Design ReExplorer: Interactive Design Narratives for Feedback, Analysis and Exploration

by

Rodolfo Francisco Sánchez Reyes
B.Sc., Instituto Tecnológico y de Estudios Superiores de Monterrey, 2003

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

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

© Rodolfo Francisco Sánchez Reyes 2014
SIMON FRASER UNIVERSITY
Fall 2014

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: Rodolfo Francisco Sánchez Reyes
Degree: Master of Science
Title: Design ReExplorer: Interactive Design Narratives for Feedback, Analysis and Exploration
Examining Committee: Chair: Jim Bizzocchi
                                                  Associate Professor
                                                  Halil I. Erhan
                                                  Senior Supervisor
                                                  Associate Professor
                                                  Robert Woodbury
                                                  Supervisor
                                                  Professor
                                                  John Dill
                                                  Internal Examiner
                                                  Professor Emeritus

Date Defended/Approved: September 10, 2014
Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the non-exclusive, royalty-free right to include a digital copy of this thesis, project or extended essay[s] and associated supplemental files ("Work") (title[s] below) in Summit, the Institutional Research Repository at SFU. SFU may also make copies of the Work for purposes of a scholarly or research nature; for users of the SFU Library; or in response to a request from another library, or educational institution, on SFU’s own behalf or for one of its users. Distribution may be in any form.

The author has further agreed that SFU may keep more than one copy of the Work for purposes of back-up and security; and that SFU may, without changing the content, translate, if technically possible, the Work to any medium or format for the purpose of preserving the Work and facilitating the exercise of SFU’s rights under this licence.

It is understood that copying, publication, or public performance of the Work for commercial purposes shall not be allowed without the author’s written permission.

While granting the above uses to SFU, the author retains copyright ownership and moral rights in the Work, and may deal with the copyright in the Work in any way consistent with the terms of this licence, including the right to change the Work for subsequent purposes, including editing and publishing the Work in whole or in part, and licensing the content to other parties as the author may desire.

The author represents and warrants that he/she has the right to grant the rights contained in this licence and that the Work does not, to the best of the author’s knowledge, infringe upon anyone’s copyright. The author has obtained written copyright permission, where required, for the use of any third-party copyrighted material contained in the Work. The author represents and warrants that the Work is his/her own original work and that he/she has not previously assigned or relinquished the rights conferred in this licence.

Simon Fraser University Library
Burnaby, British Columbia, Canada

revised Fall 2013
**Ethics Statement**

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,

or

d. as a member of a course approved in advance for minimal risk human research, by the Office of Research Ethics.

A copy of the approval letter has been filed at 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 2010
Abstract

Designers reflect-in and reflect-on their design process, for this they need a record of their design moves. In many cases the design process is as important as the design outcome itself. In parametric CAD (pCAD) this is particularly true as the record of designer actions are the elements on which the parametric model is built. Current pCAD systems provide designers with limited tools for recording, viewing or analyzing the design process. In this thesis I propose the Design Analytics framework to exalt the design process as an artefact for design; the framework consists of design process feedback, tools for analysis and enabling the re-exploration of the design space based on analytic reasoning. The Design ReExplorer prototype was developed to test these ideas and evaluate its insertion and viability in real-world scenarios through an expert panel study. The results of the study are favourable with positive feedback and multiple suggestions for future work.

Keywords: Design process; editable graphical histories; parametric modeling; visual analytics; interaction design; alternatives
To Sandra and Ámbar, my parents and brother; to my family
Acknowledgements

I would like to thank Halil for his unwavering support, his guidance and friendship during all these years of hard work, a fruitful partnership. I would also like to thank Rob for his insights, wit and thoughtful approach and John for his support in the final moments. This thesis couldn’t have been completed without my friends and colleagues. Thanks for making something quite hard a fun and fruitful endeavor through the studies, the TAships and during those long chats, emails and meals! This work was partially supported through the Boeing AeroInfo Visual Analytics Project and MITACS; the Graphics, Animation and New Media Network of Centres of Excellence of Canada; Consejo Nacional de Ciencia y Tecnología (CONACYT), Mexico; and JavOnet.com.
# Table of Contents

Approval .......................................................................................................................... ii  
Partial Copyright Licence ............................................................................................... iii  
Abstract .......................................................................................................................... iv  
Dedication ....................................................................................................................... v  
Acknowledgements ........................................................................................................ vi  
Table of Contents .......................................................................................................... vii  
List of Figures ................................................................................................................. ix  
List of Acronyms ............................................................................................................. xi  
Preface .......................................................................................................................... xii  

## 1. Introduction ...................................................................................................... 1  
1.1. Motivation and Research Goals .............................................................................. 2  
1.2. Summary .............................................................................................................. 10  
1.3. Thesis Roadmap .................................................................................................. 10  
1.4. Thesis Structure ................................................................................................... 11  

## 2. Background ........................................................................................................ 12  
2.1. Parametric modeling ............................................................................................. 12  
2.2. A Review of History Tools and Mechanisms ......................................................... 14  
2.2.1. Types of History Tools & Mechanisms ...................................................... 16  
2.3. Augmented History Tools ..................................................................................... 18  
2.4. Related Work ........................................................................................................ 24  
2.5. Section Summary ................................................................................................. 27  

## 3. A descriptive model of the design process ...................................................... 28  
3.1. Pilot study and graph-models ............................................................................... 28  
3.1.1. First iteration of the graph-model ...................................................................... 29  
3.1.2. The second iteration of the graph-model .......................................................... 30  
3.2. Process Measures ................................................................................................ 35  
3.2.1. Backtracking ............................................................................................. 35  
3.2.2. Deferral ..................................................................................................... 36  
3.3. Testing Design Narratives as models of the design process ................................. 38  
3.3.1. Part A: Observing design moves using SolidWorks ................................... 38  
3.3.2. Part B: Real-world GenerativeComponents projects ........................................ 38  

## 4. Making a difference with Design Narratives ..................................................... 42  
4.1. Design Theories to frame our research ................................................................. 43  
4.2. Design Narratives as Artefacts for Design ............................................................ 47  
4.3. From theory to practice ......................................................................................... 49  
4.4. Feedback ............................................................................................................. 50  
4.4.1. Local collections of alternatives ................................................................. 52  
4.5. Analysis ............................................................................................................... 54  
4.6. Re-exploration ...................................................................................................... 60
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Graphical history in Chimera</td>
<td>19</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Screencast</td>
<td>21</td>
</tr>
<tr>
<td>Figure 3</td>
<td>User interface developed by Makkuni</td>
<td>25</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Modeling action data with backtracking</td>
<td>30</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Second iteration of graph</td>
<td>32</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Schema A: Implicit history tree</td>
<td>32</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Schema B: Implicit history tree</td>
<td>33</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Second iteration with ‘part spans’</td>
<td>34</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Design Narrative graph with Deferral located at action a)</td>
<td>37</td>
</tr>
<tr>
<td>Figure 10</td>
<td>A sample GenerativeComponents transaction</td>
<td>39</td>
</tr>
<tr>
<td>Figure 11</td>
<td>GenerativeComponents and SolidWorks process graphs</td>
<td>41</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Three phases explained</td>
<td>49</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Design Narratives of GenerativeComponents and SolidWorks users</td>
<td>52</td>
</tr>
<tr>
<td>Figure 14</td>
<td>A local collection of alternatives</td>
<td>53</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Close-up of a GenerativeComponents narrative graph</td>
<td>55</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Backtracking and construction clusters</td>
<td>56</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Deferral in a Design Narrative graph marked with rectangles</td>
<td>57</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Analysis in a local collection of alternatives</td>
<td>58</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Histogram of values in a local collection of alternatives</td>
<td>59</td>
</tr>
<tr>
<td>Figure 20</td>
<td>The DReX prototype adds re-exploration to the static Design Narratives</td>
<td>62</td>
</tr>
<tr>
<td>Figure 21</td>
<td>DReX’s three stages of interaction</td>
<td>62</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Tower model used to test the DReX prototype</td>
<td>64</td>
</tr>
<tr>
<td>Figure 23</td>
<td>System and implementation design for DReX</td>
<td>65</td>
</tr>
</tbody>
</table>
## List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pCAD</td>
<td>Parametric Computer Assisted Design</td>
</tr>
<tr>
<td>DReX</td>
<td>Design ReExplorer</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Assisted Design</td>
</tr>
<tr>
<td>GC</td>
<td>GenerativeComponents (Bentley Systems Incorporated)</td>
</tr>
<tr>
<td>DA</td>
<td>Design Analytics</td>
</tr>
<tr>
<td>SW</td>
<td>SolidWorks (Dassault Systèmes SolidWorks Corp, 2013)</td>
</tr>
<tr>
<td>GH</td>
<td>Grasshopper (Robert McNeel &amp; Associates)</td>
</tr>
<tr>
<td>DAF</td>
<td>Design Analytics Framework</td>
</tr>
<tr>
<td>STM</td>
<td>Short Term Memory</td>
</tr>
<tr>
<td>LTM</td>
<td>Long Term Memory</td>
</tr>
</tbody>
</table>
Preface

I’m an introspective designer. I’ve always wanted to know how to be a better designer, to explain myself and teach others, and how to be better at what I love to do most. This is a personality trait that has allowed me to design buildings and houses, do research and, lately, to teach. It can also make me a slow designer, which I compensate for by occasionally making impulsive decisions. Obviously a well thought-out strategy.

In retrospect, I now see how the ideas presented in this work are important to me. I believe they satisfy me as much as they would other creative people. This became most apparent while conducting the interviews described towards the end of this thesis. I wanted to be a participant. I wanted to use the prototype and wanted it to become “real” in order to understand how this thesis was written and how I could have improved and managed alternative introductions, prefaces and the very difficult Design Analytics Framework section (more on this later or jump ahead if you dare).

And yet the ideas behind this thesis are relatively simple. Here is a sample: to know oneself; to understand history so as to not repeat the mistakes of the past; to see the importance of providing a quick word of advice to a colleague at a critical moment. A collection of ideas appears to be simple enough but are hard to transform into something tangible that can actually be used for something useful, for example, to design.

Of course economists, statisticians, historians, sociologists, day traders, politicians and even baseball coaches use an understanding of the past to make decisions. For architects, engineers, artists and designers in general, what tools do we have to understand what we have done? In school we are taught through feedback, and the process usually remains implicit. Design schools have programs built around core design courses that span four or five years and in each of those courses we are provided with one-on-one feedback by professors, teaching assistants and our peers, based on our iterations and the final outcomes. But that is usually all; the teaching remains implicit.

To understand how a design has come to fruition we must look at the choices that were made as well as the choices that were tried and discarded, and even those
that were never made. Only by understanding the reasoning behind these possibilities, within the context of each design, can we fully evaluate the true value of a design outcome. An introspective designer might stop at every step to think!

Looking back at my own practice I remember trying to reconcile the placement of a new building with an adjoining plaza and other buildings. One of the problems was moving between the buildings and how to avoid getting wet during the rainy season. We spent days going over possible solutions, reviewing alternative solutions, looking at the budget and talking to the client. We finally accepted that the two buildings would never be connected by a covered pathway. By accepting this we then looked for alternatives to reduce the impact this decision would have on the people using the buildings. Our solution was to recommend the purchase of copious amounts of umbrellas that we proposed could be left on either side of the plaza for people to use and then drop off once under cover on the other side. Our thought was that a plaza crisscrossed with multicoloured umbrellas would be better than having a textile/steel/glass-covered walkway dissecting the plaza.

As expected, once the construction was completed, criticism of the plaza and its designers began as soon as the rainy season set in, and probably continues today. Any substantial evidence to support our decisions is now lost. Most of its reasoning was never put down on paper and some is archived in old hard drives across multiple files. To those who ask, we explain our decision. Most accept our reasoning and tend to agree with it. They leave knowing that the problem was carefully analyzed and that many alternatives were considered even though there is no physical evidence to support that or to keep them dry. I believe people still think of us as incompetent when it starts to rain and they have to cross the wet plaza, even if they do it holding an umbrella.

In many cases you may disagree with the choices made, but knowing that alternative paths were considered and how they could affect the final solution if chosen should provide a greater level of respect and comfort for all of us who evaluate design from any perspective.

This thesis is about finding ways to transform the design process into a tool to help explain design to oneself, to others and, most importantly, to design.
1. Introduction

“That men do not learn very much from the lessons of history is the most important of all the lessons of history.” - Aldous Huxley

Like others, our concern is to understand what designers are thinking, and to do this we focus on what they do and how they do it.

Designers reflect-in and reflect-on their design process, and for this they need a record of their design moves (Schön, 1983) (Schön, 1985). When sketching, designers draw and make annotations which are used as precedents or references, and are later recalled and revisited. These records are more important to experts than to novices (Menezes & Lawson, 2006) (Goldschmidt, 2003). As these records accumulate they become an unstructured history or narrative of the design process (Makkuni, 1987). This narrative becomes important and useful given that the design process is sometimes as important, if not more, than the design outcome itself (van Dooren et al, 2013).

The search for knowledge and understanding within the design process has been a motivating agent for many researchers (Akin, 1986) (Lawson, 2006). A descriptive analysis of the design process using multiple models and methods has provided many clues to understand the inner workings of design and designers. Woodbury and Burrow state that design intent is best explained through a contingent history of the process (2006). This is a recurring idea throughout the thesis. The design process is the thread that runs through this thesis, guiding us towards our goals.

Designers rely on sketch books, construction documents, digital and physical models and other media, in addition to their own memory, to access their immediately preceding actions or to recall precedents (Woodbury & Burrow, 2006) (Akin, 1986). Computational tools can help record and enable sense-making of the design process based on some of these artefacts. While, designers can also rely on improvised historical mechanisms and user-defined strategies, these limit the ability to reach their
goals. The critical task to enable designers to systematically record, view and analyze—and eventually understand and learn from—the design process has not been undertaken.

This thesis presents a set of ideas to empower designers and other stakeholders by enabling them to make the ‘design process’ into an ‘artefact for design’ (Makkuni, 1987) (McCrickard et al., 2010). The ideas presented here and the prototypes/systems derived from them offer new opportunities to designers. The design process is transformed into an explicit artefact which designers can use to perform design tasks that until now were impossible to accomplish in an integrated and systematic way. The transformation of the design process from an implicit to an explicit artefact also stipulates a transformation from obscured to enlightened. As this occurs, the design process opens itself up to analysis, debate and inspiration.

This study developed a computational prototype based on these ideas and tested its principles and its viability in real-world workflows. This is important because design studios and corporate offices develop design projects that are far larger and more complex than those found in research labs. The prototype provides designers with a real-time interactive visualization of the design process, enabling an analytic process and a re-exploration of the design space. This work is based on a qualitative study with domain experts to evaluate the prototype. It presents the results of the study, a discussion and our conclusions. This prototype is a step forward towards filling a void in designers' toolsets.

The prototype adds a new tool which empowers and strengthens our existing toolset by organizing, making explicit, uncovering and exposing an artefact that was hidden and yet already embedded in the designers' workflows. The reasons to have such high level tools in our toolset are delineated in the following section.

1.1. Motivation and Research Goals

The ideas that motivate this research revolve in and around the design process. Other parallel and related ideas contribute to a greater or lesser degree: the craft of design, the search for domain-specific knowledge, alternatives and the tools designers
use. While the design process is long, complex and spans multiple media, and although most of it is occluded, it contains tangible elements that are open to analysis —collections of intermediate artefacts and design solutions in the form of sketches, models, schedules, files and notes. In addition to these, designers using computational tools leave behind footprints comprising discrete actions. Our motivation lies in observing and finding patterns within these traces, patterns in the design process that can eventually help designers and others with a vested interest in design.

