Rob himself had once talked about that push-me pull-you (collaboration model), four different states that a collaboration system can be in, right? (Participant talks about a paper done by another member of our research group.) You have the same problem along your views. Do I want the change to be pushed over, do I want to push the change manually, etc. It is a kind of little collaboration system. You have the same change propagation issues.

1:21:39 Interviewer That's an interesting remark. What do you think are the most interesting features of this prototype?

1:23:40 Participant The most interesting feature is that is exists at all. It's not so much the UI conventions, and stuff, because all of that had precedents in things. I would say it’s the individual and the multiple, the parallel editing, juxtaposition and superposition, and navigating between them. I would just really like to try to use this in a non-trivial domain. The collections of alternatives (Collections widget) reminds me of slide sorter in PowerPoint. That's to me is more a management convenience than really a creative part. The alternatives themselves and being able to cluster
them and group them is very interesting, especially when you implement other kinds of metrics and somehow harvest things from them to show me things about them. Pushing them (manually) around and grouping them is useful too. But again, this has more to do with what I am trying to use the alternatives for, what problem I am trying to solve. These three down here cost too much, those four up there, I didn’t like them... It’s very much like desktop, and I do group things on my desktop. But yeah, navigating between them, the multiples, and the superposition is most interesting and challenging, sort of, provocative part of the system to me.

1:25:44 Interviewer  What do you think are the least interesting features?

1:25:49 Participant  Get rid of Collections (the widget), put things in a bag and let me call the bag Backup. I didn’t say not useful, I just said least interesting. Absolutely useful. Yes of course you have to manage this stuff.

1:26:04 Interviewer  And if you envision this in a much more developed and polished version, how would you work with it in your practice?

1:26:35 Participant  I have an image in my mind, and it looks a little bit like clash detection (participant probably refers to collision detection). In a sense that, I would have occupied in 3D space, so it’s navigating this stuff, and occupying it. And then, because it’s not 2D, where I do this diff, I am actually looking at it in 3D view and you have to show me transparent volumes, and things. So, the getting rid of complexity, simplifying all this stuff that’s saying and making me resolve and showing me differences. This is where the clash detection would have to develop a lot. This would have to add a sense of that kind of thing. And again, the sensitivity analysis, when I change this, how does it change in all of the alternatives, and can you show me that. Again, especially in 3D space, because it gets so cluttered. Imagine yourself standing in a plant model, with 10,000 or 100,000 pipes and fittings and structural members all around you. And the alternatives are different sets of connectors used to hang the pipes. You know, being able check that, being able to see them behind the thing that they are hanging. But then, and yet, it’s not a stretch, what are the kinds of differences that we actually explore in that domain?

1:28:36 Interviewer  What do you think are the main barriers that the prototype represents to your work?

1:28:50 Participant  One is the proliferation of windows, and the reason I like superpose is that it reduces them. I can’t look at ten places at once and understand. Locus of attention is singular. So, you know, like a foveal display, that highlighted things you track my eyes in it, you only threw up images when I was looking at and did other things, and differently in my peripheral vision because of how I was looking. I am not saying you could use that but I am saying, that kind of a thing,
where you help presenting me much more information so... you might not be able to get away with
presenting it to me in a conventional way, as you are. Nor I can even consume it. I'll try, all these
windows, and I guess that's why I keep going back to the superposition because at least it's the
one that's both familiar to me and also collapses like this.

1:30:00 Participant The other barrier, beyond the multiple, fractured attention problem, is just
the familiarity problem, or users not being able to grasp the role of the different views. I think there
is just a little bit of the learning curve becoming kind of, knowing what to do with them all, and
how to create with it. Because, you are asking me to create my alternatives up front. And instead,
the way I encounter alternatives in the design process is by accident, leaving them behind me.
So I don't think “clone this alternative”, what I think was, alternative 3A is the one that I've just
showed to my client, and my boss afterwards said, that's not going to apply if you change this, I'll
change it on 3A. And it's only after the fact that I realize that the thing that I changed intersects with
something else; so now I have to go back to a previous version of it and go: what did I change?
Which is which? So again, I often encounter the alternatives singly (sic), and then the multiplicity
is something that I discover and I have to come to terms with later. And I don't do it in purpose,
actually the multiplicity is almost a problem.

1:31:25 Interviewer How would you change the prototype?

1:31:30 Participant Actually, what I would do, is two things: I wouldn’t change it in any way at
all, other than... No no, what I would do, I would just keep going with it, I want to see another kind
of representation, a graph, or an object browser, or something like that. So that I can move from
a CAD system, to an object-based information model system. So that's one kind of a sea-change in...
what the objects are. And the other one is, that I want to see this in 3D, because then you
have to encounter all of the visual problems of managing a system. That's like a Herculean... but
how would I change it. I wouldn't change anything about what you've done, actually I would just
add things to it so that you can solve other problems. You've probably gotten as far as you can get
with all of the different variations of this thing as it is. You would probably need to apply it to some
kind of domain, before you can actually answer, does X work?
Figure 13.7: Event categories for participant P4.
Participant P5

00:00 Interviewer  (Investigator introduces the participant to the topic of design alternatives.)

06:03 Participant  My only comment would be that this is probably one of the areas with most potential for development in design work, because it’s maybe one of those that are a bit more vulnerable to non-quantifiable aspects of design. For example, I often consider the design process as also the process of discovering value. And, I don’t know about the product design world, but in architecture the clients can only articulate so much about what they are looking for in projects. They can comment on the purpose of the building, and they might have a few values that they would like to see in the building, but that’s still a very vague description of everything the building will contain. And so, I often find, that a lot of the values that end up being embedded in the design, are not coming from the intents of the clients, they are coming from the architects. They are coming from the perception that the architects are getting from what is valuable for the community, what is valuable for the institution that the building is for, or the project is for, what is valuable not only at the time of construction but in the generation afterwards. So there is all these considerations that the client doesn’t have the time or the expertise to be articulating, so the architect is the partner that is helping to discover these values. Because, the architects, they do not really know much about these values at the outset of the project, it’s a discovery process that happens in the conversation with the client. It’s our responsibility to discover what our clients actually want, it’s not just something that we do out of our professional habits, or professional kind of the way we carry on our profession. It’s our responsibility to do that. We must be assisting the clients and helping them understand the values that can be stated as objectives in the project, and then try to protect those values so that they don’t get lost in value engineering, and so forth. And I think that these kind of expression of value, there is still no proper CAD solution that can include these expressions of value in the CAD design. So when people talk about design intent being expressed in CAD software, they only go so far as geometric statements. So you have geometric constraints, or geometric parallelism, or orthogonality, or formulaic expressions of parameters. In a geometry sense, there is a lot that has been advanced in CAD tools, but when it comes to embedding the values in a project, there is no solution for that.

09:17 Interviewer  Could you perhaps give some examples of what do you mean by ‘values’?

09:22 Participant  As for some examples of values in projects, in the Van Dusen project (Van-Dusen Botanical Garden in Vancouver, BC.), we didn’t want... Have you ever been there before, to the visitor center? In Vancouver, Oak Street. But the thing is, the roof, the seaming is all wood. And we didn’t want to lose that, we didn’t want to lose that, we didn’t want that to be value-engineered
into partly steel because of cost issues. We did want it to be covered with some mechanical services. We wanted it to express itself as a continuous free-flowing surface of wood. Elegance, and pure. And that was the value. The purity of expression of wood, the warmth of the texture of wood, and then having it free of visual clutter. So the freeness of visual clutter, by taking all that mechanical and putting them inside roof panels was important. So some of those values were maintained, because of some solutions, some technological or engineering solutions that would allow you to keep that impression. So to be able to take those values that are of very abstract nature, they are not mathematical per se, or they are not numerical, and express that inside of a CAD program. I haven’t seen someone attempting to do that. But what we were able to do is, we were able to take some parameters that enabled the value to be kept. So the value itself would not be expressed in the parametric program, but some process values of minimum requirements for to be able to create wood beams so that they are economical and prevent them from being value-engineered. Those specifications could be used in a parametric program. And the consequence would be that it was still wood, it wasn’t value-engineered into steel or concrete. So, there was at least an an indirect expression of values by expressing the constraints of the process to realize in this case, part of the structure of the roof, so that the value would not be lost. So in another words you would say, this is the value that I would like to keep, I don’t want this value engineered, and in order for to not lose this, here are some parameters that need to be respected in the geometry or in the fabrication of these panels so that this would still be wood. So that this would be free of mechanical services. So that’s as far as I’ve seen developed in terms or risk mitigation, protecting values.

12:19 Interviewer  OK. If I understood you well, by values you actually refer to subjective, qualitative qualities of your design that you cannot translate easily into numbers.

12:42 Participant  I think that any work toward design options can only help in the exercise of protecting values. Not only that, it could be that it could help to discover about these. Because if it’s good at visualizing, if you are visualizing design options, and it’s also giving you some information and some of the benefits, I think it’s important for any software that shows multiple options, is also capable of showing you comparable benefits or advantages between one option and another. So if you say, we have ten different options, of for example, structural connection. But of all these options, option D is the most light-weight, and that gives it preference in that category above all the others. But of all these, option H is the most economical in production. So just to show a number of design options, I think it’s helpful, but it could be much more helpful if you give the person who is reviewing options, the ability to compare them in terms of relative advantages, and multiple criteria.

13:59 Interviewer  (The investigator walks the participant through the prototype.)
23:00 Participant (Upon finishing being introduced to the prototypes.) This is not too far away from already being useful in projects. I hope that, as you add more features, it doesn’t... You know programs like 3DS Max, that are almost entirely comprised of plugins, so that if you remove the plugins from the plugins folder, the 3DS Max will still run but you’ll just be missing most of your modifiers and most of your creation objects, but it will still run. So I think that’s an interesting way to add features without running into the risk of making the projects too big to manage.

24:00 Participant But you know, in this kind of more essential form of the program, I can already see this... For example, when folks in architecture are doing planning, for example health care planning, sometimes they are working in 2D Autocad, with basic shapes, lines... Most of the time they are not even curves, they are just lines. But they are making very important decisions on the layout and distribution of spaces, and the kind of the safety levels, levels of certain spaces, and the walking paths from one space to another, and adjacencies, and things like that. But all of that is being considered in 2D environments.