The design process, we believe, has embedded in it important and relevant information which needs to be uncovered and exploited. While individually the actions designers take provide little information, in aggregate, the very actions that come together to generate a solution contain valuable information that can be saved, analyzed and shared. This information can reveal a great deal about the design process—one’s own or the process of others. A systematic analysis of this information can provide insights and enlighten what it means to design, unmasking the designer and exposing the tools used.

Current, creativity support tools, as described by Shneiderman, provide designers with a limited set of resources to keep a full and comprehensive history or record of their work (2002). While not comprehensive in its entirety, for our purposes a design history functions as a proxy for the design process (Mostow, 1985), i.e., it is sufficient to take its place. When looking at creativity support tools, we find that the majority of history tools are command logs taken from the command (Undo-Redo) stack. Their capacity is usually limited to linear records which do not allow designers to backtrack without losing subsequent work. These mechanisms are not intended to explicitly capture a design process and are mostly used to undo/redo actions stemming from simple slips and/or mistakes (Lewis and Norman, 1986) (Lewis and Norman, 1986). To work around this limitation, designers improvise solutions to record their design moves by saving multiple files, creating additional files, creating layers, cloning elements and editing these in series. These strategies are carried out in isolation or in combination with existing features already built into systems.

After reviewing existing systems we have concluded that history mechanisms within pCAD systems and research prototypes are partial or incomplete solutions to
understand the design process. Moreover, these solutions are not fully integrated into any system (Kurlander & Feiner, 1990), (Tidafi et al. 2011) (Grossman et al., 2010), (Nakamura & Igarashi, 2008). In order to help designers and other stakeholders meet their goals, creativity support tools and, in particular, pCAD systems need novel tools and features to create, manage and transform the design record into an augmented version of history mechanisms.

The need to capture and archive the design process and its importance to the development of a design has not been fully acknowledged by designers, and thus, not surprisingly, it has been ignored by developers and therefore not supported by systems. A creativity support tool that acknowledges these requirements together with the computational power to support the system can change how designers look at design (Shneiderman, 2002). We believe that decisions can be analyzed by looking at a design process. Later, with more arguments supporting each design decision, the designer may perceive new opportunities not present before.

Makkuni (1987) employs the analogy of the architect who uses tracing paper to easily explore and save copies of past decisions. Design decisions are saved temporarily and then discarded with every sheet. The used wads of tracing paper are kept as long as they remain useful to the designer as a reference, as a record of what could have been. Many questions arise from this analogy, including some previously asked by Makkuni: "What would happen if all the wads of tracing paper were saved and computational power was used to save, visualize and analyze them? How would this empower designers? What would these tools look like?"

These are just a few of the questions that arise when we start to unravel the design process. These start to become real and useful bits of information which can help us move forward in our understanding of design at various levels of thought (low to high).

A portion of the extracted information contained within the design process includes information on the design alternatives generated and their properties. The design process and, consequently, any model describing it will shed light on the design alternatives generated during the process. This is important because within the design
process are a myriad of hidden alternatives; some good, some better, and more importantly, some that have not been fully explored and are waiting to be uncovered and set free. One of the best methods to understand these alternatives, in particular those developed using parametric computer-aided design (pCAD) tools, is to unravel the design narrative that enabled —propelled, allowed, permitted or laid out— their creation. That is, the chronicle of the alternate paths that were never considered and the paths which, when reconsidered, become the correct path towards finding satisficing solutions (Woodbury & Burrow, 2006).

Alternatives are created by designers as they work and are an important part of the design practice (Lawson, 2006) (Cross, 2001). While alternatives for computational designers are just as important, they are also different. For designers using pCAD systems the generation of alternatives is an integral attribute of the design process because their creation requires relatively little effort, i.e., the cost (resources and time) of each new alternative is low. Parametric models can quickly generate a design space with explicit and implicit alternatives that is several orders of magnitude larger than what we are able to manage (Woodbury & Burrow, 2006). Ways to contain and manage alternatives are needed and the reasons to do so are clear. Designers use alternatives as a way to manage the uncertainty of a decision, assigning each alternative a parallel line of thought (Lawson, 2006 pp 297). In each line of thought the designer frames the design problem in a particular way, ignoring some elements and highlighting others. This allows expert designers in particular to move forward even when the solution is not entirely correct. Lines of thought are then merged into higher level ideas.

Woodbury and Burrow (2006) also suggest that designers tend to limit the number of alternatives they consider due to cognitive limitations. These limits can be expanded and supported through external representations and media. Computational tools can offload alternatives from designers’ short- and long-term memory and save them to secure and accessible repositories. More effective means of managing alternatives should increase the number of alternatives searched, improve the selection process and enhance designer performance.

There is great interest in finding new and effective ways to manage the plethora of alternatives that are created through the use of parametric (pCAD) tools (Woodbury,
Across domains, designers are often confronted with having too many alternatives from which to choose due to an increase in computational power and better and more extensive use of pCAD tools. The generation of alternatives within pCAD is one of its greatest features. In terms of cognitive dimensions (Green, 1989), as systems, parametric models have low viscosity when making parametric changes; structural changes would entail a semantic restructuring of the model—high viscosity. This radically changing the economics of design space exploration, as mentioned before. Because of this the designer’s task is transformed into one of containing, sorting and evaluating the desired solution among a large collection of alternatives in the solution space. The analogy is that pCAD is like fire, if left unchecked it can run rampant. Methods to manage and contain large collections of alternatives are needed to make pCAD a more effective and efficient tool. Although the number of alternatives is generally a sign of design expertise and good practice, it is also true that too many alternatives can be counterproductive (Cross, 2004). One possibility is that the cognitive load to support them is too high—either in terms of the quantity of alternatives or the number of steps needed to recreate or restore them—effectively disrupting the flow (Csikszentmihalyi, 1997 pp 110) of the design process.

Still pending is a complete paradigm shift for designers and an evolutionary jump in order for support tools to match the capabilities of systems at the forefront of innovation. We surmise that designers have already developed new abilities and strategies to cope with the influx of alternatives and to differentiate, annotate, share and compare alternatives. While challenging, this large number of alternatives also opens up new opportunities for designers and asks many questions of pCAD, and of design in general, which have yet to be answered. For example: How can we take advantage of these alternatives? How can alternatives be used for greater design exploration? What is the correct balance between too many and not enough alternatives? How should the load of the alternatives be distributed among the computer systems or the designer? Or how well are expert or novice designers coping with the proliferation of alternatives? Do we really need so many alternatives? Do experts ignore all but three alternatives?

Even with the proliferation of alternatives, designers still tend to work on a single state model that gives them access to only a single design solution at any given time (Woodbury and Burrow 2006), an expected occurrence given that this is what tools
provide. Yet users find ways around this limitation with help from the systems features or ad-hoc solutions implemented to circumvent this problem. Nevertheless these systems do not provide integrated solutions to use the implicit information within the system. Systems that provide some of the necessary features are restricted to research labs, such as *Parallel Pies* by Terry and Mynatt (2004) which allows designers to simultaneously explore multiple alternative design states in an integrated interface. The same could be said of Lunzer and Hornbæk’s (2008) subjunctive interface or the more recent work by Shireen et al. (2012) which focuses on subjunctive interfaces applied to parametric models, or CAMBRIA (Kolarić et al., 2014). In commercial systems, Adobe Photoshop has Variations, a separate interface which allows users to view multiple variations of a single image and edit them simultaneously to select the best one.

Systems tools are needed within pCAD to allow designers to search, track, filter, group, save and compare all the alternatives they desire. While some progress has been made, we have yet to see a clear understanding of how these tools would look and function in practice.

Another motivating factor is to understand how pCAD systems are used. To do this we must look at how designers use these tools and identify the benefits and challenges these tools bring. Like other tools, pCAD systems have a specific design process —not a single process but rather a process space— to which they are paired. In effect, the medium affects the process. An analogy can be made with a chisel and hammer, whose capabilities delimit a set of rules that sculptors then follow when working with marble. Likewise, the pCAD design process, with its own set of rules and properties, is determined by the pCAD tools themselves. The individual actions taken by the designer when designing using these systems compose the design process. This includes actions that go beyond what was originally intended by the tool. For the sculptor this means using the chisel to ram the hammer. Uncovering and disentangling the design process is therefore an important task towards understanding how we use these tools (within our expectations and beyond), how we can improve both existing and new tools and how we can best harness them to enable fervent design space exploration.
To manage a project’s history, there are user defined strategies—idioms of use—that circumvent the expectations of system developers. Examples include saving multiple files, making multiple copies of objects inside the same file or using layers. Users take control of their tools even if you do not want them to do that (Bhavnani & John, 2000) and (Bhavnani & John, 1996). For example, GenerativeComponents uses transaction files that enable users to partially interact and uncover implicitly saved history. For instance, users can open and edit transaction files to copy segments and save them, or edit values in a text editor instead of editing through the main interface. Grasshopper (a node-based visual programming add-on for Rhino) includes a local node recorder and tools to capture entire design states that can be restored. By saving states, users can evaluate alternate design states comprised of different parameter values. This has to be done explicitly since Grasshopper does not automatically save explored states. These are some examples of the different history/recording mechanisms found in systems today. Other examples of these strategies can be found in CAD systems as well as other tools. A list of user strategies can be found in Appendix A.

Regarding creativity in the design process, according to Dorst and Cross (2001) “Studying creative design is seen as problematic because there can be no guarantee that a creative ‘event’ will occur during a design process, and because of the difficulty of identifying a solution idea as ‘creative’. However, in every design project creativity can be found—if not in the apparent form of a distinct creative event, then as the evolution of a unique solution possessing some degree of creativity.”

For researchers looking to gather data on how designers work and think, a new set of tools would provide a fresh approach that differs from methodologies found in the literature. Previously the main focus has been on gathering data through protocol studies and other methodologies. Eastman provides an overview of the issues regarding protocol analysis, one of which is that it relies on information gathered from external sources (Eastman, 2001). Akin presents an activity-based model of the design process as part of his research on the Delft Protocols Workshop (Akin & Lin, 1995). In this study they observed six categories of data (drawings, writing, talking, thinking, examining, and listening) from multiple direct and indirect sources. Dorst and Cross (2001) also present a protocol study of industrial designers with data generated from
video-recorded think out-loud sessions, interviews and the design outcomes participants generated. More recently, Niblock and Hanna (2003) used a protocol study to gather data for their search to find differences between expert and novice designers during the conceptual stage of design by gathering data from design outcomes, interviews, questionnaires and audio recordings. In an attempt to find relationships between sketches and the design process, Do et al. (2000) ran a study to manually code architectural drawings by looking at state transformations depicting changes in the drawings. Gathering data in this manner is tedious and expensive and provides data that still requires filtering and encoding before running any type of analysis. The idea that data can be gathered directly from designers with no intermediary representation to impede or slow its analysis is critical to our research.

More recent studies in this and other fields have sought to take advantage of computational tools to generate the data, which would then help to understand the decisions made by the user, the process and the tools themselves. In his Ph.D. dissertation, David Akers (2009) showed how backtracking events such as Undo, Abort and Erase can be used to test usability problems with software by assessing them together with video capture data. Similarly, Xiaoxiao Ma et al. (2012) implemented a system on mobile phones that captures user interaction with the UI, gathering data that is then sent to be processed to obtain better usability testing data. Using UI event analysis, their study showed that their system was just as capable of discovering usability issues as traditional laboratory testing. Similar work done by Fourney et al. (2011) has allowed researchers and usability designers to analyze and solve interaction problems with software systems by conducting query log analyses of users as they search the internet looking for solutions. All these studies benefit from having data semi-automatically generated from systems deployed in real working environments with real design problems and users. Systematic methodologies that gather data straight from the designer will allow us to get closer to the designer, by not being present while they work, and shed light on processes that are often hidden in a black box.
1.2. Summary

This research aims to address some of the bottlenecks described in the previous section, to allow designers to re-explore past design states, recover from mistakes and learn from their and other designers’ achievements and errors. Therefore, it is essential to expand our understanding of the design process, the strategies designer’s use, the design patterns associated with the tools and tasks and the tools designers use. We do so by asking: How do small design decisions accumulate and congeal to form complete solutions? How can a single action devalue the success or exacerbate the failure of a given design solution? And how can this knowledge and knowhow be used to improve the craft of design?

Current pCAD systems do not provide an augmented design history. We have identified some of the bottlenecks in these tools below:

- Missing functions to navigate, edit, and explore past design states beyond simple Undo and file versioning.
- Lack of feedback on the development of a design task. The information provided is model- and state-focused and does not reveal what happens to the model.
- No explicit action records are saved or archived to provide insight into the design. Some tools only provide information on model structure and partial process records through feature trees or transaction logs.
- Limited retrieval of discarded design states that are valuable to understand and explain the design process. Such states may exist in a record but are easily lost in the noise of many minor changes and states.
- Difficult and labour-intensive identification of patterns within design histories. Existing tool features are not built to provide or encourage analysis in any meaningful way.

1.3. Thesis Roadmap

This is what we expect to accomplish in this thesis — our goals and intentions:
1. Understand how design exploration and the role of history mechanisms — such as proxies for the design process— have been studied by other research; and study related systems with which to build mechanisms to support histories in design.

2. Review the development of a descriptive model of the design process and the mechanisms designers would use to make them effective.

3. Demonstrate the implementation of a prototype and conduct a user study to evaluate how such systems are received by designers and other stakeholders.

1.4. Thesis Structure

After sharing our motivations and goals, the thesis presents relevant work found in the literature and in commercially available design tools. This primarily looks at the wide range of history mechanisms, their differences and similarities and the way designers use them. The building blocks that make up the core of the thesis are presented, followed by a description of our prototype which builds on these ideas. Lastly, an evaluation of the prototype is described and the results are discussed. Lastly, some final comments are included that give some perspective to the work.
2. Background

In the following section we explore history tools developed for commercial systems and those found in the literature, as well as time centric systems. We also review work that is most closely related to ours. This will move us closer to possible solutions to the issues raised in the introduction and in the motivation for this work. In this section we identify common issues and possible solutions for a system. These will help determine the characteristics of our prototype. To start off we present an overview of parametric CAD, which is one of our main building blocks.

2.1. Parametric modeling

Parametric CAD (pCAD) tools are a growing subset of CAD tools which are gaining popularity, mainly because of their ease in developing complex and adaptable computational models of the built environment. These tools allow users to generate a system of interrelated features, parameters and geometric elements which then define the final CAD model. These tools allow users to establish dependencies between elements in the system, allowing users to subsequently change parameter values or feature properties. When changes are made to a model, they propagate downstream through the system, automatically modifying and updating the model according to the new parameter values. Unlike conventional systems (AutoCad or SketchUp), this occurs without having to individually modify every element of the model independently. The capabilities of the system provide users with a new set of tools and techniques to create adaptable systems that can easily generate multiple alternative designs without the need to re-draw individual geometric elements independently. Generating alternatives in pCAD is a relatively easy and low-cost task.

The capacity of pCAD to push updates downstream is ideal for our research. It provides the data management structure to handle edits and pass them to the CAD
modeller, which will generate the final resultant geometry. The true investment is devising the parametric model, while the instances of the resultant geometry are comparatively inexpensive in terms of both time and resources; the exception being very large projects that require complex computation. What this entails is that the parametric model—that which can generate a great number of alternate solutions—is more important than a single solution derived from the parametric model.

The paradigm includes the concurrent design and development of two separate but dependent design artefacts, each with its own abstractions and representations (Aish & Woodbury, 2005). The driving artefact is the parametric system itself; the other is the CAD model—the resultant artefact—which is generated by the parametric system. For designers using parametric CAD tools the design process entails a concurrent cognitive process that needs to couple at least two different sets of representations, abstractions and objectives.

One of the benefits of pCAD tools is that they support parametric changes within a model once it is created. Structural changes on the other hand are not as simple to undertake without a major rework of the parametric model itself. Because of this, parametric modeling requires a broader and deeper understanding of the design task than conventional CAD tools whose structure is much simpler to disassemble and reconfigure as needed (Woodbury, 2010).

Parametric tools account for only a fraction of all CAD systems. Their capabilities make them suitable to create models containing a large number of related parameters which would otherwise be very hard or impossible to manage using conventional CAD tools. Because of this, pCAD tools are rarely used in the initial ideation phase or for sketching. And since a basic understanding of the design problem is needed before a parametric system is introduced, once the problem is sufficiently understood pCAD tools can be used to develop a design. That is not to say that pCAD tools necessarily require extensive work before providing results. Woodbury describes a collection of design patterns which demonstrates how designers can quickly generate models that provide results or that can be used to build on top of existing models (Woodbury, 2010).
2.2. A Review of History Tools and Mechanisms

History mechanisms are ubiquitous among today’s creativity support tools, and pCAD systems are not the exception. SolidWorks, Revit, GenerativeComponents, and Grasshopper all have mechanisms that traverse history to make changes to the current design state. Nevertheless, presently, history tools do not live up to their name in pCAD or in other domains. Ideally they would provide us with a rich account of what has transpired over a given length of time, similar to what we would expect from reading a book on the history of architecture. History tools provide us with much less than this. For the most part they provide a quick way to recover from slips and mistakes and do not provide a detailed account of transpired events. Even with this basic functionality their implementation in computer systems has become ubiquitous, in particular the very common functions — Undo and Redo. Multiple history mechanisms are present in the literature and have been implemented in commercial systems, from niche systems to mainstream commercial applications, web applications and mobile apps. Not only are they present to some degree in almost all current and future software but are also an expected feature. Today we expect systems to support CTRL+Z, the click of the undo button or jiggle the mobile device to undo the latest command.

History tools allow users to navigate linearly or, in some cases, traverse history to restore a past state by keeping a record of actions and states. While history tools have served as a means to keep a record of past actions, this function has been generally under-developed and underused in mainstream applications. What is clear is that history tools and mechanisms have great potential to help designers and to go beyond the basic functionality of the ubiquitous Undo and Redo commands found in many systems. Researchers and developers have already foreseen this and have succeeded in making more powerful history-keeping mechanisms (Pelaprat & Shapiro, 2002) (Kurlander & Feiner, 1990) (Hill & Hollan, 1994). Although research prototypes in niche domains often provide more interesting solutions with capabilities that surpass those found in conventional systems, they have not yet become an integral part of real-world workflows. Our interest lies in these history tools with advanced mechanisms and features. The following section reviews these systems.
The literature related to history tools is extensive. They are identified by Shneiderman as one of the basic tasks all creativity support tools must have (1996, 2002, 2007). Terry and Mynatt (2002) stress that more powerful tools and features are needed to navigate histories and manage alternatives in Parallel Pies. History tools have become indispensable in creativity support tools to the point that many users generate personal strategies on and around them to meet their goals (Bhavnani & John, 2000). This is also true in the pCAD domain. For example, Grasshopper, originally named “Explicit History”, has basic history keeping capabilities and also a feature to explicitly save selected design states. Users still use ad-hoc strategies to manage alternatives and restore past states by saving multiple files, by duplicating code within the same file or building parallel explicit histories though code. Similarly, in GenerativeComponents users can use transaction files as a pseudo history mechanism by saving and then editing them independently.

History mechanisms provide two levels of control over a project. The first is a temporal record and management of the project’s development; the second is a means to achieve control over design alternatives or parallel lines of thought. The first is intrinsic to history while the second is a result of the first. Although these two concepts are related they should be analyzed separately. Record keeping time is unavoidable and is a consequence of the process—the development of the project—while the generation of alternatives is a design strategy defined by the user. With the increase in computational power at our disposal the addition of more sophisticated history mechanisms are expected and should be encouraged. New history tools should be capable of supplying designers with enhanced capabilities that serve both these purposes—to enable comprehensive record keeping and managing and to support the multiple history paths that lead to alternatives. This we surmise will make tasks easier and design outcomes better.

As previously stated, design history serves as a proxy for the design process. It serves as a partial record of the design process, in both senses of the word “partial.” It is incomplete as it does not incorporate preliminary design stages such as initial conceptualization or hand sketching, or subsequent work done after the system is turned off. It is also partial in that history tools only record the information that is deemed important in the eyes of the developer, while leaving out other information. This partial
view of the design process can highlight or occlude elements that might otherwise be useful to stakeholders.

History tools and mechanisms serve an important function for designers and mainstream computer users. In their simplest form they are a means to solve slips through simple Undo. At a higher level history tools allow designers to explore and manage alternatives, recover from mistakes, test what-if scenarios, test tools and features and, ultimately, to record and replay design histories from the entire design session. These are the features and uses in which we are most interested.

2.2.1. **Types of History Tools & Mechanisms**

There are several types of history tools. A classification scheme based on functionality and how they are built follows. History mechanisms are built explicitly and implicitly into creativity support tools to allow users to knowingly or unknowingly manage a history of their work. Those that are created implicitly are generated as a by-product of other systems features, which are then appropriated by designers to manage the history of their projects. Implicit history mechanisms such as GenerativeComponents’ transaction files function as a history mechanism, although they were not specifically tailored as such. A combination of user-defined strategies and built-in features allows transactions files and the Transaction Stack to be used as a means to achieve some of the features expected of explicitly and fully implemented history mechanisms. While the capabilities of transaction files to function as history mechanisms are not perfect, they nevertheless hint at the possibilities of what we could expect from a fully developed history mechanism. In contrast, explicit history mechanisms are more prevalent, in particular if Undo and Redo are considered. We speculate that even though they are readily available their impact is not substantial compared to the potential impact that could result from augmented history mechanisms.

When developing history mechanisms, software designers have taken the underlying structures and capabilities in four directions. Action-based, state-based and hybrid systems make up the bulk of history mechanisms found in commercial and research prototypes today. These provide a limited amount of feedback and exploratory capabilities and are generally directed towards error correction. The last set of
prototypes (augmented or editable history tools) heads in another direction. These prototypes are based on a different paradigm, pushing history tools to go beyond error correction. A review of history tools and mechanisms follows.

1. Action-based history mechanisms keep a record of the individual actions executed by the user within the system. This record allows the system to reconstruct the past states of the project by re-executing or deleting actions. In doing so, users are able to linearly traverse the history stack, most commonly through the Undo and Redo commands. Backtracking to make edits is equivalent to deleting all subsequent actions. Examples include Adobe Photoshop, AutoCAD, Grasshopper, and Microsoft Word.

2. State-based history mechanisms, more commonly known as Version Control Systems or VCS, save an entire instance of the project state, which can then be restored, replacing the current design state. Some of the operations available when using state-based mechanisms include Merge, Checkout, and Commit. It is worth noting that these operations are not equal, their functionality varies across systems. State-based systems are storage-intensive as a copy is saved for each state, though some systems only save the differences between states to minimize this issue. Edits to previous states have to be merged with other versions to restore the current state, and conflict resolution when merging states is a common issue, which VCS have solved with varying degrees of success. Effective conflict resolution techniques are important to the development of future history tools and could be integrated to manage structural changes in histories (Bueno, 2011). Examples from commercial or open source systems include GIT, Autodesk Vault, or Google Docs.

3. Hybrid systems keep a record of both action and state levels of granularity, possibly providing the best characteristics of both mechanisms. An example is a history visualization tool developed by Heer et al. (2008) for Tableau.

4. As described in the literature, editable or augmented history mechanisms use a parametric backbone to push updates downstream when changes are made, most likely a propagation-based parametric system as described by Aish and Woodbury (2005). This allows users to change properties in past states which then propagate to the current state, updating it. This quality differentiates them from the other three categories. Examples include Chimera developed by Kurlander and Feiner (1992),
Bueno et al. (2011) and Nakamura & Igarashi (2008). These history mechanisms are the precursors to our prototype.

2.3. Augmented History Tools

Augmented history prototypes distinguish themselves from other history tools in that changes in earlier states update the current state. They are an evolutionary step above other mechanisms and the basis for our own prototype. They are different from other tools in two ways. Firstly, an underlying structure allows the designer to navigate to a previous state in time while also allowing designers to make changes which then propagate forward to change the current state of the design solution. The second difference is that, through these editable graphical histories, designers are able to modify how they design and impact their design outcomes. Through augmented histories designers start to gain an understanding of their design process, they replay actions and try alternatives without having to recreate subsequent states.

Chimera demonstrates this idea by creating editable graphical histories in an image editing environment (Kurlander & Feiner, 1988) (Kurlander & Feiner, 1990). It provides users with tools to view and edit past design states. As images are created Chimera builds a structure of past states which allows backtracking and branching by saving each design state. As changes are made new branches are created and previously undone actions are re-applied to the new branch to create a new state. The end result of the edit is shown at the end of a new branch. Users can view and edit history through a pair of panels which display the effects of user operations on the image. Panels display thumbnails of before and after effects of the relevant areas of the image using a comic book metaphor. Chimera uses heuristics to filter significant operations and group common operations together to create a hierarchical structure of high and low level operations. To make the most of history, Chimera allows users to replay past events to show their effect on the current state. Although Chimera's history model is structured linearly, it displays its history interface through pairs of thumbnails (Figure 1).
Chimera has features that are common to other history tools and are also present in our prototype. These include spatial browsing to navigate between the atomic actions using a comic book metaphor, hierarchical structures, heuristics to select semantically significant actions and varying granularity. Unlike Chimera, we do not use thumbnail pairs for all actions and have a full parametric data structure to update states.

Through Chimera the authors also introduce the idea of using macros. These dynamic macros could be shared and reused to help others, much like code snippets are used in programming or to generate alternatives by changing macro arguments (Kurlander & Feiner, 1992). Chimera demonstrates that history tools can start to more profoundly impact design. This makes this system and others like it the precursors to our prototype. The parametric engine of these systems is the key feature that allows users to make changes which then update the current state of the design.

The work by Bueno et al. (2011) shows another approach to manage history. Their prototype uses design operations performed by the user to create a data model of
the design process. This model then functions as an editable history mechanism that allows the user to backtrack and edit the design. Users can merge, generalize and specialize past operations to invoke change which then propagates through the data model, updating the current view of the design state. The system is currently implemented only on a collaborative drawing tool. The system distinguishes itself from VCS (state-based) in that it saves a record of operations (action-based). The history model developed to manage design operations is similar in structure to the pCAD systems. Unlike the majority of pCAD systems, this model is built in the background, hidden from the designers who use the system and becoming apparent only when editing the history. It hints at the possibility of adding new structural elements in a previous state, a common challenge that is still unresolved.

In Chronicle, the authors demonstrate action level recoding to create an interactive visualization of the design process for an image creation application (Grossman et al., 2010). Designers can use this tool to understand and explore the design process for raster graphics, allowing the designer to make better design decisions and to learn from both high and low level workflows. Chronicle provides a view into the past, to understand, learn or copy design elements by displaying saved video of the design process, which is then accessed using the history visualization. Chronicle includes tools for probing, filtering and brushing across the history environment. It does not have tools to identify design patterns or designer strategies or tools to explore and change the design itself. Chronicle uses video to create workflows which are indexed using revisions (similar to backtracking in Design Narratives, section 3), user interface events and an algorithm to assign hierarchical levels to groups of common actions (semantic chunking). The visualization includes multiple threads of high level UI events and a pair of thumbnail size design states that can be visually compared by the user; similar to those found in Chimera. In contrast to our prototype, detailed further on, these workflows do not include backtracking events to mark when designers return to a specific design state.

Chronicle has since moved from the research lab and is now available to download from Autodesk, renamed Screencast (Figure 2). The functionality of the system has remained the same, with the robustness and extra features of a commercial tool. Screencast provides users the ability to create screen captured videos of their work...
to share how the work is generated and how the tools are used. When used in conjunction with other Autodesk products, Screencast adds a timeline of keystroke events and dialogues to match the state shown in the video.

While Screencast and our prototype have some features in common, they remain distinctly different (see chapter 5 for more information on our prototype). Screencast does not capture any relationship between objects or related actions beyond actions associated by time stamp. For example, Undo is displayed as any other action while Delete is not included as an action. While much better than a regular screen capture tool, when used with AutoCAD Screencast still fails to capture the entire design process. The tool is better suited to record video tutorials.

![Figure 2. Screencast](image)

A test of Screencast shows the timeline. Most commands that are recorded are included in the timeline.

For Tableau, Heer et al. (2008) created a history prototype to record the data visualizations created in the system. In parallel, the prototype also gathers data to
evaluate Tableau itself. The prototype uses a hybrid approach, combining state and action data to create the history model by simultaneously saving VizQL statements and a command description stack. Text-based annotations are later used to contextualize the history interface. Chunking rules group low level actions together into higher level actions to provide better semantic meaning than that of individual actions. The system uses a branching history model for each worksheet. After users backtrack to a new state, a new analysis branch is added to the model and visualized. In contrast to Chimera (which uses pairs of thumbnails) and Chronicle (timeline with icons), this system uses a comic strip timeline of thumbnails to display design states. Branches are shown inline (an approach similar to ours) and timestamps are used to differentiate branches. The prototype also allows users to merge branches and bookmark, annotate and timestamp history items. It also includes other advanced features which should be considered when developing future tools such as semantic zoom.

Other prototypes have been less successful and demonstrate features that do not contribute to our own prototype. These include actions depicted through icons on a timeline of thumbnails in a CAD model (Su, 2007). Although sound, this research falters when transitioning to a prototype, in particular with the onscreen pictorial representations of history and states. Nakamura and Igarashi (2008) share ideas with Chimera and also introduce the idea of using a proxy to capture design moves. They also use icons/annotations on a comic book metaphor thumbnail, similar to Su’s prototype. In both cases there is no abstraction; actions related to the subject of interest are shown literally or through icons and annotations.

Research by Stones (2005) describes a visualization showing design moves to identify patterns within the colour palette selection process. Although in tune with our research, it is hindered by the domain itself. Visualizations include colour selection popularity and in some cases incomplete elements to represent temporal ordering of actions. The visualizations are visually attractive and effective but do not provide enough design process information to make a significant contribution.

These and other prototypes fail in part because they do not have access to the necessary data or decide not to visualize the relationships between each design move. In the prototype developed by Makkuni (1987) (see related work 2.4) and our prototype,
relationships between states exist beyond temporal order. These relationships are used to structure the history narrative by adding depth to the narrative, which in turn allows for better visualizations. In our prototype, backtracking adds depth to the visualizations. The model of the design process improves with an additional relationship between states (backtracking in section 3.2.1).

An idea made possible by augmented and editable history tools is that, like 'time-travelers', designers are not bound to work on single or current design states but also can work on potential states that were previously visited or missed entirely. That is why new tools are needed that support not only editing history but also simultaneous interaction with multiple points in history.

In contrast to history mechanisms, time-centric systems make time an editable variable of the system. History mechanisms record actions over time and then look back at them from a fixed timeframe. Time-centric systems also make a record of actions over time but then allow the user to change their temporal reference point to a time set in the past. In doing so, time becomes a malleable entity in the hands of a designer looking to “travel in time.” Below is a brief review of the tools that focus on time as a way to explore and manage content. These systems are used for email, desktop management and others tasks.