24:45 Interviewer Are you referring to bubble diagrams perhaps?

24:46 Participant No, just the actual layouts of buildings, where the walls... They are not really walls but just spaces that are in closed polylines and they are kind of just modified and moving together, and then accessing the areas and so forth, the functionality of spaces. So I am just imagining them doing that kind of work, but where they are handling multiple options at the same time.

25:12 Interviewer You can take the mouse, and you can play with the tool, and perhaps comment what do you think, and also how this would apply to practice.

25:22 Participant Well... that’s just it. (Starts drawing with four active alternatives.) If I was designing the way they would, and draw a space here... Assuming there is already some tools for snaps and so forth, what not. You start with the basic shape, which is maybe the classical way in which you would solve a problem. By saying, well, you know a patient room, actually if I redraw a patient room ... I’ll say a patient room. If this is the corridor, and the patient room will have its demising walls on either side, and then you have to think about where the washroom goes. So if you put the washroom here, and in this corner, you actually get not that much glass, or window, but you get the visibility of the patient then. The patient bed is here, here, and here. But if you kind of freeze for a second, and say... Let’s mute this one. (Participant mutes the alternative, thus continuing to edit three alternatives in parallel.) And then let’s modify this one. So, here, I’ll say, what if we move the washroom down to here, and we get a lot more glazing, because now we have the full width of the patient room on the far side. Like so. But we don’t get much visibility of
the patient bed. So if the patient is in distress, we might not even see his or her head. Because
the washroom is blocking the way. So let’s just hold up for a second. (The participant mutes the
alternative he just edited, thus continuing with two active alternatives.) Now, what if we did the
compromise between the two, where the patient room is not even the rectangle, and we use this
kind of a shape. Like so patient bed is still there, but just at an angle.

27:24 Interviewer  So would you say that you see a value in playing with alternatives in this way?
Do you see that this could be potentially useful for these things?

27:33 Participant  Already, yeah. I mean, of course there are certain tools that it still needs
to have, such as geometric constraints, and dimensions, and things like that. But it’s already...
if I am looking at these, I’m already having these three options. And now I want to see them
superimposed, but I want to be able to dynamically switch between this mode, like this option, and
this option. So you said here I can see the options on top of each other? (Refers to superposed
parallel editor.) Sometimes I’d like to see what they look like one on top of another, and sometimes
then... I’d like to easily switch. Because I think some, let’s say, representations or options are
appropriate in specific moments. For example, the superposed, or onion-skin representation is
something that I think would get dizzy if you look at it for a few hours, but I think it’s really effective
if you can just turn it on for a moment, evaluate, maybe print, and then turn it off, and continue with
this... This here (points to juxtaposed editor) speaks a lot more to me where I’m actually... I find it
quite easy to use, I find working in this mode where I am controlling one and all I have to look up
or look to the left, to see what the other one looks like. So I might say, you know what, maybe let’s
just make a change here, I know that this would not be very good for my envelope but I just want
to see what it looks like. (Edits an alternative.) And now I look at them, and just compare. (Tries to
delete an item.) And it’s nice that it deletes in all three active alternatives. I find this parallel delete
operation quite useful because I wouldn’t want to delete this item in each active alternative.

29:47 Participant  But I think what is even more valuable is when I select it, it selects so that I
know... because it could be that this line, I changed this so much, that this line ended up being
a vertical line over here. Like it’s a completely... doesn’t relate to each other so by selecting it,
knowing what is going to be deleted in the other options, is important.

30:06 Interviewer  So, not just brushing, but selecting (items) is also important.

30:11 Participant  It’s important to see the selection. I think it’s really important that the selection
box is showing what’s being selected in each one. Because it could be that, you know, I might
be modifying this one solo, let me see. (Soloes and edits an alternative.) If I take this one, just
because it was nearby, I might move this away to use it for different graphical elements. And then,
as a user, I will probably draw another line here, and it won’t have anything to do with (similar lines in other alternatives). If I turn these back on, and then I want to delete this line. If this didn’t show me the selection, I might be assuming that this is the one that is going to be deleted. But, this scenario of users kind of using lines, in different ways and moving them in different locations, will always happen. And not only that, users will not notice whether they clicked here first and clicked here second. So the direction of the lines, the users have no clue. They will just draw with no care... And if you began to tell them, you need to be mindful of the sequence in which you are clicking, they will not comply. Because the reaction will be, why should I begin to care about all these characteristics, why can’t the computer just take care of that for me. Why should I pay attention to minutia details of using this software, like which end to click first and which one to click second? So, part of what I do, when I am writing computer tools, is to insure that there is a consistency in geometry when there needs to be, regardless of the way in which the users click, whether they choose left or right, or up or down. So building some robustness there so that it doesn’t really matter, they can just be as free as they ish. So if they draw this with a certain style, and then tomorrow they draw it in the opposite orientation saying one was clockwise and another one was counterclockwise, the result will still be the same. It will still handle correctly regardless of whether they do it one way or another. But this here is important because then I say, if I ever want the properties of this line to go to here, then as a user at a minimum I want to know whether that is going to happen or not. So if I click on it and say, if I change the property of this it’s not going to go to this line, it’s going to that one, then what I want to do, is I want to have the ability to make the difference and say: can you please switch these (two lines) around, can you please make this one the counterpart of that line. Because right now this is the counterpart of that line. I would like to have the option of saying, I want this line to be the counterpart of that line. So there would be a transfer. I think that would be one basic functionality, yeah.

33:44 Interviewer  So in a way, you would like to actually select both of them, and say...

33:50 Participant  And say, you now are associated with this one, not this one anymore. Which, I think, programmatically, is not that difficult to do because if they are both the same geometry type, if they are both lines, then you can just do a swap of ID or, if they are a part of a collection, you can just switch the item or the ID of the collection.

34:11 Interviewer  So what would you say about deleting this one, and deleting this one (continues editing, passes items). OK. Then you do this, select mute, and execute pass items, and it's being replicated. What do you think about this (pass items), is this a round-about way to do, or is it workable for you?
34:50 Participant  No, I think in many cases... In this case it works well because it happens to coincide with the other lines.

34:59 Interviewer  If you had something that already exists, you would like something extra, right? You just select them, and pass all these properties back and forth.

35:07 Participant  Yeah, I think both would be good. Like this, what you just did, I think is very valuable. But I think that sometimes another design will be shifted enough that this might not be most convenient, like it's more a matter of convenience. But I think that being able to do this is already a very good solution, it's better than a workaround actually. Because in many cases, it works perfectly.

35:36 Interviewer  OK. But just to make sure, you build something here, and then you have some geometry, and then at some point you'd actually like to establish the correspondence between these two parts, right?

35:46 Participant  I think that would be useful. What you just did here, is even more useful, but there will be situations in which establishing the correspondence would also be a good feature. And I think designers will understand really easily that the correspondence can be only be done between certain geometry types. You can only make a line correspond to another line, or arc to another arc. I think that is not too much to ask for designers to do, because it makes sense, it's like a common sense requirement.

36:19 Interviewer  And if you go perhaps even further, for example, you can select different doors in Revit or whatever, you can have French doors in one alternative and other doors elsewhere, and just establish connection somehow.

36:32 Participant  Exactly. I'm surprised that there are not very many programs that actually are able to allow the user to establish correspondence between instances.

36:48 Participant  But you have options in Revit, right?

36:54 Participant  You have design options... The way it works, it really just the filter. They way which design options works in Revit is that, if you have one hundred walls, and eighty of the walls are your design option A, and then you've added another ten walls that are for option B, and you add another ten walls for option C, but in option B you've also deleted five of the walls of option A. So it's really just turning on some walls, and turning off other walls, it's a visibility thing of turning on and off. That is nothing compared to this. This here, you actually have unique sets, you have four models. And you can change the models, but you are really just establishing correspondence
between elements in the model. This is light years ahead of what Revit handles in terms of design options. This is the way it should be in Revit actually. Revit pales in comparison to this in that sense.

**38:19 Interviewer**  What do you think are the most interesting features of this prototype?

**39:25 Participant**  For me, the most interesting feature is to see the different design options at the same time. That's the first one that comes to mind. To be able to visualize them at the same time. I also appreciate a lot the management tools of establishing option groups, I think that's a very good feature. I like the Solo and Mute controls, I think they are very elegant and simple, and they make sense. And, I like the ability to pass on elements from one option to another. All of those I think are great contributions.

**40:23 Interviewer**  What do you think are the least interesting features?

**40:49 Participant**  I think just being able to change the thickness of the lines. I didn't even bother to change thickness of the lines, because that’s one of the last things that comes to mind when I am exploring design options. Because, I see that more as more related to drafting, or documenting of your design, more than actual design. I am sure that it would come in handy in some situations, but that’s kind of the last things that comes to mind in terms of design.

**41:44 Interviewer**  If you envision this in a much more developed and polished version, let's say if you had a big company throwing its weight behind it, how would you use it in your practice?

**42:01 Participant**  First of all, I see it more in a conference table that’s touch-sensitive, where you have space for say eight different options, but multiple people, two or three people are standing around the table. And, I’m working with one option, some guy is doing something else with another option, and I am looking at it, and say, what did you just do there, I’ve just changed that, can you pass that over? So he just hits and passes it, oh ok, let me try to use that. I’m kind of like, more of a social way of interactively passing on ideas. In a parametric way, passing on suggestions.

**42:44 Participant**  But then, you could maybe record notes and say, well I just passed on this feature, why do you want that one? And you say, well, I like the lean aspect of it, or the ratio of it, or the elegance of this curve, can you audio-record it? And that’s kind of associated with that one feature. Collaboration, with some kind of audio clips, because that would be a much more convenient way than having to annotate notes. And then, at the end, you’ve basically finished your design charrettes, you know one-hour, or half-hour design experience, and then some of them would take all this development. Like, after the battle, you have all these different remnants of the design, and it’s actually maybe timeline-recorded as well. So, someone could say, why did we
end up moving that there in the first place, in this option? Do you remember? Let’s just play the recording. The recording, maybe just zoom or fast-forward to that feature, and say, what did the person say at that moment? Play the recording, and oh, that’s the reason why we moved it there. Oh, this is a good reason, a good idea.

43:58 Interviewer So you think that having design history would actually be useful here in this context?