The Timewarp toolkit enables asynchronous, autonomous parallel collaboration between individuals (W. Edwards & Mynatt, 1997). Users can work on the same project independently from each other using different timelines, with a specific history for each. The user interface employs multiple timelines with which users can organize and coordinate their work. The interface shows history as a collection of states and the state that is readily editable. Users can traverse the timeline and edit states at any point in the timeline. The timeline is a tree graph with nodes and links, an acyclic directed graph. Nodes are discrete states in the document (project) and edges represent an action that is performed to achieve the next state. Similar to pCAD, Timewarp uses a parametric engine —as far as we can tell— to allow users to make changes which then propagate downstream. In Timewarp, the end result is a single integrated alternative where all branches are either merged or deleted, whereas in pCAD one explicit alternative is shown while other implicit alternatives remain easily accessible. Conflicts/paradoxes
between timelines are accepted as part of asynchronous collaboration. Of interest to our research is that, like other time-centric prototypes, Timewarp brings forth the notion that time can be explicit within the interface and that a parametric backbone can be persistently used to manage “time travel.”

2.4. Related Work

Woodbury and Burrow (2006) suggest that to know the telos (its ultimate aim) of a feature or final design is to look at the paths taken across the design space and the collection of alternatives that are left behind. Authors have previously identified similar issues in history tools as well as histories of action events as an area of interest. Headway has been made on both research and the development of prototypes. Their approaches have been varied and so have their results. These prototypes and their underlying research run parallel to ours.

Makkuni (1987) shows how the design process can be captured and visualized; transformed from a transient (ephemeral) state into an artefact to understand and supplement exploration with the use of computational tools. The design process as an artefact contains information about the design, the tools and the designer. It also becomes the medium for further exploration of the design space, as a predecessor to our prototype. In his visualizations, Makkuni exposes the branching of alternative paths to users.

Makkuni (1987) uses Chinese temples as a design case for which he developed a computational backbone to record and visualize the design process. The prototype provides designers with exploration tools to navigate across and from this historical record, in order to develop designs based on past and parallel states. Designers could develop and explore their designs by establishing parametric dependencies between the numerous compositional elements that constitute a Chinese temple. Changes made to parameters in the design process would update dependent parameters further along in the process. Designers could thereby develop a sense of history of their designs, where they came from and how they differ from alternative designs (parallel paths). This was visualized as a graph with edges showing the relationship between states (Figure 3).
Figure 3  

User interface developed by Makkuni

The user interface shows icons for each activity. Threads across the canvas represent alternative lines of thought in the design of the facade.

Another implementation of a similar idea was put forward by Derthick and Roth (2001). They began by looking at the record of user interactions as a data set that could be analyzed. They sought to uncover the paths users take by looking at how they moved across a process—not only the steps forward but also through backtracking. The system most closely resembles a state-based history model using a database to save states. New states are saved and loaded when necessary. To build their visualization they used the existing Visual Analytics software Visage, creating a navigable timeline to “time travel” from present to past states, where branching is possible and is shown through a graph and thumbnails. The interfaces enabled performing a side-by-side comparison of alternatives and an analysis of the overall
process though timestamps. Even though the system does not have a parametric backbone to enable updating states, the prototype demonstrates the power of having a strong data visualization system to support history.

Other researchers have used action level interaction data to uncover patterns and insight. Tidafi et al. (2011) use parametric objects as means to record designers’ actions and understand designers’ thought processes. The parametric objects also allow for alternatives to be compared and subsequent design exploration to resume once a decision is made from an intermediate design state. As described earlier, augmented history tools use a parametric backbone to update states. This prototype also develops its parametric backbone, but in contrast to other prototypes mentioned previously, it does so as a plug-in paired to an existing CAD system, which is similar to our own approach.

Akers et al. (2009) take a similar route when checking for usability problems in user interfaces. A large data set of actions is used to find indicators of backtracking. Akers’ research is important for our prototype because, as a user strategy, backtracking is an important marker or measure to capture the narrative of a design process. While designer actions move forward through time, backtracking moves signal designers to return to a previous state. A more detailed account of backtracking and how it relates to our project can be found in section 3.2.1.

Hill et al. (1992), and Hill and Hollan (1994) describe a collection of prototypes based on the idea of wear. Analogous to the wear of physical objects, digital objects also have wear and through prototype visualizations they are able to show the wear of the artefact. In contrast to others, their visualizations are created on the very objects of study. For example, a heat map is overlaid over a spreadsheet to show which cells have been edited the most.

Plaisant et al. (1999) generated a prototype with a visual history of replayable actions for collaborative learning in the engineering simulation domain. These “Learning Histories” are similar to other systems, with an emphasis on their capacity to transmit knowledge for education. They include collaborative tools to share histories with others and annotations to emphasize or share information (textual, audio or graphic). The
prototype shows two timelines: one for the actions users take and another as feedback based on the current state of the system. The prototype does not have a parametric engine to propagate changes within the timeline visualization but does save states and parametric changes.

2.5. Section Summary

Many of the history mechanisms found in the literature build structures that in many ways resemble those found in parametric systems. They are similar in terms of the data structure they create to manage history data. The most important of these features is the capability to make these history mechanisms editable, as in Chimera (Kurlander and Feiner, 1988). In the case of our prototype, the base pCAD system itself takes on that work, making it unnecessary to build an extra data structure. Many of the prototypes described in this section have useful features that contribute towards our prototype. Other elements needed to complete a prototype that satisfies our requirements are still missing. The next section outlines a model of the design process.
3. A descriptive model of the design process

To better understand the design process a model was needed that would allow us to analyze designer actions at multiple levels and build upon it. Design Narratives provide us with a representation of the design process which serves as feedback on what designers have done and as a model that is open for analysis.

Design Narratives are a descriptive model of the design process and our attempt to integrate our model and visual analytics. They are an abstraction that captures only what is needed and discards the rest. As mentioned in the background section we use Design Narratives as a proxy for the design process. The Design Narrative graph is an important component derived from our preliminary studies and also a crucial part of the Design ReExplorer prototype (DReX) which adds interaction to Design Narratives (Section 5).

Each of the preliminary studies through which the Design Narratives were developed also saw the consolidation of concepts that now shape the DReX prototype. Through the preliminary studies, and the development of the Design Narratives in particular, we explored the relevance and scope of the DReX prototype, especially with the introduction of interaction which greatly changes the way Design Narratives support designers and the craft of design.

3.1. Pilot study and graph-models

We conducted two studies to understand the design behavior of designers using pCAD tools and to envision potential solutions to support the design process using design histories tools. Below we describe the process we followed and the insights gained from each study.
Before we conducted the studies, we developed a design process-graph scheme as an apparatus to visualize and analyze patterns of use in pCAD. We also identified signals that reveal backtracking and deferral (more on backtracking and deferral in section 3.2). We obtained these by running an initial pilot study based on a single designer with advanced SolidWorks skills, who was asked to design a bus stop and a beach changing room. The encoding of the pilot study was based on the participant's feedback as well as the measures described by Akers and Akers et al. (2009a and 2009b). The pilot study helped us refine the design and encode guidelines for the full study. The following signals were identified as measures and used to encode data from both studies:

- Undo: Reversing the previous action performed
- Delete: Deleting parts of the pCAD model or features
- Add actions: Inserting new geometric features to the model.
- Modifying actions: Editing existing feature properties.
- Within and between states: Actions that are executed and applied within the same state or in between different states.

### 3.1.1. First iteration of the graph-model

The first process graph was created by encoding video of designers to reconstruct their design history using insights from the pilot study (Figure 4). It was an iterative process that consisted of researchers viewing, identifying and generating records of the sequential actions designers took while completing the design tasks. Actions are marked as nodes, and Undo, Delete, Add and Modify actions are shown as edges connecting the node where an action is performed to the initial node where the object being edited was first introduced. When designers performed backtracking actions they created variations (minor changes) or alternatives (major changes) in the model; variations are shown as edges while alternatives are shown as branches on the graph. The initial encoding was very detailed and captured more user actions than were necessary. This added noise to the data and masked relevant actions and patterns. The initial graph was not included in any further analysis except as a source of insight into the overall study process.
Figure 4. **Modeling action data with backtracking**

The first iteration of a graph to model action data (nodes) and backtracking (edges). The long backtracking edges towards the end of the process suggest the deferral of decisions.

This first iteration of the graph model was visually complex and generally ineffective as a model of the design process. We identified three key sources for this. First, many local edits (direct or short indirect edges) appeared to be simple error correction (caused by either designer or system) which were corrected using Undo or Delete. Second, distinguishing operations by type (revisiting variables, deletions and structure changes) suppressed the overall picture of change. Hence, we decided to remove suspected Undo actions and make all nodes and edges of one type. Finally, branching was not a good model for pCAD, primarily because the downstream propagation of changes created many branches in the graph, adding visual clutter and unnecessary complexity. Backtracking became the most salient behavior (horizontal lines in Figure 4) captured from this first iteration of the graph model.

### 3.1.2. The second iteration of the graph-model

The number of actions was reduced after studying the actions designers take in pCAD systems, their relationships, the visual representation and the overall structure. The only actions included in the graph became Add, Delete and Modify. Other low-level actions such as UI-commands, error correction, zooming or changing display styles were excluded. While these actions remain relevant to the usability of the system, as demonstrated in the literature, they provide little information about the design process itself.
The second process graph scheme includes three discrete elements of parametric modeling: constructs, backtracking and design variations (Figure 5). Constructs are all the actions designers take within a system to build the parametric mode, such as inserting a feature, creating a parametric relationship, or encapsulating a selection of low-level actions. Constructs are a subset of all the Add actions from the first graph. Backtracking takes the place of the Modify and Delete actions.

In this graph, constructs are shown as grey nodes. Backtracking nodes (larger grey nodes) are placed when designers backtrack and make changes to constructs, for example, by changing the value of a parameter or deleting a section of the model. With each backtracking node, a corresponding edge is created representing the relationship between the construct and its backtracking node. The third element is design variations, which are shown below the constructs as unfilled circles. These are a record of the parametric changes made by the designer either immediately after establishing them or after backtracking. We built the second graph using the encoded data from one of the participants in a study (part A) (3.3.1). The second iteration of the process graph is shown in Figure 5.
Figure 5. **Second iteration of graph**

The second iteration of the process graph is based on the abstraction of designer actions into a small defined set (constructs, backtracking and design variations). Actions that build the model are shown as small solid nodes. Backtracking nodes are shown as larger nodes and backtracking edges link both. Variation on the design are shown as simple circles.

In the second graph (above), the actions shown as nodes are ordered from left to right. The graph shows alternatives—branches in a conventional tree graph—which are created when signaled by explicit user intention or upon observation of a “major” change to the model. Distinct symbols identify revisiting variables and deletions. In this process, the flow of control is managed by the designer and system. This is described in Figure 6, which differs from the conventional tree graphs (rooted tree or arborescence) used to represent design.

Figure 6. **Schema A: Implicit history tree**

During any point in the design process (a) designers add new elements to the design (b). At this point the designer decides to change the value of a preceding element (c). The system updates all downstream states and takes the designer to the state where the change is initiated. (d) The graph keeps the record of change as a backtracking edge.
In schema B (Figure 7), each design variation is equivalent to a branch. However, the resulting graph does not show branching as it would appear in a tree graph (c), but rather, branches are implicit as in (d) in schema A. Given sequential nodes a, b and c and design state α, a backtracking edge from c to b will create an implicit branch and changes will propagate downstream. The value of c and all subsequent nodes will depend on the new value of b and the parametric relationships associated with b, creating design state β. Design state α would no longer exist as a model state; the explicit branch would become implicit. With this schema a tree graph with multiple branches can be collapsed into a graph with a single explicit branch and one or many implicit branches.

The flow of control shown in schema A reconciles the graph with the pCAD modelling paradigm, where the design process is in a constant state of flux. The fully collapsible structure of the process graph shown in schema B (Figure 7) is an important feature of the Design Narrative graph. This allows the graph to reduce its complexity (number of elements and relationships) and visual complexity by minimizing its footprint, a characteristic that becomes important as Design Narratives moves to the implementation stage.

Edges representing backtracking actions were changed from straight edges to curves (in this instance circular segments) as demonstrated in Thread Arcs (Kerr, 2003), as these disambiguate edge crossings and improve visual tracing in and out of nodes. This presented a problem during the implementation phase when designers backtracked from one end to another, creating a very large arc. The technique used in the prototype is similar to the one designed by Holten (2006), which helps reduce the overall space.
needed for arcs. The next refinement added part-editing spans under the graph representing work done on one of several parts in an assembly. These color-coded bars were placed below the graph corresponding to the appropriate node (Figure 8).

The ideas from schema A are fairly clear in the graph below, those from schema B are harder to grasp. From schema A, we know that the user from any point of the design process can add new elements to the model or backtrack to another point to make a change. At this point the system updates the model, adds a backtracking edge and takes the user back to the state where the change was initiated. From schema B, design variations are created when users backtrack and make changes. The small triangles below some of the nodes represent parametric variations, or alternatives, that were visited during the design process. These alternatives do not belong to the current state of the pCAD model, as in d) in Schema B (Figure 7); rather they are implicit alternatives of the pCAD model.

Figure 8. Second iteration with ‘part spans’
This graph was generated from a SolidWorks model where the designer used five parts which were later integrated into a single assembly. Each bar matches these parts.
3.2. Process Measures

After gathering action data from designers, we needed to know what to look for. Process measures are the markers we have identified to build the Design Narratives. So far we have identified two specific measures which will help model the design process.

The literature presents two high-level and interrelated strategies designers adopt when designing and working. The first is backtracking which refers to the action of a user or designer returning to a previously visited design state. As a measure, backtracking was used to identify usability issues in software (Akers et al., 2009). The second is deferral (Woodbury & Burrow, 2006) (Woodbury, 2010). This measure refers to a strategy designers use to model objects and relations approximately, knowing they will refine these later. Deferral is strongly supported by pCAD as the cost of deferral strategies is low.

3.2.1. Backtracking

Backtracking has different implications in pCAD than in other systems. Although backtracking is ubiquitous to many domains its importance to pCAD is paramount. It is a strategy that exalts pCAD’s virtues and differentiates it from conventional CAD tools and other creativity support tools. This occurs because pCAD systems allow users to make changes to the parametric model at any stage, which then propagate downstream and update the model. Akers et al. (2009) provides a taxonomy of reasons for backtracking related to software usability studies. This includes error recovery, exploring the interface, exploring design alternatives, revising temporary actions, understanding action consequences and reversing undesirable system actions. Our interest focuses on backtracking when it is used to explore design alternatives and when revising temporary actions; i.e. when designers create temporary model states that can later be deleted or edited once their purpose is exhausted.

Backtracking in pCAD can be expanded beyond the working definition given by Akers (2009), which only includes Undo and Erase as signals for backtracking. A broader definition is required to accurately map designers’ actions in parametric
modeling. In pCAD designers trigger backtracking as a process measure when they revisit a previously established parameter or feature and make a change. These modifications include adding new features, suppressing or deleting features, and simple Undo.

3.2.2. Deferral

The second measure is a design strategy that is a direct consequence of being able to backtrack. Deferral is defined as the act of postponing a design decision until warranted by the system, the user or the design itself. In pCAD deferral is a consequence of the parametric engine that allows the system to solve updated parameters. It is therefore closely tied to the very nature of parametric modeling (Woodbury & Burrow, 2006) (Woodbury, 2010). The low cost of backtracking within pCAD also makes deferral an inexpensive strategy for designers.