44:02 Participant Oh absolutely, because I think design history is useful for any design tool. It’s just all too often that someone, you know, three months after the decision was made, look at each other, and say, do you remember why we did this? Do you remember why this was moved here? Do you remember why this was moved here? And if the person who came up with that idea is not there at the moment in the office, no one will have the answer to that question. There is a lot of design rationale that gets lost because you cannot conveniently record the rationale while you are making the changes.

44:33 Interviewer If I may ask, how do you actually deal right now with this problem in practice?

44:36 Participant I don’t think we are. I don’t think we are actually dealing with that because we don’t have the proper tools to deal with it. There are notes, like if you have a parametric tool, you can place a note. But the problem is, when you place a note, you have to stop designing for a moment, and document. My analogy is this: I like using the 3D mouse to orbit around in the 3D model, and the reason why I am doing that is that I don’t have to stop using my right hand for the mouse, if I can keep designing while I am orbiting around. It saves time in terms of orbiting, but what I appreciate most about it, is that I don’t have to interrupt my designing, my train of thought. I am uninterrupted while I am just doing a change in the perspective view. That I think is the most value for having a 3D mouse. So similarly for documentation of design, if you could just provide the technology or tool that allows you to document your design without interrupting the trail of thought, that would be the greatest contribution there.

45:53 Interviewer What do you think would be the greatest barriers when using such a tool for your work?

46:06 Participant The first thing that I can think about is interoperability with other software. The fact that you are in a separate environment is usually the first thing that comes to mind. The interoperability for a few years now has been a central theme in the design world. There are some really good design tools that are coming out but because they don’t interoperate with the main softwares, like Revits or Catias of this world... That alone, just the fact that it does not interoperate smoothly, is enough argument for many people to not even use these design tools.
46:47 Interviewer I do remember IFC was actually designed for this, but I kind of stopped tracking this. So, IFC is still not convenient of good enough for the purposes of interoperability?

46:56 Participant IFC is still not good enough, and not so much because of the standard, I think the standard probably has room for improvement, but the implementation of the standard in programs like Revit, there is still a lot of work to be done there. So, if the interoperability is taken care of, if someone says, well I am going to spend a lot of time working with this, but is it as easy as you hitting the button for to have that go into my Revit model, then if you say yes, it's as easy as that, you hit a button and what you just designed there, boom, goes into your Revit model. They will say, oh great, when can I start using it?

47:40 Interviewer What about if you were able to just connect such an application to Revit directly?

47:47 Participant Even better. If it's like an add-in, or some sort of added functionality in Revit, even better. The problem with embedding it inside of Revit is that Revit is a heavy program, it handles a lot of information. So it can't be nearly as responsive as this. Like this is lightweight environment, it's very nimble. The moment that you try to do this inside Revit, Revit will have all this other weight, the computational weight. The experience will probably not be as smooth and free-flowing as it is here. Like here, I really appreciate the fact that I click this, and it's easy to click. Whereas in Revit, when you are clicking on something, there are multiple objects on top of each other, and you have to cycle through them with the Tab key. And when you make a mistake, you misselect an item, and so forth. Once you finally selected the item, then moving it in Revit might not be as smooth as it is here, because here it's a line, whereas in Revit it's a family with data, their constraints, dimensions, etc. It's just becomes heavy. If you could have some sort of inter-model state maybe, where you have this as an add-in, but it temporarily turns off all the processes of Revit, allows you to work freely, then turn back on again, and now it updates the Revit model, something like that, would probably be more effective than having this to pay the price of all the processes that Revit is running.

49:29 Interviewer This gives me an idea of working with just 2D sections of a model. Another question: you work for a prominent company, with lots of resources, able to buy powerful computers. But still, even when utilizing powerful computers, you are still saying that it's pretty slow, right?

50:12 Participant There are situations where SketchUp models, Revit models, can become very power-hungry for large projects. There is just so many components in the project that you have to throw a lot of resources to it, and even if the computer makes it fast, it's just the fact, it's the
cognitive load too to the people. When you’re working with a model that has so many components in it, then there is this cognitive load as well. Visual clutter, having to thing about a lot of things at the same time. I think there must be a threshold that’s probably relative for one person to another. When you surpass the threshold, the thoughts are no longer as free-flowing as they could be. That’s why I am a great advocate for more schematic type formats for design work.

51:15 Interviewer  Could you see yourself using such a tool where you actually at any instant of time work with just 2D sections of your models?

51:26 Participant  Sometimes. I would want to have the freedom to be able to zoom into a detail, and working at section for a while, for as long as I need to, and then zoom out, and work at the whole building. I think architecture is often, you work at various levels of detail. Sometimes you are looking at the entire building, and sometimes you are looking at just the canopy. Within each day, there could be multiple levels of detail, in just one day.

51:59 Interviewer  That’s interesting. If I may ask this, how is Revit supporting this? Well, or badly?

52:05 Participant  I think Revit as a design tool, it still has a lot of room for improvement, but where it really begins to be useful, is frankly when the design is already mostly developed and you need to now document you design, then Revit has some powerful features for organizing data, scheduling, generating views, and creating 3D visuals. Revit is quite good at, once everything is already mostly spoken for in terms of design, then it becomes a powerful tool for managing and tweaking the design that you’ve already defined. The problem with Revit, usually most of the debates of relevance of Revit, is in the early days of design. Like in the schematic design process, or the conceptual design process, that’s where most of the debates are happening. People are saying, Revit is too slow, it’s lacking modeling tools, and so forth. And things like this, like new ideas of handling multiple options at the same time, that’s when most people complain about the lack of resources in Revit.

53:21 Interviewer  So, would you say that ... do you guys actually use bubble diagrams?

53:25 Participant  Yeah. But we are also generating bubble diagrams computationally too. So most of the time bubble diagrams are done manually, but we’ve already been for some time been generating bubble diagrams computationally as well. I’ve developed a custom tool, that’s part of the work that I’ve been doing... But we only touched the tip of the iceberg, it’s not like... We recognize that there are other ways to how to generate bubble diagrams, but we’ve only been looking at one or two different ways, but we know that there are a myriad of other ways in which you could generate the same bubble diagrams.
54:04 Interviewer  OK. A couple of minutes left. Looking at this too, how would you change it? Improve it, modify it for your needs?

54:21 Participant  The first thing I would do, I would, this whole half of this side of the screen (Arrangement view), I would have a button here to collapse it (So that I can maximize parallel editor.) The first thing I would do is, because, I only need to see this (i.e. Arrangement view, or Alternatives view) for a moment. This is just something, if I could have just the button that makes it show up. I would just click the button, show it when I need to, and then click... Because this is the mode, if this (juxtaposed parallel editor) can be maximized, I would want this maximized 95% of the time.

55:24 Interviewer  Do you have anything else to add?

55:32 Participant  Well, I think it has great potential. Obviously, as anyone would say, it would be great to see it in the 3rd dimension. But to be able to take it to the 3rd dimensions, it's not adding more to the core new ideas. The new ideas can all be expressed in 2D, you don't have to see them in 3D. The fact that you (can) implement all these ideas in 3D view makes it all that much better. But before seeing this developed in 3D, I would much prefer to see more new ideas, or develop further the new ideas that this has in the 2D state that it has right now. Because to spend time in making it, you know, give it the 3D aspect, I mean, everyone knows what 3D is. Everyone understands the value of working in 3D, there is nothing new about that, it's been around for so many years. But in the way it's presented here in the 2D environment, there is some new ideas here. There is new ideas that are very refreshing for the design world. So if I were you, I would capitalize on the innovative aspects of this software, instead of spending time implementing aspects that are already known, like 3D for example.
Figure 13.8: Event categories for participant P5.
Participant P6

26:52 Participant  (After the walk-through.) Can you pass a property between different elements? So for example, you select that line, and you want to pass that property to other lines in the canvas?

27:04 Interviewer  I see. So, you would find this useful? Selecting this one, and pass it to this one.

27:11 Participant  Yeah, from this one, and then pass it to all these, for example. Like, while you are designing, you might start doing one thing, and you start looking at this color, and you want to pass it to a few different elements.

27:23 Interviewer  Any ideas as of how would you like to perform this in such an interface?

27:30 Participant  Yeah. I mean, it is similar to the interface that you have here, if you right-click and "pass-to item", for example. So you know, if you do a "pass-to", the system would then ask you which item you want to pass to. And then you select the second line, for example. Like, if you, right-click, and pass item, you would say "pass item to" as a third option here (in the contextual, pop up menu). Once you do that, the software will prompt you to select the second item. And once you select it, it might give you the options of what kinds of things that you want to pass. It's sort of like, are you familiar with Autocad, like it's a very, very old feature called "match property". So basically, it matches properties.... Well it doesn't have all the different finer-grained elements. Basically, you could select a line that's in a certain layer, and of certain color. And you would match property, and start selecting different lines and it would match the attributes.

28:39 Interviewer  All right. So this is then a feature that is missing. Now, if you would like to open this project, we can perhaps move to the third part of the study, and you can just play with the tool a little bit. Perform simple tasks.

31:38 Participant  (Tries out editing in the prototype; add a couple of lines depicting a zig-zag shape.) Can you delete something?

31:41 Interviewer  Yes, but you have to switch to Select mode. Yeah, this one, and you can toggle the alternative with Mute and Solo. Right-click for deleting an item.

32:17 Participant  Can I select any of those control points for the line?

32:20 Interviewer  Yes. First you deselect it, then click on the button for Select Control Points.
32:27 Participant  Oh I see. OK. (Continues interacting with four alternatives in the juxtaposed parallel editor.) Interesting. The scroll is different from what I am used to.

32:50 Interviewer  Oh, the direction (of zooming in using mouse wheel action). Other participants told me the same thing. Because everybody is used to standard applications such as Google Maps, and similar.

33:07 Participant  Yeah. Google Maps, Revit works that way. So... I am not going to be asking about snaps, and all that, but...

33:17 Interviewer  You have snap functionality in that other prototype, if you wish.

33:19 Participant  No, that’s OK. I am just seeing what exactly you can do and what you can’t. (Continues editing four alternatives in the juxtaposed parallel editor.) OK, so basically I started with that (points at an alternative), and I switched over. If you select this one (control point), it basically selects the same point, and it starts moving it relative to the position that’s in there. That’s interesting.

34:20 Interviewer  You said it’s interesting. In the sense of being useful, or confusing?

34:34 Participant  No, it’s good. It’s logical when you think about it. I haven’t used it enough time to formulate an opinion whether it’s useful or not. (Continues editing four alternatives in the juxtaposed parallel editor.) OK, so basically I started with that (points at an alternative), and I switched over. If you select this one (control point), it basically selects the same point, and it starts moving it relative to the position that’s in there. That’s interesting.

34:34 Participant  No, it’s good. It’s logical when you think about it. I haven’t used it enough time to formulate an opinion whether it’s useful or not. (Continues editing four alternatives in the juxtaposed parallel editor.) OK, so basically I started with that (points at an alternative), and I switched over. If you select this one (control point), it basically selects the same point, and it starts moving it relative to the position that’s in there. That’s interesting.

37:28 Interviewer  Interesting. So, yeah... You would click here, on this control point, and somehow you would expect a line to appear between these two control points in this second alternative?
Participant: Yeah. I would click here. I understand that there are two points... I mean, if I click on the open canvas, like this, maybe I’d expect it to be the same everywhere. But if I click in an end point of one particular element, I would expect it to be... Because now I’m clicking here (at the end of line 1), and it’s actually showing this. So I create a line, and I want to create it from that end point (end of line 1), to this start point (start point of line 2), irrelative to where exactly they are (in other alternatives).

Interviewer: Interesting. So you would pick this control point, and for you that would be a signal that I actually want to start drawing everywhere a line from all equivalent control points in all activated alternatives.

Participant: Exactly. Yes. It’s something that basically works relatively in any location because the way design works, from one... I mean, when you look at the Starck’s design of that lemon squeezer, there were alterations that actually work with the same number of elements, but the shape changes. So if you able to draw, irrelative of the position of those shapes, you are able to enhance them in a way. So that’s one thing that I see here. You know, I am just trying to do a very simple select in a way, and see how that select could change, if it’s a landscape or whatever. And whether I’ll be able to trace that, but now there seems to be a gap. So to actually make it, then I need to mute this, mute this, etc. I have to sort of... (Participant states that he is forced do it for every alternative separately.)

Interviewer: Or you can just click on Solo.

Participant: Or, yeah, click on Solo. So if I click on Solo here, and I want to delete this one. And now I am in Solo here. But essentially, to replicate what I did here, now I am Solo I will be moving one of these points up here, in order to create that Select. I will do the same thing here (soloes another alternative). Manually moving them from one point to the other, if I want to redesign it. But in a way, this could be smartly done if you just track all the points (in all the alternatives).

Interviewer: In this older prototype, we have very simple snapping. (Investigator demonstrates snapping of control points in the older prototype.)

Participant: I think it is interesting to see this superposition too, that’s an interesting way to view things. (Participant comments on the superposed parallel editor.)

Interviewer: Yes, perhaps you can use this one instead. But here, you actually have basic snapping. (Investigator works with four alternatives in the older prototype, using snapping of control points. Creates four landscapes, or mountain profiles.)
41:12 Participant I see. That's what I am looking for. What are these buttons for? Oh, line thickness. How do you snap this? (Tries to snap two control points into one another, and drags a control point in three different alternatives.) Oh, this is interesting. It's good. (Continues editing in parallel. The prototype crashes here, causing a delay. The participant recreates the model.)

45:43 Interviewer So, what do you think about this? If you had done this before, like you snapped two control points together, what would you expect to have here, because I saw you activating this one?

45:57 Participant Yeah, exactly. No, I was trying to see how the program will do. What would I expect to have there?

46:15 Participant How do you pass property...? Oh that's only shape. I can't pass the movement. (Passes shape from one to two alternatives.) OK. I see. So this one passes property shape, so basically making them the same again. (Soloes one alternative, and moves one control point up. Soloes another alternative and tries to move the same point up, however the snap was lost.)

46:47 Interviewer Yeah, the snap was lost. It's still buggy. But you would expect the snap to be preserved here, right?

46:55 Participant Yes. So I am relinking snap here, using only this guy. So there is three versions of that design (at this point the participant has three similar design alternatives on screen). So what happens when I don’t want to use this one anymore? (Points to the fourth alternative which lost any resemblance to the other three.) Can I take it off the collection? I basically just do that? (Reselects just the three similar alternatives in the Arrangement view.)

47:13 Interviewer Yeah. But what would you like to do here? Would you like to have an option to remove it from the grid?

47:19 Participant I mean, like a lot of time you are basically going to work with... Say for example that you are starting on one thing, and then you work on another, and then work on another, and then you discard old things, because you don’t want it anymore. And, you would work with the new one. So there is always things that you are basically discarding as you go.

47:35 Interviewer Would you like to some, like, X button or something, just to remove it?

47:39 Participant Yeah, exactly. Instead of reselecting all of them (in the Arrangement view), you could just remove it (in the parallel editor). Oh, control + select works too. It's good.
CHAPTER 12. CONCLUSIONS

48:09 Participant  This is a very interesting view. (Participant switches his attention to the superposed parallel editor.)

48:12 Interviewer  Yeah, that was what I was about to ask, because you have been predominantly working here (in the juxtaposed parallel editor).

48:16 Participant  Can you actually work from here too? Oh, I see. I wasn’t working here because I didn’t know that. But, yeah, the superposition parallel editor is probably the most useful, definitely. I would think about it, if you are designing, and you are evaluating options, you need to see them over the top of each other to actually understand the subtle differences. Perhaps if you are zooming into one option to actually do a particular piece of design maybe you are going to work in this one (i.e. juxtaposition parallel editor) because of the clarity of just using it. But, in a way, once you are coming to the moment of comparing results or sort of identifying which ones you like more, it’s good to see them in reference to all others, and this (juxtaposition editor) doesn’t really allow that. This (superposition editor) allows the comparative type of looking at it.

49:13 Interviewer  What would you say, what’s the advantage of having this one (juxtaposition parallel editor)?

49:19 Participant  The advantage of juxtaposition editor would be in a more complex design situation. Often times you will be confused with all the different overlays. It’s pretty simple here because of having just three lines here (in juxtaposition editor), but you can see seven lines here (in the superposition editor). But, when you multiply it, these three lines are actually three hundred, and all three hundred are actually moving, it might end up with a few thousands. It can be very confusing at times, but in a way, if you know exactly what you want to look at, or what you want to compare, it’s good to see them right beside each other. Especially if you are looking at, you know, curvatures or selects, or particular forms. If you are looking at the design of this (points at the computer mouse), you know... Mouse. Looking at different ways that you can actually replicate your hand curvature, and you start playing around with the curvature. It would be good to compare all the 3D elements of the same place (refers to the upper part of the mouse), because if you look at it on its own you might not see the subtle difference. I think about this particular design (mouse), I think about design that is two millimeters or three millimeters higher. If you look at them in the juxtaposition editor, it’s really hard to see those differences. So I think this (superposition editor) is very very useful to have this type of superposition.

50:53 Interviewer  You said differences. Would you find a view showing just the differences between designs useful? For example, two mice, one is to two millimeters higher, and you would have which actually shows just this part that is a little bit higher. For example, highlight it in red, or something.
51:33 **Participant**  It could be useful, not sure if it’s that much useful over the top of just what you have here, because in a way I would be interested to look at a wholistic design. It might be useful for a particular detail just to turn it off and on, to actually highlight the problem or highlight a particular difference. But, while you are designing, you might want to look at the hollistic thing and see its proportions. It might be good if you want to, for example, just to see the distance between this and that, it might be good just to highlight it. Your eyes are drawn to it right away, and not too many lines in them all. But when you are comparing designs, I think it’s good to have the overall look.

52:21 **Interviewer**  You would not want to lose context.

52:23 **Participant**  Yes. Exactly. Just the having the big picture is pretty important.

52:30 **Participant**  One thing I want to try, is... I have all these alternative. (Works with alternatives.) Can I replicate one of these guys? (Tries to replicate an alternative in the Alternatives view.) Can I replicate one? (Clicks on the duplicate button, and then edits the originating and duplicated alternative.)

53:26 **Participant**  It’s actually interesting, because I think of it as history as well, like it’s a way of seeing history. As you basically duplicate a copy and see, this is something that I am interested in, I like it, but I want to keep that. Or else, I want to create just a little bit, change it, and I don’t want to lose that. So instead of having to go forward, like redo and undo kind of thing, you could create a new option, and see it right beside it. You know, what’s the particular implication of changing that little piece. You can judge it over the top of that because you could, if it would be possible to delete those two guys (points to other two alternatives in the selected set of four alternatives).

54:19 **Participant**  It would be interesting to actually be able to remove these two from here. Like instead of going back to these views (Arrangement view), like for example, I want to reserve those two (originating and duplicated alternative), and only compare these two, and sort of discard those (two alternatives at the bottom of the parallel editor). Right now, I’m clicking here over the options (points at the Alternatives window) but if you have too many options you might not be able to visually realize it in a way. So, it would be interesting if you could just minimize things (in the parallel editor). So you can see them, and it would be interesting to have the comparison between just these two.

54:57 **Interviewer**  You are saying that selecting and navigating in the Alternatives view could be done better, right? It’s not optimal right now.

55:04 **Participant**  Totally. I agree.
55:06 Participant  Can you move... (Tries to move alternatives in the iconlist view.) It would be good if you could group the alternatives. (Switches from Iconlist view to Arrangement view in the Alternatives widget.) So, that's also a possibility. That you can basically start arranging the ones that are similar. (Arranges alternatives in the Arrangement view.)

55:36 Participant  OK. I think that the canvas, or the arrangement part here... I think the thumbnails (i.e. icon list view) might not be the first go-to for me. I think arrangement works better because I have control over how they are put together. It will be good, maybe, if we have, but that's another set of hierarchy... Like you have a category in the thumbnails (the icon list view)... So we now have like a collection of alternatives, and then you have alternatives, but maybe within those alternatives I have the preferred ones, for example. I have a group, and I can group the other ones as "discarded", and I can group the other ones as, you know, different type.

56:18 Interviewer  So you are saying that this could be a little bit confusing for the user, because you would have groupings here (in the Arrangement view), and those actually show the collection. And I have them here as well (in the Collections widget). Are you referring to that?