Deferral is also present in conventional CAD, although it is significantly different from that in pCAD. This is because backtracking and making changes is significantly easier than conventional CAD. In other words, the underlying logic behind a decision can be set in place when using a pCAD system while in a conventional CAD model only a spatial placeholder can be used. With a deferral strategy, designers build or use representations that can admit changes to earlier decisions without much change in the representation. We believe that the reasons behind the deferral strategy are determined by at least four factors. The first two are deferral of parametric values and deferral of the structural elements in the parametric system. The other two factors are the deferral of design decisions and the deferral of work. The deferral of a specific part of a CAD model is included within one or more of the four deferral factors.

In the example below, the designer backtracks to a specific node representing a Design Table at point a). This SolidWorks feature allows designers to modify parameters in an Excel sheet which is linked to the parametric model. In this example the designer repeatedly goes back and adjusts the values of each parameter to generate a different design solution. By adding this feature the designer is able to manage multiple parameters, generate alternative design outcomes and easily defer decisions. During the construction of the parametric structure the designer likely knew that the
initial values given to parameters were not going to be final. It was used as a placeholder in much the same way as headings are placed in text documents and are later filled in. We may conclude that a deferral strategy was likely used at point a).

![Design Narrative graph with Deferral located at action a)](image)

**Figure 9. Design Narrative graph with Deferral located at action a)**

The previous conclusion has its caveats. While backtracking is clearly shown as edges in Design Narratives, the same is not true for deferral. Certain patterns of backtracking may suggest the presence of deferral, but there is no definitive way to identify deferral based solely on backtracking. Likewise, designers may add features or build pCAD structures that enable deferral but this does not imply that backtracking will happen. Designers may ultimately decide not to revisit these features. In general, deferral can be considered a secondary measure while backtracking is a primary measure.
3.3. Testing Design Narratives as models of the design process

The goal of this study was to compare and contrast the design process graphs from two pCAD commercial systems—SolidWorks and GenerativeComponents—under both real-world and controlled design tasks conditions. The results would allow us to assess the value of the Design Narratives as models of the design process and evaluate our process measures (Erhan, 2012).

3.3.1. Part A: Observing design moves using SolidWorks

Part A of the study consisted of 16 participants who were asked to use SolidWorks to complete two separate design tasks over the course of approximately two hours. The participants were upper-division undergraduate and graduate students from the School of Interactive Arts and Technology (SIAT) at Simon Fraser University. All undergraduate participants had previously taken advanced design courses that included the use of SolidWorks. Graduate students had a design background. All participants were screened by filling out a questionnaire and interviewed to confirm that they were either intermediate or advanced users of SolidWorks.

For this study, participants were given two design tasks: a bus stop and a beach changing room. Our assertion is that these two tasks are comparable, given that the prototypical design of these two structures is similar in overall size, number of individual parts, spatial complexity, structural complexity and difficulty.

As a first step, we used the apparatus we developed to visualize the encoded process graphs of 7 out of 33 design solutions we had collected. These were selected based on the number of actions and number of backtracking actions, while others were selected at random. These process graphs are shown in the right column of Figure 11.

3.3.2. Part B: Real-world GenerativeComponents projects

The second part of the study used GenerativeComponents models collected from the "wild" as records of design work (Erhan et al., 2012). In GenerativeComponents, transactions files, or logs, are records generated by the system which includes the
discrete changes that build a parametric model. Transactions are an integral component of GenerativeComponents system functioning as they serve as a repository of data that define the parametric model. However, transactions are not necessarily part of a designer’s workflow; their use is optional, with some users choosing to develop their models without explicitly manipulating transactions. Transaction files are generated when GenerativeComponents is used. A single transaction can include individual design steps or groups of actions. A sample is shown in Figure 10.

```plaintext
transaction modelChange 'Add Floor_solid; change curve02, plane02'
{
  node User.Objects.plane02 Bentley.GC.Features.Plane
  {
    GraphLocation = {807.6, 21.2};
  }
  node User.Objects.curve02 Bentley.GC.Features.Curve
  {
    GraphLocation = {1084.0, 39.0};
  }
  node User.Objects.Floor_solid Bentley.GC.Features.Solid
  {
    Technique = 'offsetFromClosedCurve';
    Closedcurveroffset = curve02;
    OffsetAbovecurve = 0;
    OffsetBelowCurve = 1;
    AboveCurveDirection = baseCS.ZDirection;
    AutoHideInputs = true;
    Density = 1.0;
    GraphLocation = {1447.01055457457, -113.228175561609};
  }
}
```

**Figure 10. A sample GenerativeComponents transaction**

Transaction logs or files can contain many transactions depending on the scale of the parametric model. This sample transaction log fragment adds Floor_solid to the model and changes curve02 and plane02.

To generate the process graphs for GenerativeComponents, transaction files were parsed to extract relevant data. The parsing revealed some issues that are inherent in the files themselves. The first issue is that individual transactions ignored the order in which actions were made and sometimes included information that was either ambiguous and/or unnecessary for our purpose, such as minor edits. E.g. edits to curve02 and plane02 shown above are only related to their position within the dependency graph; a UI action, which is not included in the process graphs. Another issue is that GenerativeComponents transaction files are user-editable but do not record these modifications as a separate action and, therefore, they may not accurately reflect
all the work done in a project. To avoid this issue as much as possible we selected transaction files that were not substantially changed once the designer completed their design task. Care was taken to make sure that the criteria used in encoding SolidWorks and parsing GenerativeComponents files were similar, given the differences between the systems. The resulting GenerativeComponents process graphs are shown in the left column in Figure 11.
Figure 11. GenerativeComponents and SolidWorks process graphs

Design process graphs of GC and SW designers using backtracking as a marker (shown here as edges between nodes). Actions encoded from five different real-world projects using GC, study 2 (Left), and actions from participants using SW which created different solutions for the beach changing room and the bus stop in Study 1. Note: graphs are not at the same scale; these were scaled independently to make them fit into the figure, given that the number of nodes in each graph varies substantially.
4. Making a difference with Design Narratives

Akin (1986) says: “Representations are as much for evoking design ideas by providing stimuli with which to react as they are for assisting memory and recall. Often these reactions are critical in refining or further developing design ideas”. When sketching, designers get more from sketches than is put in through the process of backtalk (Goldschmidt, 2003). A similar pattern may be observed with respect to the design process but at a higher cognitive level. In other words, an internal dialogue is initiated between the designer and a process, in contrast to a single instance or artefact such as a hand-drawn sketch. This dialogue allows designers to extract information from a process that is underway or complete. Similar to expert sketchers, expert designers will be more fluent in their interpretation of the design process model and better able to extract relevant information, as suggested by Schön (1983, pp 104). It follows that if Design Narratives become prevalent in design workflows then they will provide an additional representation with which designers can converse, and experts would likely benefit most from the interaction.

We have taken a closer look at the design process within a pCAD system. As described in the previous section, we developed Design Narratives (Erhan, 2012) to tell the story related to the work—the design process. Design Narratives are a model of the design process which chronicles how individual actions and alternatives build upon each other and how different solutions come into being. The resulting Design Narrative model is a timeline/graph which shows a designer’s forward and backwards actions and the record of the alternatives that were created. The Design Narratives model was developed to uncover the internal and external relationships within the design process and reveal how they relate to other factors in a designer’s workflow. Design Narratives should not be confused with those proposed by Haymaker (2006) which are built to help define, share and integrate design and analysis processes. These models come from different perspectives, where the former is a descriptive model of the design process and the latter is a prescriptive model.
The Design Narratives model allows designers and researchers to confirm past hypotheses and ask new questions within the design research domain. The intention of the model is to contribute towards understanding complex phenomena within the domain. A task that is also undertaken by others including Eastman (2001), Akin (1986) and Dorst and Cross (2001). Models of the design process are important artefacts which help to understand the design process itself and develop better research tools (Mostow, 1985) (Akin, 1986). So far, research methodologies and tools, including protocol studies, have come with many caveats (Dorst, 2008) (Cross, 2011). More systematic studies with rigorous methods (Akin, 1986) (Dorst, 2008) should be emphasized. It is our hope that Design Narratives and systems similar to our prototype contribute to making this possible, specifically by changing how design studies are run and how design data are collected and analyzed.

When transforming the design process into an artefact for design, the new Design Narratives model provides two forms of information. The first is the Design Narrative with its timeline and backtracking edges and the other is the base data itself — the parameter name and value, among others. To be successful as artefacts for design, Design Narratives and our prototype (DReX) have to use this data to help designers and other stakeholders achieve their goals. To this end, we present the theories and mechanisms that explain how designers solve design problems and how Design Narratives, and later DReX, relate to these. See chapter 5 for more details about DReX.

For Design Narratives and DReX to be useful they must be supported by an underlying framework of concepts that guide their development, their transformation into artefacts for design and their eventual use by designers and others.

4.1. Design Theories to frame our research

Many design theories can help us explain how Design Narratives and DReX will benefit designers and describe how they will be used. The work by Schön and Simon stand at opposite ends. More recently other researchers have found common ground in hopes of identifying a more unifying theory of design. In the following section we briefly
describe these theories and how they relate to Design Narratives and the DReX prototype.

Design Narratives provide a new representation which designers and others can use for reflection-in-action and reflection-on-action, or both. These mechanisms were first described by Schön (1983) as part of his theory of reflective practice. Design solutions move forward based on reflection supported by problem setting and the framing of the design task. The mechanisms described by Schön make Design Narratives and the prototype on which they are based a promising representation for the design domain.

Schön (1983 pp 68) introduced the ideas of reflection-in-action and reflection-on-action as a counterbalance to the technical rationality or science-based perspective of the decision making process. These were meant to describe the mechanisms used by designers, and decision makers in general, to solve problems in real-world scenarios far from textbooks and purely technical expertise. Reflection includes the tacit knowledge and knowhow used intuitively by practitioners—those who practice a profession—to make decisions in a particular context and in dialogue with the problem. The skills and knowledge for reflection come from the experience of solving ill-defined problems in real-world environments.

In this context, practitioners include all those who practice a profession. In order to solve problems within their domain, they require a combination of technical knowledge and, to a greater or lesser degree, reflection. Practitioners include doctors, engineers, architects and lawyers, among others, all of whom learn to solve problems that have shared characteristics, while each is also unique. Just like a doctor, who in order to determine the appropriate course of action, makes decisions based not only on data but also on intuition resulting from years of experience. The resulting decision is based on scientific knowledge as well as on reflection-in-action.

Schön (1983 pp 68) positions technical rationality within the larger context of reflection. That is, it does not exclude technical knowledge resulting from reflection, which has proven to be so useful. It is through reflection that Schön describes how practitioners solve problems that are unique and situated far from the bounds of
technical knowledge. It is through reflection-in-action that Schön describes resolving problems with a continuous discourse between thinking and doing and by postulating and testing.

Reflection-in-action is largely based on the idea of framing. It allows practitioners to temporarily disregard certain aspects of the problem, giving them the freedom to explore ideas that previously would have been impossible due to the complexity and contradicting factors that surround design and other ill-defined problems. If a solution to the problem is found, partially or in its entirety, it can be reflected on and included in the rest of the design. Lawson and Dorst (2009) believe framing is likely to be Schön’s greatest contribution to our understanding of design. It is the key component of reflection-in-action which helps explain how practitioners solve ill-defined problems.

Lawson and Dorst (2009) also believe framing contributes to design expertise. That is most noticeable when expert designers are able to not only find solutions but also find ‘interesting’ solutions. For expert designers, framing entails switching between multiple frames, changing the qualities of the frame to test solutions, managing the interplay of frames and then knowingly select the most appropriate path to pursue. They participate in an active conversation with the representations, identifying solutions that would be very hard to find otherwise.

Argyris and Schön (1974, pp 19), in the context of theories in use, describe provide a useful diagram to represent single and double loop learning. In single loop learning a person learns how to resolve existing variable. In contrast, double loop learning, the underlying governing variables are changed. A useful diagram to represent single and double loop learning is shown below (Figure 12). For our purposes, designing takes the place of learning and framing takes the place of double loop learning.
While reflection-in-action is a dynamic undertaking, reflection-on-action focuses on what has already transpired. It is removed from the action (the design process) and requires practitioners to think back and remember how they solved problems. Reflection-on-action can be approached in two ways. Practitioners can actively recall the process and reflect on the events that transpired, or passively wait for insights and answers to come to them. In both cases, reflection-on-action shifts the focus from the current state of the solution to the process. This contrasts with reflection-in-action which looks at each step of the development process as it happens.

The work by Simon and Newell helps us look at reflection-in-action and on-action from a different perspective. From a human information processing perspective (Simon & Newell, 1971 pp 792), representations are used to help offload information from designers’ short term memory (STM), reducing their cognitive load. This allows them greater overall access to information and, as a result, enables them to reflect-in-action using more information to formulate and test ideas.

Representations are even more important for reflection-on-action than for reflection-in-action. It is through representations that reflection-on-action can be effective beyond designers’ STM and LTM. Through representations that focus on process, such as Design Narratives, reflection-on-action could possibly be extended beyond the capabilities of LTM and even shared with others.

Simon presents a scientific approach towards design which uses the tools of science to understand design and establish design as a science—rational, structured and systematic. Although sometimes Simon’s approach may be overly structured and
rigid for many design scenarios, his ideas and approach towards understanding design and designers is nevertheless useful.

Although Schön’s reflection-in and on-action stand out as a counterbalance to Simon’s ideas about design as a science, there is a middle ground and opportunities exist for both theories to coexist and help us to understand design. In search of these similarities and the middle ground between the two theories, others such as Lawson (2006), Dorst and Dijkhuis (1995), Cross (2001) and Meng (2009) have reconciled Schön’s and Simon’s differences. In particular, Lawson (2006) proposes an alternate design theory which explicitly builds on Schön ideas and tacitly on Simon’s.

Lawson’s (2006) model of designing finds a good balance between existing theories. It builds and unifies concepts to develop a more inclusive and flexible model. This model will help describe how Design Narratives and DReX can aid designers to achieve their goals. It will help explain the flexibility that is sometimes needed while still containing all the necessary elements to add rigor to the practice of designing, whenever necessary. The elements Lawson uses are categorized as: formulating, moving, representing, evaluating and reflecting (Lawson, 2006 pp 291).

The search to identify the most adequate model of the design process is an ongoing endeavor (Dorst & Dijkhuis, 1995). The two opposing views represented by the work of Schön and Simon, respectively— help us understand how Design Narratives and DReX will help designers but other frameworks could be used such as sensemaking.

4.2. Design Narratives as Artefacts for Design

It is through Schön’s ideas that Design Narratives can become powerful tools for designers. They are a new representation with which to reflect-in and reflect-on in support of decisions. Both Design Narratives and the DReX prototype support Schön’s reflection-in and on-action (Schön, 1983 pp 68) not only in terms of design outcomes but also by making the design process explicit.
As design tasks are completed, the Design Narrative graph will expand in parallel. A new opportunity for reflection-in-action becomes available after each step. To use Schön’s (1983 pp 61) terms, Design Narratives are artefacts which enable *deliberate* reflection-on-action. Although Design Narratives are not a representation directly created by the designer but, rather, a consequence of their actions, they should still work similarly to other representations.

Design Narratives provide low-level information in the form of sequential actions and higher-level information through process measures (backtracking and deferral). These two levels of information, high and low, are available for reflection and each can be used with in-action and on-action mechanisms.

Even without measures such as backtracking and deferral to enable reflection, Schön suggests designers themselves will make sense of the process. Under Schön’s paradigm a designer transforms into a specialized researcher of the practice and experiments in order to ask and answer questions (Schön, 1983 pp 68). By doing so, the designer’s moves the design solution closer to completion and also experiments, bring understanding to the process. This is similar to the idea that designers develop alternatives and parallel lines of thought (Lawson, 2006 pp 297) in hopes of finding better design solutions.