56:29 Participant  No, I understand the Collections of Alternatives (view), that's basically regarded as a top level. Alternatives (view) is what you are actually working on right now of that particular design piece. What I am saying is that you could actually use some grouping to those design pieces so you have like, you mentioned Starck working on 30-35 different sketches. At the end of the day, you might, after lunch, say: I like those three and tomorrow morning I would want to continue on. But instead of tomorrow morning coming in, I actually started looking between the thirty and starting to find them, navigate through them, it would be interesting to already have them grouped as my preferred options, or the ones I don't like anymore.

57:13 Interviewer  So you would like to have this... Could you pinpoint the exact location? How would you actually like to work with it?

57:23 Participant  Maybe in the arrangement view I could do that, and I would highlight them in a box or something and say, it's a group. Like a container kind of thing. Maybe if it's in the thumbnails (icon list view) I could actually have different text tags, like if I do add a group, tags "group 1", or "group 2", or "discarded" (tags on top of the icons). You have a list of something. A category kind of thing.

57:57 Interviewer  So would you as far as having a full folder structure, like Windows Explorer, or Mac Finder. Or, would this be an overkill.
**58:10 Participant**  This might be an overkill. Just having this functionality at one level, just one level would be enough. It's different from how everyone works. And like how many levels of details you want to want to think about it. And for very complex projects, you might want to start to think about, OK, there are actually more, like, more hierarchy to the groups... But I think it's some sort of functionality for grouping, or a way to put together things that have certain things in common. Design alternatives that have certain things in common. If you go back to that Starck drawing, you might start thinking, OK, there is one concept of those alternatives that can group four or five of these, that have those particular leg-like thing (qualities). And others that are sort of working on one leg. So these are maybe the five-leg options, and you have like four of them. And these are the one-thing, and you have three of those. And, you know, you might want to group your designs. If you are opening up the canvas to play around with multiple alternatives at the same time.

**59:37 Interviewer**  Opening up the canvas. Hmm... An interesting observation. Like opening in the sense of being able to create shallow hierarchies or groupings?

**59:51 Participant**  Yeah, it's a way of organizing your artist's canvas. Again, I am referring back to that sketch thing from Starck. (The participant refers to the example of alternatives, drawn by designer Philippe Starck, shown in the first part of the study.) You are saying that, this is the canvas, here is a grouping. Or, like, the hierarchy went in a way like, he was working counter-clockwise, or something like that. And I am kind of seeing this (points at an arrangement of several alternatives) is part of the arrangement that you might want to arrange in this way but, at the same time, you might further on decide that this is a design option here, and one here that shares some attributes that are like each other, you might want to group the together. Or, use one that sort of is merging this and that. So you might want to have just one level of control of how you put them together, so that your mind is understanding the differences between them.

**1:00:50 Interviewer**  Would you prefer to create another collection here (in the Collections widget), and move other alternatives into this one? So you could have different collections here, like 'legs' designs, or whatever.

**1:01:15 Participant**  Yeah, if it's possible to actually move design alternatives between the collections (that would be useful). But, for some reason, when I first saw the program, it felt to me that this (points to the Collection widget) is more like a different project. That's why I understand it, you could be working in a different way, but I thought of it as, this is actually totally different projects (points to a thumbnail in Collections), and this is where your working space is (points to Arrangement view), but could be regarded as you're opening up this for the same design problem, you are just grouping it as a collection of them. Yeah. That actually works with that one-level hierarchy, if you think about it that way.
1:01:58 Interviewer   Yeah, actually, this is very valuable, because we already have some comments in this direction.

1:02:10 Participant   Although, like when you are saying that, it actually makes sense that you might be able to, but I am not sure if there is a possibility to drag and drop.

1:02:20 Interviewer   Would you like to have that?

1:02:21 Participant   Yes. If you are using this as a collection of designs, or collection of alternatives (points to Collections widget)... Yeah, being able to take these guys (alternatives), drag it, it would sort of copy or move to here (points to another collection in Collections). You might be able to open up that (collection) and Control-X and Control-V for example.

1:02:44 Interviewer   But does it feel more natural for you to have this capability of grouping collections and so forth to have here (in the Arrangement view)? Would you say that it feels more natural (compared to the Collections widget)?

1:02:55 Participant   Yes, it does. And that's probably why it came to my mind at the beginning... Yeah, if you can group them, or at least some sort of highlight those areas together? So even if you are able to draw a rectangle and, you know, has a blue background and be able to type a text (on top of that rectangle), or right-click on the rectangle and name that rectangle. It would at least, you can visualize or understand what exactly you are trying to do with these options. Maybe for these other ones, they have a different name.

1:03:31 Interviewer   So, a shallow one-level hierarchy would help.

1:03:34 Participant   Yes.

1:03:35 Interviewer   So here, in this newer model (opens the newer prototype), here I can actually switch to subwindow mode, and let’s say I switch to...

1:03:48 Participant   So that would show different collections of alternatives?

1:03:52 Interviewer   Yeah. So in this sense, I have different collections. Here, I can just drag them here (into another window/collection). But this is window-based. But here (in the Arrangement view), then, we should have some capability to drag alternatives among containers, right?

1:04:20 Participant   Yeah, exactly. If you have a container (in the Arrangement view) you should be able to move this here (moves an alternative), and then it becomes a subset of that container (a clustering of alternatives), right?
1:04:29 Interviewer So, but in this case, it's actually pretty trivial, because we already have groupings, right? (The investigator refers to spatial grouping i.e. grouping by spatial proximity.) So we can consider this to be containers, in a way? But you would prefer, let's say, a rectangle around it or something like that, for example? Some indication that this is a grouping?

1:04:44 Participant Yeah. It's the visual part of me that wants this, in a way? Yeah, like similar to Grasshopper, for example, where you can actually group things and write comments and things like that. Maybe just an ability to just right-click text that could help you understand exactly what is that. Because, I'm just thinking as you open up the canvas to, like, thirty of forty or fifty different things, and you work on this thing for like two days, it might get quite confusing. And again, it really depends on how organized the designer is, or the particular person who is doing it, because some people will be very consistent and keep deleting things they don't like anymore, some people will like to keep history of a lot of things because they might recur to it. I think just the one level of grouping and being able to add some text into (alternative), I can describe what this option is, just from the visual standpoint here and say, ok, this is the one that has high peak, for example, or something like that. This might be helpful.

1:06:00 Interviewer What do you think are the most interesting features of these prototypes?

1:06:30 Participant The most interesting of all is actually the superposition view (parallel editor). Yes. Again, I think it's new in a way, I don't think I've seen if before in a 3D program, that you are able to visualize different things on top of each other. Especially when you think about it in terms of design alternatives. In AutoCAD, for example, yes you can have different layers for different things, and draw them, but you never... You don't use it as a design alternative, it would be like totally different objects that you are creating in those layers. Right now, it's basically, I think of it, this could be a potential for the same design, the same object, it's just... how I am moving it, or sculpturing it in time.

1:07:29 Interviewer What do you think about layers in AutoCAD?

1:07:37 par Layers in AutoCAD are very useful. But often it's not about design alternatives, it's more about the different subsystems, things like that. You wouldn't usually use it as an alternative because you can't visualize it at the same time. I think this (points to the superposition parallel editor) is very interesting, especially how certain ones are sort of half-toned here. (Refers to the fact that at any time, just one alternative is rendered normally, while the rest are grayed out.) I am not sure why now they are half-toned, but how I think of it, it should be that if I am working in one alternative for example (in the juxtaposed parallel editor), and all the others are turned off, here, I'd be able to see this A-5. (Participant would like the two editors to be synchronized in terms of active and inactive alternatives.) Exactly. So that's how I would expect it to work.
One idea that we have, is actually be able to select multiple of these here (in superposition editor), and also being able to edit them in parallel as well.

Yes, you should be able to do that. Be able to edit multiple design alternatives in parallel in the superposition parallel editor. This is what I think is most attractive about this.

Also, the possibility to use the different... OK, I tend to think of it as times in history, when you are thinking, you know. You always think about one thing, and say, OK, what happens if I just move that part a little bit, would it be better in proportion or not? It's something that you don’t even know the answer of, until you do it. So problems of current softwares is, like, you do it, then to go back, you are undoing things. It's not really leaving that as a viable option anymore, like you either do this, or do that.

So you have to switch rapidly between undo and redo, undo and redo, undo and redo, to see the difference, right?

Yeah. Sometimes it’s not just the one thing (i.e. just one model). So, that's the problem. It might be OK if it's a one move, but sometimes it's actually a series of twenty-thirty different moves here. You are actually modeling a piece, and you are creating it, and it's not as simple as one comment. So sometimes it’s hard to visualize difference between two different things and you have to save as different files for example. Or, create rendering now, and create rendering later, and save that as an archive of all these things. So, it's interesting to be able to see them and write side by side, and also impose on each other. That's the most interesting.

This feels like something that you would really like to have in practice, right?

Yeah, because... Again, thinking about practice, designing one thing, I am always in the seek of whether this action that I am going to do, is going to be better or not. I don’t know while I am doing it, until I do it. And that's why I’m modeling it, in a way, to actually visualize it. Sometimes I have a sense of, yes, that's probably going to be a better way, but sometimes it’s not going to be. So be able to judge and compare between the different things that I just did, versus the one that I already have, and not losing that information, like creating a timeline in a way and being able to see instances of that timeline, I think that's very important.

Interesting. So you mentioned design history a couple of times. Would you like to have some sort of design history here (in the Arrangement view)? Let me just show you quickly a mockup. (Investigator opens a GUI mockup graphic, showing a graph of how designs are derived from each other.) So I have this design, I do some edit on it, and then I end up with
a new design. Do you think this would be useful, such a tree of derivation of designs? Or do you think this (Arrangement view) would be more useful, perhaps having arrows here. Because if you get this one, you have A-5, then you clone it, you would have an arrow pointing at the new design. Which means, you arrived at this one, from this one. Or, you don’t find this useful?