Schön allows us to use DReX in two ways. Through reflection-in-action we allow designers to use real-time information to frame their design task. Through reflection-on-action we generate additional information which allows designers to once again re-frame their design task based on the design process feedback provided by Design Narratives and DReX.

Some aspects are best supported by the model brought forth by Simon, in particular those that require a more scientific approach. Design Narratives and DReX should not only be useful to designers but also contribute to the development of design as a science. Design Narratives as models of the design process and DReX as a tool to systematically capture design processes can help us understand design. Design Narratives, design solutions and raw data can be archived and studied using scientific methods or any one of the design theories mentioned earlier in this section.
We believe Design Narratives are representations that can be observed through various lenses. They can provide a designer with feedback which the designer can interpret based on their domain of expertise, while at the same time they generate data that can be systematically analyzed to identify patterns and relationships. Conclusions obtained from one process can contribute to further reflection in other design tasks.

### 4.3. From theory to practice

For Design Narratives to be an effective tool, designers need to use it to analyze and reflect on their design process. To accomplish this, the Design ReExplorer prototype (DReX) supplies designers with a modified Design Narrative graph as the first phase of an interaction loop (Figure 12). Its function is to provide design process feedback that support design decisions designer can make decisions. The graph is generated using the same principles that were used to generate previous versions of the graphs (constructs, backtracking and variations). The data to build and update the modified graph is gathered from users' interaction with the prototype. This allows designers to continuously analyze or reflect on the design process and design outcomes as they design. To complete the feedback loop, DReX provides tools that enable designers to edit their design based on or independently of their analysis. Once an action is taken the graph updates, restarting the cycle. The cycle continues until the design task is completed. A complete description of DReX is available in chapter 5.

**Figure 13. Three phases explained**

Design Narratives allow us to complete two parts of the feedback loop. The third component of the feedback loop is re-exploration. This is enabled by the DReX prototype.
The feedback loop has three phases and two actors (DReX and the designer). The first phase is feedback. In this phase designers and other stakeholders receive design process feedback through two features found in DReX. The first is a modified Design Narrative graph and the second are local collections of alternatives (more details are available in section 4.4.1). The second phase, analysis, asks the designer or others to analyze or reflect on the feedback provided. There are various mechanisms to achieve this, as described in section 4.5. The third phase, ReExploration, takes place when designers or others decide to re-explore the design space. Re-exploration may be triggered by the previous phase, but not necessarily. The three phases are described in greater detail in the following sections.

4.4. Feedback

Receiving feedback on a design process is in many cases more important than receiving feedback on the design outcome. Feedback is the first step towards teaching, learning and deepening understanding and knowledge. Researchers have acknowledged the importance of feedback in various fields but have been unable to make a significant impact on the tools we use. The idea that changing a designer’s process has a larger impact than changing a single design solution is of great relevance and should be acknowledged. Shneiderman (2007) comes close when he suggests that researchers use data logging as a means to understand, and emphasises the need for "rich history keeping" in creativity support tools. Nevertheless, there is no reference of history being used as feedback; instead it is seems like a personal log instead of a direct and systematic evaluation of the process. While process feedback has not been a priority in system development, as Te’eni (1992) suggests, that may change with more computational power at our disposal which would make process feedback more readily available and provide new opportunities to use it. The DReX prototype is a system that makes use of this feedback.

Many systems provide low-level feedback, such as Microsoft Word which provides feedback on grammar or spelling, or IDEs which inform users of invalid syntax or unused variables. Meanwhile, higher-level feedback is harder to accomplish. Systems such as Rhino’s Grasshopper anticipate your next action by monitoring the
command stack looking for patterns using Markov chains and suggest a command you would probably want to use next. While these tools are useful, they do not provide higher-level feedback to allow designers or others to make a comprehensive analysis of a design process and, thus, to change higher-level design decisions.

There is little research in the decision-making literature that focuses on how feedback enhances decision-making. Most of the work on decision making and feedback has focused on end results (Te’eni, 1992). Nevertheless there are exceptions. For instance, in developing a new course to teach CAD, Bhavnani et al. (1999) developed Action Sequences as a way to make the design process “public.” In doing so they enabled instructors to give feedback to students based on their design strategies.

For our purposes feedback is defined as the supply of design process information to designers or other stakeholders, based on which they can complete an analysis and extract meaningful information to enable better decision making. Feedback can come in multiple forms. In this thesis, examples of feedback can be seen in the Design Narrative graphs and the local collections of alternatives. Other types of feedback that are possible, but are beyond the scope of this thesis, include heat maps, alphanumerical data, sound, haptic feedback or other visualizations.

Design Narratives provide users with design process feedback. They provide feedback by showing users a directed graph of all the actions they took to develop a design as well as the backtracking edges generated whenever they went back to make modifications to the pCAD model. Design Narratives are a model which shows designers what they did and how they did it. The examples below (Figure 13) presents two different design tasks, users and pCAD systems, which generate different feedback using Design Narratives.
Figure 14.  

**Design Narratives of GenerativeComponents and SolidWorks users**

Narrative a) is from a pCAD model developed using GenerativeComponents with many Action Nodes and backtracking edges. It contrasts with the less intense SolidWorks narrative, b). These narratives (a and b) model different systems, tasks, time spans, users and skill levels.

4.4.1. **Local collections of alternatives**

Our prototype, the Design ReExplorer (DReX), provides an additional category of feedback (section 5.3). This feedback stems from the implicit history tree in schema B, described in Figure 7. Within the implicit history tree is a record of the implicit branches and design variations that are generated after designers backtrack. In DReX, after a designer backtracks, alternatives are stored in local collections of alternatives (section 5.5). In the process graph, branches are implicit in the sense that they are not part of the current state of the model. They do however remain accessible to the user through the local collections of alternatives.

Local collections of alternatives are created when designers backtrack and make changes to a parameter. Each time the designer backtracks the previous alternative with its corresponding numerical value (if applicable) is saved into the collection.
Backtracking multiple times to the same parameter will create a larger collection of local alternatives. The example below shows both the model and the numerical state of each alternative (Figure 14).

Figure 15. A local collection of alternatives

For parametric objects that allow numerical values, a local collection of alternatives is possible as shown above. Models and values provide designers with feedback.

Similar to Design Narratives, local collections of alternatives also model the design process. While Design Narratives work at a global scale, the alternatives in these local collections tell the story at a local scale, i.e., at a parameter or node level. At this scale the design process related to local collections is represented with a minimum of components: the 3D models, their corresponding parameter value and the order in which they are created.

Local collections of alternatives expand the idea of Design Narratives. They provide process feedback at a different scale using the modified Design Narrative graph within DReX. While backtracking and deferral provide high-level feedback, local collections provide feedback from the low-level elements in the design process. A model of the design process that can represent this range will help designers as they use the feedback to make decisions. Local collections of alternatives allow us to potentially identify relationships between low- and high-level information and are important even when they represent low-level information, such as graphic representations. As is evident in the figure above, the top right image, a), has more immediate meaning than its corresponding 2.8 value on the right, in b).
4.5. Analysis

Analysis is the second phase of the feedback loop. In this phase the objective is to enable and provide tools for analysis in order to obtain insights and understanding based on feedback. Feedback and analysis are naturally coupled. Design Narratives are an example of feedback that can be used to analyze and obtain insights on a design process, a designer, a group of designers, tools or a design solution, among others. In the Design Analytics framework, analysis is the bridge between feedback and taking action through re-exploration.

Analysis can be accomplished at different levels. At a global scale analysis of a Design Narrative is achieved by viewing and comparing entire graphs or large sections of the graph. At a local level it can be achieved by focusing on smaller sections of the graph, or at an even smaller scale by looking at the local collections of alternatives, and at an atomic level we can look at the individual actions that build the pCAD model.

Through feedback designers are able to reflect on and analyze the design process and intermediate design solutions, based on the mechanisms described in section 4.1. Through these mechanisms designers may be able to gather information to make logical inferences and deductions about their design process at different levels and stages of that process. They are also able to use analogies, to reflect-in and reflect-on the design process, as well as share and compare, or experiment with different scenarios.

Large amounts of information can be condensed into a single narrative. Below we see a close-up of a Design Narrative with a small number of Action Nodes and a large concentration of backtracking edges coming from across the design process (Figure 15). Even without knowing the task, expertise level of the designer or value of the design solution, we can infer that these initial actions are sufficiently important as to attract a myriad of edits.
Design Narratives allow us to complete a rudimentary analysis of the design process. The initial actions taken by this designer have been changed multiple times from across the design process. Further analysis as to the origin and nature of this clustering is necessary.

Global analysis of the design process needs to be done on larger sections of the graph. In the example below, backtracking (empty boxes) and construction (solid boxes) clusters are highlighted in Figure 16. Larger spans of backtracking edges run across the entire graph while local edges can also be found. In this example there is no clear backtracking pattern. While there are concentrations of edges, they also span the entire graph and come in all sizes. More detailed information about the edited parameters, the designers and the design outcomes would make it possible to expand the analysis in the hope of making it more in-depth and fruitful.
Figure 17. **Backtracking and construction clusters**

Construction clusters are shown as solid boxes while backtracking clusters are shown as empty. The remaining is a mixture of backtracking and construction nodes. This behaviour shows how the designer switched back and forth between building the model and backtracking to make changes.

Deferral can be observed in the three areas of the graph below marked with empty boxes (Figure 17). From this information designers are able to make more informed decisions about their subsequent design moves. Long backtracking edges from the last third to the first third of the graph are consistent with the behaviour of other designers (Erhan et al., 2012). This may suggest that the designer is following a strategy similar to another, which would be positive if other designs were successful. While backtracking is based on data, deferral requires interpretation. Any conclusions based on deferral have to take this into account.
At a local scale, local collections of alternatives allow designers to run an analysis based on two properties. The first is the 3D CAD models themselves, by comparing and contrasting their attributes (scale, proportion, functionality etc.). And secondly, the model’s corresponding parametric values can be analyzed (see Figure 18). This analysis can be done in parallel with the 3D model and its value, or it can be done individually. In both cases, patterns, outliers, aesthetics and other analyses can help designers guide future design decisions. A description of how local collections are created can be found in section 4.4.1.

**Figure 18. Deferral in a Design Narrative graph marked with rectangles**
Statistical analysis of local collections of alternatives is another tool available to make sense of the data. Statistical tests and data representations such as histograms can help reveal patterns and opportunities for exploration. For example, the value 5.5 has not been evaluated in the parametric model, as shown in the example below (Figure 19). This may be accidental or the value of 5.5 may result in an error within the pCAD model. Many more tools are available to make sense of the numerical data. Given the scope of this thesis, we will only mention that statistical tools might be beneficial to Design Narratives or DReX.

![Figure 19. Analysis in a local collection of alternatives](image)
The analysis of Design Narratives and local collections becomes easier with experience; that is, after comparing multiple graphs and collections at different scales and with various characteristics. In Figure 11 (section: 3.3.2) we compare 12 different Design Narrative graphs from five GenerativeComponents and 7 SolidWorks users. All the graphs have elements in common, with the exception of g), as well as unique features. While these singularities can be attributed to many factors, they all represent user behaviours that can be identified and analyzed. The study participant whose design created graph g) is the easiest to analyze, although there is always uncertainty when evaluating a design process based exclusively on the Design Narrative graph. Two scenarios are possible. One is that the designer was both an experienced SolidWorks user and an experienced designer who knew precisely how and what he would design and simply executed it without ever having to backtrack. The other possibility is the opposite, a designer who just executed a design without concern for detail or iterating on their design. Further analysis of the process would require knowledge about the designer and the final design solution.

Each one of these graphs can tell us something. For the GC Design Narratives graphs in Figure 13 we can only analyze the resultant graph. Under different conditions, closer to the real-world, we would know the designer’s task, the current state of the design solution and the designers themselves, as we do for the SolidWorks graphs. Having access to this information would help to match the design process analysis with
the artefacts and the people who created them. In the case of graph g), without all the necessary information we can only speculate about what may have actually happened. All of this may be clarified if the design solution or designer were included in the analysis.

The analysis phase has the greatest potential to make a difference for designers. It is through analysis and reflection that the quality of a design can be improved, the process streamlined or unique solutions found. With further development it can also be transformed into a powerful research tool for people and systems. Of the three phases, this one is the hardest to control because it depends mostly on the designer. In contrast to the other two —feedback and re-exploration— we can only provide tools that will facilitate reflection and analysis and then hope designers can extract insights.

4.6. Re-exploration

The last phase is re-exploration. It is the endpoint of the feedback loop before the designer makes a change within the parametric model and restarts the cycle. While re-exploration is not possible in Design Narratives, it is possible in the Design Re-Explorer prototype (section 5). Re-exploration allows designers to take advantage of the knowledge and insights obtained through the analytic process in order to inform, inspire and guide them towards their next design move. This entails providing and/or enhancing the exploratory capabilities already within the pCAD system or developing new parallel systems that work together, such as our prototype.

In practical terms, re-exploration allows designers to make design moves or design decisions based on their analysis within the same interface that provides the feedback. In the case of DReX, re-exploring is done within a single user interface which provides both feedback and tools for re-exploration. This minimizes the user’s effort by reducing the cognitive distance between feedback and decision making. The features to accomplish this are explained further in the following section.
5. The Design ReExplorer Prototype

The Design ReExplorer (DReX) prototype is a computationally supported extension of Design Narratives. It was developed to satisfy some of the issues presented in the motivation section and serves as a bridge between theory and practice. It incorporates elements of Design Narratives and insights derived from an iterative design process using sketches and simple prototypes. The result is an augmented history tool for a pCAD system which includes a visual analytics component to help designers manage, understand and promote re-exploration of the design space from a process-centric platform.

While Design Narratives are static representations of the design process, DReX is an interactive prototype that uses a modified Design Narrative graph that grows in parallel with the pCAD model. DReX is a system that continuously enables reflection, and analytic reasoning through each step of the design process, an extension of Design Narratives capabilities. To do this DReX provides features for re-exploration of the design space by connecting it to a base pCAD system, which is accessible when actions are necessary or desired.

The integration between DReX and the base pCAD system enables users to move through three stages of a feedback loop: feedback, analysis and re-exploration (Figure 20). While the Design Narratives graph (a model of the design process) provides users with design process feedback and enables analytical reasoning and reflection, DReX allows users to complete the loop by enabling re-exploration of the design space.
A description of the three stages follows (Figure 21). First, users get design process feedback through a modified Design Narrative graph after an action is taken. This is equivalent to constructs in Design Narratives (section 3.1.2). Next, users optionally analyze, reflect or make sense of the feedback (analysis), or they move on. Users can then freely re-explore the design space, or not; and their reasoning may or may not include insights obtained through analysis. Any action within DReX would restart the loop by providing more feedback.

Figure 22  DReX’s three stages of interaction

DReX generates design process feedback after actions are taken. Feedback is then open for reflection, prompting action through re-exploration, which in turn restarts the loop.
The transformation of Design Narratives into an interactive system makes DReX a tool well suited for integration into a designers’ workflow. We believe it is capable of changing how some designers accomplish their goals and also contributes towards gathering data for researchers and others. DReX falls within the family of systems found at the intersection between creativity support tools and decision support tools.

The three stages—feedback, analysis and re-exploration—are an integral part of the conceptual design of the prototype and are pervasive throughout the user interface, interaction and user experience.

DReX provides design process feedback in two ways, through a computationally driven Design Narrative graph and by *local collections of alternatives* which add a new component to the graph and interface. The modified Design Narrative graph is generated with user actions within DReX and updates immediately after an action is recorded. It is dependant only on the speed of the system and the complexity of the parametric model. The prototype is designed to gather data from individuals working on a tower model which serves as a sample parametric model (Figure 22). Although design process feedback is primarily for designers, other stakeholders are known and encouraged to take advantage of this feedback to support decisions pertaining to the design itself or to other related issues. Decisions can be supported through ‘analysis’ which encompasses reflection and analytical reasoning. While DReX does not provide specific tools to help users interpret feedback, since it is visual in nature we believe the feedback is sufficient to enable reflection. ReExploration is also implemented through the modified Design Narratives graph, enabling users to interact with the existing pCAD system.
Figure 23    Tower model used to test the DReX prototype

Parametric model of a tower generated using Grasshopper and displayed in Rhino as a 3D model. The user can modify the shape of the perimeter, number of floors and their rotation using DReX.

5.1. System Design

The design of the prototype is based on transforming Design Narratives into an interactive model and on a series of ideas that permeate the entire thesis. While not meant to be comprehensive or complete, the prototype resembles real-world situations whenever possible. For that reason, one of the initial design decisions was to pair the prototype with an existing pCAD system. In doing so, developers and participants in the study were able to interact with the prototype knowing that the level of complexity of the models would be at par with the complexity of real-world projects. Further, this facilitated the completion of the system (prototype plus pCAD system) as a whole by eliminating the need to build the underlying parametric engine and CAD system to generate the 3D models. This allowed greater focus to be directed to the user interface, visualizations and interaction, and resulted in the research for this study having more ecological validity.
The prototype is a fusion of commercially available software and end user programming code. As a system, it comprises three main parts: the pCAD system (Rhino and Grasshopper), the user interface and data controller (DReX) and the elements that interconnect the system (Microsoft Excel, RhinoScript and the JavOnet API).

![System Design Diagram](image)

![Implementation Design Diagram](image)

*Figure 24. System and implementation design for DReX*

### 5.1.1. Flow of Data

The flow of data across the different subsystems will help explain how the prototype works. The data flow starts when a value is changed within DReX. This value is then passed on to an Excel spreadsheet using the JavOnet API. A Visual Basic script in Grasshopper reads the value in Excel and adds it to a predefined parametric model in Grasshopper. The value propagates through the results of the parametric model using two actions. First, Grasshopper updates the 3D model in Rhino and, second, it triggers a script within Rhino to take a snapshot of the 3D model. DReX then loads the image, updating its UI, and creates a new backtracking edge between the parameter and the present state, thereby completing a full data cycle.

The prototype is set up to function with an existing parametric model in Grasshopper (see appendix C). This means that the initial state of DReX is fixed and
editing in the Grasshopper model has no effect on DReX (see section 5.6). Future versions of the prototype would gather action data directly from Grasshopper or any other pCAD system; most likely from the command stack (Gamma et al., 1994).

In DReX, designers can see a history of the actions used to create a parametric model as well as the feedback provided by a Design Narrative graph and the local collection of alternatives. Through sliders or textboxes, designers can make changes to parameters to search for alternative design solutions. Each change made to the Grasshopper model results in a modification to the Design Narrative graph.

5.2. The Implementation

DReX was developed in Java in combination with a set of libraries, including the Processing core API and the G4P user interface library (Lager). While the prototype has been patched together using multiple systems, programming languages and APIs in order to streamline the functionality of the prototype, future versions should have a direct connection to the host pCAD system.

5.2.1. Grasshopper and Rhino

Grasshopper is a parametric modeller plug-in for Rhinoceros 3D CAD software (Rhino) (Robert McNeel & Associates). It is a visual programming language which allows users to drag and drop components onto the canvas to create visual generative algorithms. Data is passed from one component to another using “wires” that run from one port to another. As in other parametric CAD systems, in Grasshopper users create 3D geometry which is easily changed by manipulating the values or properties of the parametric components. In contrast to other systems, Grasshopper does not display the geometry but rather passes it on to Rhino to display.
While Grasshopper is different than other parametric systems in terms of interaction, it is very similar in other respects. It was selected as the base system primarily because of the ease with which it could connect to the other systems needed to create the prototype, as well as because of our familiarity with its use.

The Grasshopper model used for this study can be replaced with other models, such as the one in Figure 24. During the prototype’s development, several parametric models with varying degrees of complexity were used to test the prototype’s features. Although these were relatively simple models, the prototype’s operation suggested we should continue using a simple model. Though simple, the tower model could still cause DReX to lag when manipulating certain parameters.

5.2.2. Connecting Elements

Connecting code between the main system components passes data from one component to another. The first segment of code runs between DReX and Microsoft
Excel. This is done using the JavOnet API, which connects DReX to Excel via a .NET dynamic link library (dll), allowing values from DReX to be read and written in MS Excel. JavOnet lets users connect Java applications to others using the .NET framework.

5.3. Default setup

The prototype was designed to test key concepts, user interactions and specific use cases. It has very clear limitations; specifically, it is designed for a specific time span during the design process, i.e., while the designer is using a pCAD system. The prototype is built to present users with a default design scenario through which they can explore the design space based on the existing model and the feedback provided. Feedback is provided via a Design Narrative graph and local collections of alternatives.

In the implementation of the prototype, action data (constructs, backtracking or variations) is not directly gathered from the pCAD system (Grasshopper) as would occur in a fully functional system. Data flow between the systems is one-sided, going from DReX to the other connecting elements and ending with Grasshopper.

The default state of the prototype was defined by deconstructing the sample parametric model (Tower) and having the prototype load the default state when starting. To recreate the action data, the Grasshopper nodes and wires were used as guides to populate an action data collection, similar to those gathered in previous studies. This included the node name, relative position, current value and backtracking information. This data allowed us to create a base Design Narrative graph of sequential actions onto which backtracking edges could be added. The edges were added to resemble the characteristics of other Design Narratives created in our previous studies (section 3.1.1 and 3.1.2), together with our own insights in designing and building the sample Tower model itself.

In the default setup, local collections of alternatives are empty even though they should show the 3D model as matching the initial value set to the parameter. After backtracking and editing a parameter value, the local collection should contain two thumbnails.
5.4. User Interface

The DReX user interface is an extension of the ideas brought forth by the Design Narratives. Three important ideas—re-exploration, analysis and feedback—are represented through features, interaction and the UI design (see Figure 26). The central component of the UI is a modified Design Narrative graph (comprised of Action Nodes, Backtracking Nodes and edges) which runs horizontally across the screen; a timeline. As described in the Design Narratives section (section 3), the graph includes the actions made by a user across time (forward actions and backtracking). This graph serves as a base upon which other Design Narrative features and the re-exploration components are built.

The lower third of the UI is dedicated to the Design Narrative graph, which contains backtracking edges that connect Backtracking Nodes to Action Nodes. Action Nodes represent all the actions designers take within the system. These are equivalent to constructing nodes in Design Narratives. Likewise, Backtracking Nodes are similar to those found in Design Narratives (see section 3.1.2).

The top section is dedicated to Action Nodes and their local collections of alternatives. Each Action Node has a set of controls from which the user can re-explore the design space or interact with alternatives saved in the local collection. This means that users can change the values of the parametric model by moving a slider or entering a value in a text box, effectively changing the resultant geometry.
Just as in Design Narratives, Backtracking Nodes are created after a designer makes a change to the parametric model through any of the previously created parameter. These Backtracking Nodes are placed on the far right of the timeline, i.e., in the present moment. As mentioned earlier this only happens when the designer makes changes to an Action Node. Future versions would also include backtracking within the base parametric system (Grasshopper in this case). Together with the Backtracking Node, a backtracking edge is drawn, joining it with the Action Node that was just edited.

Action Nodes are an expanded and interactive version of nodes in Design Narratives. They contain two ideas which are important to the success of the prototype: re-exploration capabilities and feedback. Feedback is provided through a local collection of alternatives and re-exploration is provided by a slider and a direct input textbox. The collection of alternatives is made up of thumbnails and their corresponding parameter values, as previously described in section 4.4.1. All past alternatives are saved here for later use, either as an alternative or as support of an analysis. Changes to the parameter value within the Action Node are equivalent to updates made to the same parameter in the base system (Grasshopper in this case).
is edited a new local alternative is created and saved to the local collection of alternatives. A new backtracking edge is also drawn from the Action Node to its matching Backtracking Node in the Design Narrative.

Once changes are made in the Action Node the new value propagates down the parametric model, updating the final current state of the model. This state can only be seen within Rhino. The current version of the prototype does not update all the thumbnails for every 3D model in every Action Node. This decision was made during the implementation phase since it greatly increased the amount of work needed. Future versions of the prototype would update all images in Action Nodes when necessary.

In the example below, a single Action Node contains four distinct parts (Figure 26). These are, from top to bottom: a local collection of alternatives, an image of the last updated version of the parameter, a slider and direct input to edit parameter values, and the Design Narrative graph with its backtracking edges. Each Action Node is connected to multiple potential backtracking edges since Action Nodes can be revisited multiple times. In the example below, the parameter 'Height' has been updated to 59. A new image appears with the updated model and a thumbnail image is saved to the local collection of alternatives.
These are equivalent to a construct node in previous Design Narrative graphs (see Figure 5). On the right side of the figure is the name of each component. On the left side are the capabilities the node provides.

Branching in the prototype’s Narrative graph is identical to the original Design Narrative graph. Branching is implicit, meaning that branches are hidden within the main trunk. Adding a new alternative to the Action Node is equivalent to creating a branch as described in schema B (Figure 7). This is possible because the parametric engine within Grasshopper, and the prototype, ensure that all branches in the conventional “tree graph”—the alternative paths and solutions—are saved.

Even though these alternate paths and solutions are saved there is no way to restore them. Currently, the prototype only saves the value of the active Action Node. A complete restore would require all parameter values to be saved together with the 3D model image. To restore a specific saved solution the designer could then select an image.
5.5. Use cases

The Design Analytics framework and the use cases in DReX are paired. The three core threads—feedback, analysis and re-exploration—were guides to develop the prototype. Use cases are equally aligned with the framework. The following section describes a selection of the use cases for DReX. The term “user” includes designers, managers, instructors, clients and other stakeholders in the design process and outcomes.

1. Users view and find specific parameters within a history of actions. The system provides them with an Action Node on a graph for each of their actions and a local collection of alternatives within each node. A node is created for every significant action in the main pCAD system (Action Nodes are preset in the DReX prototype). Users can use the mouse to pan across the graph to see all nodes and their collections (Figure 27).

![Figure 28. History of actions and local history of parameters](image)

Action Nodes include the parameter name they represent. These are sequentially placed across the timeline/graph. Each Action Node contains a local collection of alternatives also placed sequentially from left to right and bottom to top.
2. Users identify parameter values within a local collection of alternatives that have not been tested previously. DReX provides a local collection of alternatives for every action parameter within the Design Narrative, (a) in Figure 28. Local collections contain a thumbnail of the CAD model and its linked numerical value (b). Users can view and access the value of each parameter individually. Local collections are empty in the default setup, although they should contain the initial 3D model and link value.

3. Users analyze local collections of alternatives. The system provides thumbnails and numerical values of alternatives within each collection to be analyzed by the user (Figure 28). Users can see the value that is linked to any of the alternatives found in the local collection by clicking on the thumbnail. In the image above, the Line Length is set to 403 and is displayed as shown in (b).

4. Users want to control the number of alternatives added to the local collections. The system provides a toggle switch to control whether or not alternatives are saved, as shown in (f) in Figure 28.

5. Users want to save specific alternatives when they are not automatically saved. The system provides a control to save the current parameter value to the collection. The
user can request that the current value be saved as an alternative, (g) in Figure 28. The system computes the new model and displays an updated version of the current parameter state (h) and adds a thumbnail to the local collection (i).

6. Users change the parametric model. The system provides users with controls to change the parametric values, which users can edit at any time. The system computes the new model using the new values, displays an updated version of the local parameter state, (h) in Figure 28, and adds a thumbnail to the local collection, (i) in Figure 28.

7. Users search and identify backtracking clusters or the lack thereof. Users pan across the graph and view the density of backtracking edges as they link to Action Nodes a) and Backtracking Nodes b) in Figure 29. Different densities are shown in j), k) and m). Based on this, users can perform an analysis to identify the deferral of work or decisions (Section 3.2).

Figure 30. Global analysis using backtracking edges in DReX and Design Narratives

A more thorough explanation of the features found in each Action Node is described below and correspond to Figure 28.

- In a), a local collection of alternatives is shown for the parameter LineLength. Eleven alternatives have been tried before and are shown as thumbnails. The parameter of the latest alternative is set to 403 and its corresponding geometry is shown as a large thumbnail at the centre of the Action Node. The Action Node itself and the local collection of alternatives provide design process feedback to the user.
• In b), the thumbnail for the latest alternative is highlighted and its corresponding parameter value is displayed. The user can click on any of the thumbnails to see their corresponding value and can compare the thumbnail and reflect on their next design move.

• In c), the Action Node shows the effect of an edited LineLength parameter. The change in the parameter value is shown by the addition of a new small thumbnail to the local collection of alternatives and a larger thumbnail at the centre of the Action Node.

• In d), a parameter value can be edited by typing a new value in the textbox and hitting the Enter key or pressing the button. This will update the LineLength parameter value in Grasshopper and create a new set of thumbnails which will update the Action Node, as shown in b) and c). The textbox enables immediate re-exploration of the design space.

• In e), the user can use the slider to edit the parameter value. The edit does not take effect until the user releases the mouse button. The effects are then the same as in d). The slider enables immediate re-exploration of the design space.

• In f), the user has the option to enable or disable the saving of thumbnails when exploring different parameter values. The user can deselect the checkbox and move the slider, or enter values into the textbox without the local collection of alternatives overflowing with thumbnails. The large thumbnail is always updated.

• In g), a Save button can be pressed to override the checkbox in f). This will save the large thumbnail to the local collection of alternatives. Clicking this button does not override the checkbox in f).

• In h), a large thumbnail is shown of the latest parameter value explored by the user. It corresponds to the parameter value set to 63 in e). A smaller thumbnail is added to the local collection, as shown in i). It provides the user with feedback from the latest edit and contributes to the design process feedback as a whole.

• In i), a new thumbnail has been added to the local collection of alternatives after the parameter value was set to 63 in e). This serves as a record of the alternatives previously tested, contributing to the local design process feedback.
5.6. Prototype limitations

DReX is a proof of a concept prototype. The following section describes its most prominent limitations.

The most important limitation is that there is no bi-directional link between Grasshopper and DReX. This means that changes in DReX impact Grasshopper but not vice-versa, and design moves in Grasshopper are not captured in the prototype as should occur in a fully functional system. The second limitation is that the use of the prototype is restricted to value changes in Grasshopper nodes; the capacity to make any structural change is not implemented in any way. Backtracking to make structural changes to the parametric model (such as replacing a division node with a multiplication node) is not possible within the scope of this prototype. The prototype is also incapable of dealing with any higher level UI interactions.

Another issue is that the way in which the prototype and the Design Narratives look at the design process is not self explanatory. The hardest idea to grasp is that any changes that are made to the parametric model appear at the end of the graph. So backtracking and making a value change to the parametric model makes nodes appear at the end of the graph. Although this is obvious if the user remembers that DReX is a history tool in which changes are always shown in the present, i.e., the end of the graph, it is counterintuitive in some cases as nodes are created at the beginning of the process. This is where the idea of time travel is helpful to understand and explain to others. Schema A (Figure 6) and B (Figure 7) also help explain this in greater detail. In contrast, making edits to an Action Node is simpler. We can surmise that for the user, the parameter and its corresponding Action Node are tightly mapped and their relationship is clear. This is not so for Backtracking Nodes.