1:12:39 Participant  This (the graph of design derivation) is a little bit confusing to me. The reason is, sometimes I don’t really want to visualize and capture everything that I move. So it would be hard, or maybe it’s another functionality that it could be… I might want to select certain things within the history. Visually, the history, or like how you would see history in programs here, is a history of commands. So you actually got a translation, you got a scale, rotation, added a line, ... Sometimes those commands are very fine-grained, and it doesn’t really matter. I don’t want to see twenty things that, I moved this one, and then.... I actually want to capture certain points in history. So this point, versus this point, versus that point. And maybe there are like 300 different commands in-between. But I want to go back in time to one thing, it’s a different design state, so it’s not really thinking about commands, and a history of commands, it’s a history of particular snapshots of the same design. Like, I might not be interested in a lot of the incremental changes, I might be interested in how I arrived at option A, I did a lot of things to arrive at option B, but I want to compare A to B, I don’t want to compare A.2 to B, for example. So something like this might create too much information that is not really useful. Maybe it would be better if you are actually able to select that particular instance which why I’m referring to...

1:14:22 Participant  Back to (switches back to the prototype). So while I think of this, right now, I’m basically, I’m off with all this (works in the juxtaposed parallel editor), they are saved as the state where I want them to be saved, and I am taking this guy and moving it up and down (drags a line around in one edited alternative), and changing so many different things, and all that. I don’t really care about the three different commands that I did right now, I care about this particular result that I think looks good. I want to compare it to that one. So this here (points at the juxtaposition editor in general, i.e. at all four edited alternatives) is for me the history of the design, where this (points to an alternative) was one thing that I thought was good, this is another alternative of it that’s good, it’s not the particular one extra step, it might be like a few hundred steps. That’s why I am interested to see this more as... It’s a snapshot of a particular evolvement of the design, a different snapshot.

1:15:27 Interviewer  What do you think are the least interesting features?

1:15:59 Participant  The ones on the right, like the parallel editing, juxtaposition and superposition, these are very very helpful. The arrangement and be able to see everything in a canvas is helpful too. Let me go to other views in the Alternatives widget. (Participant takes a look at the
Arrangement is more interesting than the datagrid but I can also see myself at some point of time maybe wanting to see it (in the datagrid view of Alternatives widget). It's probably going to be one percent of my time spent on it, but it might be useful. This (Arrangement view) would be something that I would be having all the time, because I want to see how things are progressing, it's good to have these (thumbnails) updated as well. Yeah, exactly, it's live (updating thumbnails in Arrangement view as alternatives are being edited in parallel editor). So OK, I think this is interesting.

1:17:07 Participant Data grid is less interesting to me than the arrangement but I also see it's useful. As you said, maybe it can be grouped, up or in a way, it's more like high-level things. I don’t want to know the exact second. And maybe like things that are one week ago could all be grouped together. It could have some level of better design. It's useful. Thumbnails (the icon list view)... I don’t really see the benefit of thumbnails versus arrangement. Because arrangement already has thumbnails. So for me it’s a duplication of what we already have, but in fact Arrangement view gives me a better control over how I want to group those thumbnails together, or how I want to visualize them. Again, if I have more control over the arrangement, and how I group them together, how I can add text comments, that would be the only view that I want to see.

1:18:28 Interviewer Comments? You mentioned comments? You would like to have comments as well?

1:18:32 Participant Yeah. So when I mention text, if I can have, you know, like I can place text, like here for example, if I can just write, "I like these two". Simple annotation, very very simple text box where I can type in. I think that would be helpful: to group, name the grouping, and comment on it, or add the simple annotation. If I have this functionality, yeah, I have the thumbnails, I think thumbnails (icon list view) might not be needed at all.

1:19:20 Participant Collections of alternatives, I mean, it's... Depending on how exactly you see the thumbnails (iconlist view). My initial reaction was, I see it as a project (view), which, again, if you have projects then you might not want to see them right now, like it might be in a different windows folder or something. But if you think of it as a combination of alternatives, then you might want to see it. But actually, I think it would be better to have all these (points at both the Collections and Alternatives widget) as one, and show me the hierarchy in that canvas. So you know, you have the whole canvas, and you are free to group them as you want. You know, if you want to group these together, and you have like four or five, and maybe you think of this as a collection of alternatives, right? And you think of another one here, but it's more visual that you can actually play with. I think this (Collection widget) is a duplication of (Arrangement view). Well, it's hierarchy, I understand that, but hierarchy could be visualized in one panel, or one canvas.
Even if you are in the alternatives and looking at a way of having, say, this is a tab and it says "The five-legged options", and you have four different alternatives for that, and this is the "obsolete" (collection), I am not using those anymore. If you have those. And this is a totally different project C, and I have four of those. This could be a way of visualizing things that show the hierarchy but at the same time is in one panel. You might be able to have an opportunity, instead of having those very hierarchical, very ordered (collections), you might be able that same way of looking at it, but it’s, you know. I want this rectangle, I want to have big text here, and I want to see the three different alternatives. This one is this type of rectangle, the text here. And they have the same name. You can think of it as itemized, canvas, I prefer this type of thing. You are just free with the canvas, to put things around, and put them where they are. Because this type of thing (Collections widget), you can always go back to Windows Explorer, and see the different files that are saved, in that particular view. To me, this (Collections widget) and thumbnails (icon list view) would be least interesting. Arrangement view I think needs some modification so that I can actually take the functionality of this (Collections view) into Arrangement (view). And this also simplifies the whole view if you do that. A few little things that I need to worry about.

1:22:10 Interviewer I’m actually glad to here that, because this is a pretty consistent observa-
tion. People really would like to have everything in one panel.

1:22:20 Interviewer Envision this in a much more polished and developed version. How would you use it in practice?

1:23:03 Participant So if I think about working in Revit, right now you have a project browser, which basically shows a different views of that browser. Maybe in the very beginning there is this panel that basically shows design alternatives, and it’s one of the items on the project browser. You know, it’s a particular view, similar to a plan view. In Revit, there is basically a project browser that shows, there is plan 1, plan 2, section 4, 3D 2, for example. I click on a one of those, bring up a window here, it would show me that plan. Maybe here there is a canvas thing, or panel, or alternatives window, or alternatives canvas, for example. It’s a view. I click on it, and bring up this view that would show me all the different options similar to one of those, right? And then I would have a view, like hand in hand, I would have a plan. If I am working with a plan, for example. I could see it as, you know... You have a project browser, it would say, plan, section, 3D, whatever. And then it would have alternatives thing, alternatives canvas I would call it. For that alternatives canvas I’d say, put it in one of the screens, and I’d be able to see the map of all the things that I have, A1, 2, 3, and grouping them together for example. On the other screen, because I like more real estate, I would have for example here a plan view, or a section, or a 3D, and they will have a 3D up here, and have a section here. And I will be expecting to, once I click on three things, I could actually see those happening in plan, 3D and section simultaneously. It probably needs
more real estate than we have right now, because if you think about it, each plan would be, if I am selecting four options, it would actually be divided into four. This 3D view would be divided into three sections, that one is the same as this one, that one is the same as this one.

1:26:17 Interviewer  OK. So if you have a house design in Revit, you would have three different versions of the design of the house.

1:26:25 Participant  So you would have three different versions of a house design, perhaps it's even the design of the wall or something. And you are able to see them in plan, 3D, and section, like different view that you want. Because what you are working with, I always have different views working on the same thing. Because I might do an edit in elevation but I actually want to see the result in 3D. So I would have those hand-in-hand in parallel. So looking at it from different directions or different projections. But, at the same time, because you are working on different views... Each of those windows that's showing a 3D or section, it's actually split into different views. It might be harder and harder to have like ten different alternatives, and look at them at the same time, it might get confusing, just because of the amount of subdivisions. So maybe four would be a maximum, and it's would still be viable that you only see four at the same time, and if you need to see more, then you actually want to minimize one of those and see the other. It might be that you don't really want to see more than four in that particular view, when it's sufficient to see the ten alternatives in the thumbnails. Usually, like if you just with your mind trying to visualize things, and see how they are different in design, you are probably not judging more than three or four, because your mind can't really go more than that.

1:28:01 Interviewer  If I heard you correctly, you have another dimension here of your model, "part". Because you mentioned the wall, this implies that you actually move among subsystems of your model.

1:28:18 Participant  Yes. Again, these can be groups. That's the group that I am changing, the north wall. In this particular set of options, the design all the same, except that wall that is changing. And I might be selecting these together, and seeing them here in three different ways. It might be the different grouping of categories of options that I am talking about. Like, when I was talking about the design of the 5-leg option versus 1-leg option, it's a similar way to think about it, ok, I'm looking at the design of the north wall right now, and here I am looking at the overall shape of the roof, for example.

1:29:09 Interviewer  So different windows for different parts.

1:29:12 Participant  Maybe. But I would leave this open for the designer to group them, because we really change a lot between how we group things, how we think about design, like... House
for example is made of so many components. Usually, you would not design a house and another house, and try to compare them side by side. It's most of the time, comparing elements of the design. Like, you are comparing either the overall form, and you are bending it or shaping it in a way, or you are comparing different heights of the same thing. What happens with this roof, instead of that, it's actually it more shallow, or too shallow, and something like that. And everything else might be the same for this particular study that you are doing. You just want to see the effect of this particular roof. Usually, I wouldn't go too far into, I design one thing, and a totally different thing, and a third different thing. Because, to me, it seems like three different projects. There isn't really that benefit of overlaying things on top of each other because they are not relevant any more. Like, they are just too far outside of each other, they look too different, so you don't want to superimpose them. You might be able to just, at the end of the day, just print the three of and see which design is better. Making those, like, smaller-grain design decisions on things that you are changing in your particular design, it's probably elements of design.

1:30:46 Interviewer  What do you think are the main barriers to effective work using the tool?

1:30:53 Participant  Complexity. Oh yeah. Complexity related to multiple designs, and multiple things that are changing in that one particular design. One thing is too see the proof of concept on a set of three lines, and actually coming and working through it, and say, OK, you are snapping through this point but it might lose this snap point on another one because you change things so much that it lost the connection or something like that. And if you are trying to connect with three hundred different objects and each of them is associated with another thing, and... I don't know exactly if it's going to work correctly or not. It might be too complex to actually achieve. So it's complexity in both ways: the number of items that you are trying to control, and visualize at the same time. And also, the multiple alternatives that you are visualizing.

1:32:04 Interviewer  So you are saying that it's complex in the sense that you could potentially not know “what is happening” in your alternatives

1:32:12 Participant  Yeah, so even the model would not be able to handle that complexity because it's calculating 300 different items at the same time. For me, in GC for example, sometimes it's actually hard with the computer power that I have to, because of all the different components, it's thousands of things that they are calculating and visualizing at the same time, to actually move an element here because, this line actually triggers 3000 different components that are calculating the sun's angle, and they are changing different orientations of that () device. So it's really hard on the computer power to do it on one alternative.

1:32:51 Interviewer  Alright, so you are worried about computational costs of dealing with multiple alternatives as well, because you think it would be slow in real life.
1:32:56 Participant Yeah, definitely. Because I already know that it could be slow with only one alternative. While the idea is pretty good to actually be able to see two and three and four, and be able to, let’s move this one here, you are actually seeing it move in the thumbnail. Maybe there is an option to only see the thumbnails changing when you select it. Doesn’t have to update all the windows all the time. When you have four different options, and when you think about the thousands little pieces that you are updating on the computer, how to calculate and update them, in one alternative, then you will find this by four, so it’s quarter of the power. This is something that worries me and is challenging about this system. Especially the graphics, like a lot of times there are ways in different 3D programs where you would turn off the graphical update, so you do one thing, work through it, and then turn it on to see how it works graphically. Like switching to wireframe graphics and similar. All these different things and tricks that you would do to sort of make your model move faster, because now you are opening up the canvas to, like, multiply, multiply, all these different things, it might just crash on you.

1:34:22 Interviewer Yeah. I mentioned this to another participant. You probably have powerful machines (at your work), but he said, yes we have, but still it could be really slow.

1:34:32 Participant It’s slow. Like, no matter how powerful the machine is, you might get into doing some computational drill that you are basically just updating thousands and thousands of lines, it takes time to update. You can always sort of maximize your use, like push it to the edge, in a way? It’s easy to do that, especially if you don’t know what you are doing. If you don’t know the implications of how the computer in the background works, with the certain elements that you are trying to update, you might just run into things that would really really slow down your process.

1:35:44 Interviewer How would you change the tool?

1:35:58 Participant Yeah. We sort of discussed that as well (already). How I would change it, it would essentially be combining together the Collection of alternatives and Alternatives into one now, maybe not having the thumbnails (icon list view) at all, and enhancing Arrangement view into allowing more control for designers to arrange things together, group them together, add annotations.

1:36:23 Participant Maybe even scale different thumbnails, because if you like one of the option you could change the size of that, so that is has more visual power than the others. I could be able to select just this one (alternative), and resize that because I like that option. This is the option, and there are a few different alternatives of it that I could arrange below it, and smaller thumbnails. It’s actually more control over how you can arrange things, would be better, and having it one canvas, like you don’t really need the multiples of collections of alternatives. That’s in terms of UI.
1:37:09 Participant In terms of the actual working with the model, one thing that I thought would be an improvement was the if you are able to, associate certain elements relative to other elements. So if I create a new line for example, I could be able to associate it with a certain element that was there before. Say for example, end point of a previous line to the start point of another line. And that would be the way it updates in all the graphics, and not necessarily the exact location of X, Y, and Z. There are probably other things as well to work on, with the functionality of it, but to me it seems like it’s an idea, more than a working scenario, because it’s limited in functionality so I can’t really test all the things that I would do in a design.

1:38:33 Interviewer OK. Anything else to add?

1:38:45 Participant This is pretty good, I like the concept and ideas behind it. If this was implemented in a software that I use, or an add-on, something that is more robust that I can actually use for design, it’s definitely something that I would like to work with.
Figure 13.9: Event categories for participant P6.
Appendix B: A Formalism for Subjunctive pCAD Models

This section contains an abridged version of the material that I wrote for the “depth” part [192] of my comprehensive examination on August 31, 2012, and in which I described a formalism for pCAD models supporting multiple design alternatives.

Introduction to Subjunctive Parametric Models

Parametric models are characterized by nodes (i.e. objects describing a part of the model), their parameters or ports (i.e. node properties—primitive variables defining a design), and data propagation between dependent parameters/ports. They can be represented as directed graphs with arcs denoting data propagation from one node to another (Fig. 14.10).

![Figure 14.10](image)

Figure 14.10: A simple parametric model based on data propagation. Data emanates from node \( p_0 \) and is being consumed by nodes \( q_0, q_1 \) and \( q_2 \).

In Figure 14.10, all the nodes \( (p_0, q_0, q_1 \) and \( q_2) \) belong to the class of so-called concrete nodes. I now introduce a new node type, a subjunctive node, to represent a choice among different (concrete or subjunctive) parameterized nodes — see node \( S \) in Fig. 14.11.

![Figure 14.11](image)

Figure 14.11: A simple subjunctive parametric model \( SPM \).
The word “subjunctive” labels *inter alia* a verb tense describing contingent (or wished for) states or actions. A subjunctive node acts as a collection of alternative nodes; another way to view such is as a node’s superstructure in the associated model. Figure 14.11 shows a simple subjunctive parametric model $\text{SPM}$, containing the *subjunctive parametric node* $S$ (indicated by red dotted rectangle), which acts as a collection of alternative *concrete nodes* $p_0$, $p_1$ and $p_2$. In this example, nodes $p_0$, $p_1$ and $p_2$ are concrete, but any subjunctive node can contain either concrete nodes, or other subjunctive nodes.

![Figure 14.12: All three possible realizations of the subjunctive parametric model $\text{SPM}$ shown in Figure 14.11 are the three concrete parametric models $\text{CPM}_1$, $\text{CPM}_2$ and $\text{CPM}_3$, depending on which alternative node ($p_0$, $p_1$ or $p_2$) has been selected in the subjunctive node $S$.](image)

**Realization.** Realization refers to the process of selecting a node in some or all subjunctive nodes contained in a graph. Realization applies to both subjunctive nodes and subjunctive parametric graphs. There are three types of realization:

1. *Node realization.* Node realization denotes the process of selecting an alternative, and applies to a single subjunctive parametric node. For example, subjunctive node $S_0$ in Fig. 14.11 can be realized as three concrete nodes: $p_0$, $p_1$, or $p_2$.

2. *Partial graph realization.* Partial graph realization denotes the process of realizing some subset of the set of all subjunctive parametric nodes in a graph.

3. *Complete graph realization.* Complete graph realization denotes the process of realizing all subjunctive parametric nodes for a graph, thus resulting in a conventional (non-subjunctive) parametric graph, which we also call *concrete parametric graphs*.

In other words, after a complete graph realization, there is no subjunctive node left in the resulting graph. Fig. 14.12 showcases three complete realizations of the subjunctive parametric model $\text{SPM}$ in Fig. 14.11; each of these three complete realizations result in three different concrete parametric models $\text{CPM}_1$, $\text{CPM}_2$, and $\text{CPM}_3$.

**Instantiation.** Instantiation refers to the process of instantiating all free variables of a completely realized subjunctive graph (i.e a concrete parametric graph), thus creating a state. In other words, the combined process of realization and instantiation produces all possible states for an arbitrary but fixed subjunctive parametric graph. Realizations produce a number of different concrete parametric graphs, and each of these can then be instantiated into many different states.
Figure 14.13: A subjunctive node can contain subjunctive nodes and/or concrete nodes. In this example, subjunctive node S0 contains concrete nodes p0, p1 and p2. Subjunctive node S1, on the other hand, contains one subjunctive node S0, and two concrete nodes p3 and p4.

Figure 14.14: The three possible partial realizations of the subjunctive model SPM00 shown in Fig. 14.15.

Figure 14.14 shows the three possible partial realizations of the model shown in Figure 14.13, resulting in the following three parametric models: one subjunctive parametric model SPM00, and two concrete parametric models CPM01 and CPM02, depending on which alternative node (S0, p3 or p4) has been selected in the subjunctive node S1. The subjunctive model SPM00 can be realized further as three concrete models shown in Figure 14.15.

Figure 14.15: Further realization of the subjunctive model SPM00 into three concrete models CPM000, CPM001, and CPM002, depending on which alternative node (p0, p1 or p2) has been selected in the subjunctive node S0.

Design states in the subjunctive pCAD model formalism thus come into existence through the combined process of realization and instantiation. Figure 14.16 shows how realization produces concrete parametric graphs (five in this example: CPM01, CPM02, CPM000, CPM001 and CPM002). Instantiations of concrete parametric graphs result in design states, here designated by five instances boxes (note that each box designates a different space of instantiations). Design states for the subjunctive model SPM0 is therefore the set-theoretic union of all instances of the five resulting concrete parametric graphs.
Figure 14.16: Design states in subjunctive parametric graphs are created through the combined process of realization and instantiation.
Discussion

In the formalism described above, to represent design states, one would perhaps wish to integrate the rather general model of states in “worlds” [399] (basically, following the “do all what your heart wants, without any consequences” approach), with the parametric shape grammars’ [355] disciplined notion of what it means to have a transition between states. Thus any act of sprouting a world could be identified with the act of applying a scheme, and all subsequent exploration of (parametric) shapes in a world would happen through parameter variation, and not yet another schemata (or rule) application.

And indeed, subjunctive graphs are capable of capturing exactly this notion, however without “launching” every realized concrete parametric graph into its own world. Let us suppose for a moment that we are able to do exactly this type of “launching”, and that we are able to control sprouting, through some set of schemata. Let us designate this hybrid abstraction (subjunctive graph + worlds + rules) formalism by letter $M$. By varying independent variables of a concrete parametric graph, we can now experiment and explore in our own sand-boxed world, and if we are not satisfied with the outcome of our experimentation, we can simply discard it, or backtrack.

One could, therefore, argue that world transition in $M$ is a “more significant” change, since it in general produces a configurational change (different configurations of subjunctive and concrete nodes), and for this we use schemata. An application of a schema sprouts a new world. In this world/state we then perform “less significant” changes, in the form of variable variations\(^1\). À propos, such an approach echoes a problem found in a current prominent (commercial) parametric system, where more significant changes are conflated with less significant changes; anecdotal evidence points to complaints of “not wanting to have all (variational) changes registered in the list of transactions” (which gives an indication of the relative importance of both types of change).

Another implication is that in this case we would end up with a shallow hierarchy of design states in $M$: a space of global states (entire worlds), and local states or “statelets” contained within worlds (obtained through different instantiations of a concrete subjunctive graph in its own world i.e. through varying values of independent variables). One could probably develop this idea further, e.g. arbitrarily deep hierarchies of states/scopes, or organizations of design states which are more complex than the simple graph-theoretical notion of nodes and edge.

\(^1\)Note that “more significant” and “less significant” is relative here. All changes in design are, in principle, equally important, only that the first type of change leads to a “revolutionary” (configurational, discrete) change in design, while the second type leads to an “evolutionary” (variational, continuous) change. Harada et al propose one way to holistically approach the problem of unpredictable transitions in [150].
Appendix C: Glossary

- **Active alternative.** An alternative that is currently in focus i.e., being modified directly by the user. See also: Alternative, Linked Alternative, Parallel Command.

- **Alternative.** Any solution to a given design problem. See also: Variation.

- **Branching.** A process of “exploring multiple paths” (mostly in parallel) [369], thus leading to the so-called “non-linear” model of design history. In single-state systems, this includes making copies of the current document and saving them under a different name, as well as inserting multiple alternatives within a single document (e.g., layers or side-by-side alternatives in image editing applications, or multiple chunks of text lined up one after another in text editing applications). See: Design history.

- **Collection of alternatives.** A synonym for Set of alternatives. See also: Set of alternatives, Alternative.

- **Command.** An instruction to the system or computational tool to execute an action; an user-invoked operation. It can be represented by an edge (arc) in the directed acyclic graph (DAG) model of design history, representing a transition from one design state to another. See also: Parallel Command.

- **Comparison of alternatives.** Refers to the process of identifying both the similarities as well as differences between alternatives [352].

- **Design alternative.** See Alternative and Variation.

- **Design history.** All design states visited by the user, or users. See: Design state.

- **Design state.** The totality of variables and their values defining a design, or computational model.

- **Experimentation.** A process of “discovering available options and testing out hypotheses” [369]. In the context of computational tools, in includes both (1) trying out different operations on a computational model, as well as (2) trying out different parameters for a specific operation.
• **Exploration.** The activity of searching for something, while not knowing exactly (or knowing only vaguely) what is being sought for, nor where to find it exactly. (See p. 218.)

• **Human-Computer Interaction (HCI).** Discipline “concerned with the design, evaluation, and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them”[4].

• **Interaction design (IxD).** The activity of “designing interactive products to support the way people communicate and interact in their everyday and working lives”[328]. IxD is therefore of wider scope compared to HCI, and in fact subsumes HCI [328, Section 1.3.1].

• **Item.** In the context of this work, a variable or a part found in a pCAD model. See also: Part, Variable.

• **Iteration.** A version of a specific design solution, “captured” or “seized” at a specific point in time, thus comprising a part of the solution’s “evolution” [370]. An iteration is the process of “incrementally solving portions of a problem, and building upon and combining these individual solutions” [369].

• **Linked alternative.** An alternative that is responding in some way to changes propagates from another alternative. See also: Alternative, Parallel Command, Unlinked alternative.

• **Matching.** Refers to the process of mapping similarities and differences among two or more design alternatives. In automatic matching, such relationships can be maintained for example by assigning unique part identifiers to equivalent parts across multiple alternatives.

• **Operator.** Interactive functionality for manipulating a visualization. Also known as “interaction techniques”, “tasks”, “operations”, “manipulations”, or “behaviours” [311]. An operator answers the “how?” question [49] of completing a user task.

• **Parallel command.** A multiplexed or subjunctive command, i.e., a command which leads to an effect across all the currently “linked” alternatives. See: Linked Alternative.

• **Part.** In the context of this work, a graphical item or a variable found in a pCAD model. See also: Item, Variable.

• **Revision.** The act of revising or altering (involving reconsideration and modification), or an object receiving such an act.

• **Set of alternatives.** A user-designated collection of alternatives [370]. See: Alternative, Collection of alternatives.
CHAPTER 12. CONCLUSIONS

- **Unlinked alternative.** An alternative that is barred from receiving commands propagated from other alternative(s) currently under modification. See also: Alternative, Parallel Command, Linked alternative.

- **User task.** The objective or activity that can support a user’s goal [311]. Answers the “why?” question [49]. See also: Operator.

- **Variable.** Within the context of this work, a graphical item or a part found in a drawing (pCAD model). See also: Item, Part.

- **Variation.** An alternative, i.e., a distinct alternative design solution. A variation of a solution is not the same as an iteration of a solution; variations “arise in moments of ambiguity and uncertainty, when the problem solver has inadequate information to choose one solution over another” [370]. An iteration is, on the other hand, a version of the same solution captured at a certain point of revision. I note here that an iteration can also be considered to be a variation (i.e., an alternative). See also: Iteration.

- **Version.** See: Revision.

- **Visual isomorph.** An isomorph, a term familiar in mathematics, is an entity that is similar or identical in some aspects to another entity. Correspondingly, a visual isomorph is “a representation of equivalent information in a different visual structure” [312], thus taking advantage of the fact that different representations can lead to different levels (from factors 10 to 100) of human problem solving performance [149].

**Additional Considerations**

While all the concepts listed above are important, I feel that the concepts of alternative, version, iteration, and variation, due to their ubiquity in this work, call for additional clarification and explanation.

- **Alternative.** Any solution to a given design problem. The concept of alternative thus implies a certain notion of choice; a person can choose among multiple alternatives, or alternative design solutions.

- **Iteration.** An iteration is a version of the same solution captured at a certain point of revision. I note here that an iteration can also be considered to be a variation (i.e., an alternative) if it is sufficiently different from the originating design [367].

- **Revision.** The act of revising or altering (involving reconsideration and modification), or an object receiving such an act. Further, this word denotes the identifier or attribute that distinguishes one closely-related design variation from another [293].
• **Variation.** An alternative, i.e., a *distinct* alternative design solution; M. Terry for example appears to use *variation* as a synonym for *alternative* [367]. A variation of a solution is not the same as an *iteration* of a design solution though; variations “*arise in moments of ambiguity and uncertainty, when the problem solver has inadequate information to choose one solution over another*”[370]. It may further denote the evolution of an item in the general sense, regardless if a formal revision has been issued or not [293]. In industry, there is “*little general support for using the term “variation”*”[293].

• **Version.** According to some sources, this is an “disfavored” [293] term, because it applies only to software releases; furthermore, it may be regarded as an “*optional alias for a revision*” [293] and that one “*cannot have a version without also specifying an associated revision*”.

Using *WordNet* [251], a prominent lexical database for the English language, my analysis yields the following:

• **Alternative.**
  
  – *Meanings*: one of a number of things from which only one can be chosen.
  – *A type of*: deciding, decision making.
  – *Types*: default; default option; druthers (i.e., the right or chance to choose); Hobson’s choice (i.e., the choice of taking what is offered or nothing at all); impossibility or impossible action (i.e., an alternative that is not available); obverse (i.e., the more conspicuous of two alternatives, cases or sides); opening; possibility; possible action; preference; soft option (i.e., an easier alternative).

• **Iteration.**
  
  – *Meanings*: Doing or saying again; a repeated performance; (computing) a single execution of a set of instructions that are to be repeated; (computing) executing the same set of instructions a given number of times or until a specified result is obtained.
  – *A type of*: physical process; process; repeating; repetition.
  – *Types*: (none listed)

• **Revision.**
  
  – *Meanings*: The act of revising or altering (involving reconsideration and modification); The act of rewriting something; Something that has been written again.
  – *A type of*: piece of writing; revising; rewriting; transformation; translation; writing; written material.
– **Types:** metamorphosis (i.e., a striking change in appearance, character or circumstances); modulation (i.e., the act of modifying or adjusting according to due measure and proportion [as with regard to artistic effect]); qualification (the act of modifying or changing the strength of some idea); reorganization (the imposition of a new organization; organizing differently [often involving extensive and drastic changes]); transfiguration (a striking change in appearance, character or circumstances).

**Variation.**

– **Meanings:** An instance of change; the rate or magnitude of change; An activity that varies from a norm or standard; A repetition of a musical theme in which it is modified or embellished; Something a little different from others of the same type; An artifact that deviates from a norm or standard; The process of varying or being varied; The act of changing or altering something slightly but noticeably from the norm or standard.

  - A type of: activity; alteration; being; change; idea; modification; organism; perturbation; physical process; process; saltation ([genetics] a mutation that drastically changes the phenotype of an organism or species); theme; thing; type.

– **Types:** allomerism ([chemistry]) variability in chemical composition without variation in crystalline form; departure; deviation; difference; divergence; permutation; replacement; substitution; transposition.

**Version.**

– **Meanings:** An interpretation of a matter from a particular viewpoint; Something a little different from others of the same type; (computing) a particular edition or revision of something (esp. software), or a number or tag denoting a particular version; A written work (as a novel) that has been recast in a new form; A written communication in a second language having the same meaning as the written communication in a first language; A mental representation of the meaning or significance of something.

  - A type of: internal representation; mental representation; interpretation; piece of writing; representation; turn; turning; type; writing; written account; written material; written record.

– **Types:** approximation; reinterpretation; retroversion (a return to an earlier state).
Figure 15.17: Is-a thesaurus graphs of important concepts, as per WordNet [251] and WordVis (wordvis.com).
Bibliography


420


[57] Bill Buxton. Sketching user experiences: getting the design right and the right design (interactive technologies). Morgan Kaufmann, 2007. 50


[103] Steven P Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L Schwartz, and Scott R Klemmer. Parallel prototyping leads to better design results, more divergence, and increased self-efficacy. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 17(4):18, 2010. 73, 76, 111, 131


[128] Dedre Gentner, Jeffrey Loewenstein, and Leigh Thompson. Learning and transfer: A general role for analogical encoding. *Journal of Educational Psychology*, 95(2):393, 2003. 77, 131, 169


BIBLIOGRAPHY


[250] George A Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956. 66


[283] Charles Sanders Peirce. The fixation of belief, 1877. 39, 40, 74, 276


[286] Peter Pirolli and Stuart Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. 85


[315] Bertrand Russell. History of western philosophy, and its connection with political and social circumstances from the earliest times to the present day. 1945. 39


[329] Mehdi Sheikholeslami. You can get more than you make. Master's thesis, School of Interactive Arts and Technology, Simon Fraser University, 2009. 103


[422] Fritz Zwicky et al. Discovery, invention, research through the morphological approach. 1969. xxiii, 125