The granularity of data is predefined in the prototype and the Tower example. Giving users the ability to select the granularity at which data is collected would benefit the identification of patterns and relationships. This would allow them to select which parameters were included and excluded from the graph. Currently users can only delete alternatives in local collections and have the option to disable automatic saving of alternatives as described in section 5.5. On the other hand, this prototype is both a
history mechanism and also a data collection and visualization system whose purpose is to gather and help users identify patterns in the data. If users are allowed to delete or exclude data from the graph, this will weaken the functionality of the tool. The compromise is to provide users with control over granularity but not the ability to delete records—these can always be restored.

The prototype made no provisions for more complex design moves such as Copy and Paste, merging files or clustering, deleting nodes or groups of nodes, parallel editing of pCAD models or other more sophisticated design moves. Also, the prototype only connected to nodes that required sliders; this meant that no Boolean toggles, numbers, or other controls found as Grasshopper nodes were transferred to DReX.

The prototype has many other limitations. The following are abstract in the sense that they are not being evaluated and serve mostly as a guide for future implementation. These include: non-collaborative tools, only local node updates, no push update on other nodes, no global update, no propagation control, fixed bitmap thumbnails instead of vector graphics, no zoom or global view of graph, no hierarchical edge bundling for backtracking edges or synchronization between processes across system.

5.7. Features to Improve DReX

We have described a system that satisfies the requirements described in the motivation section. In this next section we describe additional features that would improve the Design ReExplorer. Although these were not implemented they serve as a guide towards developing future prototypes, both lightweight and fully functional. They also serve as a guide to determine the use cases and scenarios that will give shape to the prototypes. These use cases and scenarios cover a broad range of features and tools which we believe to be applicable to parametric design as well as to other domains under real-world conditions.

Design Narratives and DReX are unfinished tools still under development. They have helped to advance our knowledge of the parametric design process and serve as a starting point for other prototypes and for the development of a conceptual framework.
which includes DReX and other similar systems. We are already able to envision future features and tools for this improved prototype and the results it may generate. Next we will present some of the features this system could include to improve its functionality.

5.7.1. Nested hierarchies

Hierarchical structures are needed to manage the complexity found in many design processes. As suggested by Holzmann (2012), abstraction and decomposition are the two strategies used to allow our model of the design process to become an interactive artefact capable of providing the right information in the correct way.

An improved prototype of our system would include a hierarchical structure that allows the constituting elements of Design Narratives —constructs/Action Nodes, Backtracking Nodes and design variations— to be grouped together to create nested hierarchies. One alternative is to replicate the structure of the parametric model itself when generating the Design Narrative graph and use this pairing as the basis for the abstraction. This entails matching the parts of the pCAD model to what is shown in the Design Narrative graph. For example, if the model has two assemblies with three parts each, then the Design Narrative graph must match this structure. This would result in graphs similar to the ones below (Figure 30 and Figure 31).
Figure 31.  

Possible nested hierarchical structure to manage complexity

The left and right columns show how three levels of nested hierarchies may be implemented in a Design Narrative graph. The right column shows how a group of constructs, Backtracking Nodes and edges can be nested. The left column shows these same elements as part of a larger graph.

Nested elements in Design Narratives are shown in the image above (Figure 30). The bars below each group of nodes indicate elements belonging to the same part with a larger assembly. In the right column, elements from two parts are nested together. The expanded elements are shown in a’). In b’) all constructs, Backtracking Nodes and variations are nested within two new nodes. These nodes retain some of the information shown in a’) and discard the rest. Further nesting is shown in c’), where a new super node contains two parts within a single node. The left column —a), b) and c)— shows the same nesting scheme together with other parts that remain expanded or un-nested. The same scheme is shown with the graph below (Figure 31).
Figure 32. A graph with three levels of nested nodes

This Design Narrative graph shows how three levels of nested nodes could be implemented. The horizontal bars below groups of nodes indicate to which part of the assembly they belong. Numbers to the right of an edge indicate how many Backtracking Nodes are nested. Numbers to the left indicate the number of parts that are nested.

Using these techniques can reduce the space needed to model and display an entire project. The same collapsed and expanded Design Narrative graphs can look significantly different, as shown when comparing i) and ii) in Figure 32 below.

Nested nodes can also convey different information. In the collapsed version of the Design Narrative graph (Figure 32 - ii) backtracking edges across parts stands out more while local backtracking within parts is hidden, shown only by a single dot, e.g. a) in ii).

Figure 33. Expanded and contracted view of the same graph

Displaying relationships between low- and high-level elements makes some relationships more salient than others.
5.7.2. **Semantically significant chunks**

Low-level actions can be grouped together to create semantically meaningful higher-level chunks or blocks of actions. By doing this, the user and system can focus on manipulating and organizing higher-level objects. This technique is used and recommended for many tools, including history tools (Gotz & Zhou, 2009) (Terry & Mynatt, 2002), and helps to avoid overloading the system with low-level user actions that have no meaning individually.

A hierarchical organization based on semantically meaningful chunks of actions would group actions together. This would change how relationships are shown in the Design Narrative graph, making the identification of the relationships between individual low-level actions and higher-level elements (parts or tasks) in the graph more salient, as shown in (Figure 32).

The ability to change the granularity of the blocks or chunks in the graph is important to any future implementation, so as to control the amount of elements in the system at any given time and to be able to do analysis across different levels of information (Low-Low, High-High and Low–High). Each permutation provides the analyst different data from which important information can be extracted. Semantic hierarchies are visually similar to nested hierarchies, as shown in Figure 32 where chunking of SolidWorks parts are shown with bars on the bottom of the graph.

There are many examples of semantic chunking in the literature. Edwards et al. (2000) use semantic chunking to group common actions together for their prototype that builds on command histories using transaction files; similar to those used in GenerativeComponents. In Timewarp semantic chunking is used for conflict resolution and granularity control of saved states (Edwards & Mynatt, 1997). Flatland, Mynatt et al. (1999) also incorporate semantic chunking in their whiteboard project. In the visual analytics domain, CZSaw (Kadivar et al., 2009) uses semantically significant actions (script-driven) to build a parametric history tool using a dependency graph which allows users to replay and edit their analyses. Similar techniques are used to generate Design Galleries (Marks et al., 1997) to differentiate between images. In Chronicle (Grossman et al., 2010), a clustering algorithm is used to delete idle video sections and undo
sequences. Semantic identification of important video frames is also used in Chronicle and SmartPlayer (Cheng et al., 2009).

5.7.3. Edges bundling

How edges are rendered is another possible improvement to the Design Narrative graph. In Figure 31, edges are bundled together to reduce occlusion and clustering and enhance overall readability. These graphically compact, rounded rectangular edges preserve start-end point detection and help reduce the crossing ambiguity of the circular edges evident in our previous diagrams. Under this scheme edges that share a common path are bundled together. The engrossing of the stoke weight is used to differentiate single edges from groups of edges. The design of these edges is based on techniques developed by Holten (2006).

This section describes an expert panel review of the DReX prototype. This study was conducted in an effort to assess the validity and viability of a future system that will be more refined and robust, and to evaluate how the core ideas presented here are received by both novice and expert designers in real-world environments.

6.1. Methods

A qualitative study was completed over a period of two weeks. Two participants were included in the pilot studies and 7 in the full study. The participants’ answers, questions and feedback were recorded as they were shown a presentation, a demo of the prototype and finally as they answered a set of open-ended questions. These recordings were analyzed and the results are presented below.

The study was divided into four parts: (1) Participants filled out a questionnaire and signed a consent form; (2) Participants were shown a PowerPoint presentation to introduce the main ideas and the prototype. The slides provided an introduction to the domain, preliminary studies, the core ideas behind the Design Analytics framework and an introduction to the prototype’s user interface; (3) Participants were then shown a demonstration of the prototype using scenarios and use cases (see Appendix D). They were then encouraged to try the prototype for themselves; and (4) They were finally asked a set of open-ended questions.

During the course of the study, participants were encouraged to ask questions about the questionnaire, the presentation and the prototype. The majority of questions were answered immediately while others were answered later with the aid of the presentation, the prototype or during the final interview session. Though open-ended questions were pre-set, follow-up questions were frequently used to further understand the participants’ responses or complement their feedback. This often resulted in an
open dialogue between the participant and researcher, expanding and deepening their analysis and feedback. Though the study was designed to run between 1 hour and 1.5 hours, the open-ended questions often resulted in it taking longer than expected, approximately 2 hours.

![Figure 34 Screenshot of the two-monitor setup used during the study](image)

A two-monitor setup was used whenever possible, given the multiple interfaces that were shown to the participants. Future users of DReX are likely to need multiple monitors to take advantage of the system. DReX is shown on the left on a large monitor while Rhino and Grasshopper are shown on the laptop screen on the right.

The prototype uses a relatively simple parametric model of a tower developed in Grasshopper. The model is used to demonstrate possible use cases and scenarios related to the prototype. The tower example transforms 2D geometry into a 3D object through an amassing of features and operations. A more complex model would impose greater computational challenges and requirements beyond the scope of this study. The tower model was created for demonstration purposes only, as mentioned previously; editing was limited to values attached to Grasshopper nodes while structural changes of the model were not available.
The development of the Tower example used in the study involves 9 stages. Each Action Node in DReX is paired to initial stage a) and to final stage i); their parametric values controlling the geometry are accessible through sliders in DReX. The Tower’s development is shown here as it appears in Rhino while DReX is being executed.

The presentation and interview process was carried out in person using a laptop running Windows 7, with half of the participants viewing the presentation and prototype on a second 23” monitor. The two screens provided a better viewing experience; other participants viewed the prototype by switching between applications, or by placing the application side by side on the same monitor.

A standalone digital audio recorder was used to record participants’ comments during the study for future analysis. Audio recordings of the sessions were made at the start of the presentation, as questions and comments were frequent during this period, and continued until the end of the study. The questionnaire, consent form, PowerPoint slides and preset questions are available in Appendix B.

The study was conducted in several locations in enclosed, private rooms at Simon Fraser University on the Surrey and Harbour Centre campus, as well as in conference rooms at design firms and private offices at the University of British Columbia.
6.2. Participants

The validity and relevance of the study depended on an appropriate selection of experts to interview. Their professional and personal experience, knowledge and knowhow would contribute to providing the best possible feedback. Towards achieving this goal a wide range of participants was sought within the CAD, computational design and AEC domain. Specifically, the idea was to find individuals with experience in real-world scenarios who dealt with complex design problems and had experience managing and teaching other designers. Additionally, we wanted feedback from computational designers and developers as well as novice participants who understood the domain even though they were not experts in the pCAD domain.

To this end, each participant was selected based on their individual characteristics. These covered multiple axes: novices and expert designers, graduate students/researchers and professionals at design studios, those in management and non-management positions, and different levels of knowledge of pCAD. Each participant to varying degrees had an academic background in one or more of the following: architecture, engineering, computer science, cognitive science and business.

Responses from the questionnaire provided confirmation of the participants’ background, level of expertise using tools and experience as designers, as well as their level of involvement in their organization.
6.3. User tasks and scenarios

During the demonstration of the DReX, a series of use cases and scenarios were presented to the participants, similar to those described in section 5.5. These were repeated upon the participant’s request. During the sessions there was special emphasis on our part to demonstrate local feedback through the local collections of alternatives within Action Nodes. The same treatment was given to global feedback which is based on the nodes and edges within Design Narratives. Use cases were presented to introduce specific system features such as panning across the timeline or changing a slider value. Scenarios were used to emphasize or complement higher level concepts related to the pCAD modeling process, design in general and relevant scenarios that would help the participant connect with the prototype.

6.4. Results & Conclusions

In this section we discuss the results generated from the expert panel review. Participants’ feedback provided us with invaluable insights into the prototype. They also provided feedback on future directions for more refined and robust systems, such as
new and interesting features, original uses of the prototype, technological bottlenecks and expectations from designers which we are certain to encounter in the future.

The feedback was overwhelmingly positive but not without a measure of constructive criticism and specific feedback about several aspects, including the user experience, feasibility within existing workflows, complexity of real-world environments, synchronous and asynchronous collaboration, interplay with clients, user interface issues, capturing complex user behaviours and other system improvements. As a proof of concept, the prototype passed the scrutiny of the experts and in some cases even exceeded their expectations and surprised them. Nevertheless they all had suggestions to make DReX better and made it clear that a fully developed system would require further development.

The majority of the feedback was focused on the prototype. Feedback related to the Design Analytics framework was significantly less explicit. The framework’s main ideas—feedback, analysis and re-exploration—appear to have made a seamless transition into the prototype and to have been accepted by the participant as such. This transition was helped by the PowerPoint presentation which introduced and tied together the ideas and the prototype before the participants saw the prototype in action.

In this section we look more closely at our experts’ feedback and provide a collection of key insights as well as features, systems and user interaction improvements which need to be addressed by future prototype iterations, or when re-implementing prototypes based on our framework. Finally, we discuss future work that can be derived from these findings.

6.4.1. Results

The open-ended questions provided ample raw data. In many cases participants’ answers—those relevant to this study—were spread across multiple questions. In analyzing the interviews we grouped together answers that related to the same theme even if spread across the interview. As expected, many comments focused on issues that can be easily solved or that provided no insight into the future of the tool or framework and, therefore, were omitted. As expected, answers by experts were
consistent, with the occasional outlier; personality, background and experience likely shifted answers one way or the other.

Participants’ response was enthusiastic about the prototype and the ideas behind it. When asked about integrating a future prototype into their workflows answers were positive. There were several reasons for this. The first is that DReX automatically saves a record of the design process. The loss of an alternative, a common occurrence for designers, is a shows how DReX would be useful.

“(...) it’s just not that easy because you didn’t save it at the right point. Or you modified something before you realized that you wanted to go back to that, and then you can’t get back to it again, it’s that elusive point that was perfect and you never get back to it” (P4)

The second is that the prototype transforms the design process from an abstract idea into an artefact that is displayed onscreen.

“(...) I think, it’s really interesting because it (DReX) just opens up everything you worked through in a visual way. And you immediately see the (...) progression of your design and (...) how you are affecting it.” (P3)

“You know, it’s a black box in a way and this (DReX) is opening the black box, or trying to.” (P3)

For participants, this meant that known or unknown processes suddenly had a visual representation on the screen that they could suddenly see for the first time. This was a very rewarding experience to witness in particular with experienced participants that hold the design process in high regard.

On a similar note, participants responded positively to the local collections of alternatives which run across the interface. Two features grabbed their attention. Firstly, participants liked having all alternatives together, per Action Node, and to be able to compare them within collections as well as with previous and following collections. They got a sense of how the pCAD model developed across time irrespective of how it was constructed in the base system. This of course was coincidental because of the
way the Tower example was setup, nevertheless the reformatting of Action Nodes and the structure of the graph is conceivable.

The second feature was that two simultaneous design process representations were shown inside DReX. Since alternatives are automatically saved and placed in a repository (local collections), irrespective of when they were created, designers can view their design process in terms of depth (left to right along the Narrative graph) and in terms of breadth (as alternatives saved into the local collections). It appears that even though Design Narratives have implicit branches, when the “fruit” is made visible to the designer the Design Narrative model becomes a lush and exuberant tree, i.e., designers are particularly interested in alternatives when they recorded and displayed effectively.

Regarding the design process, two observations were made that we feel are important to share. One is the importance of design process feedback to the teaching of design. While this idea is supported by the literature, it was refreshing to hear it directly from participants involved in the teaching of design at a graduate level. The following is a transcript from participant 2 when asked about the value of the design process for them and their practice:

“...we’re always very concerned with process with our students (...) and so I’m always encouraging them to document their process because our reviews are (...) much about process, (because) it’s more important (...) that they (...) develop a strong architectural process than it is for their end result (...) being like perfect. (...) so what we’re mostly interested in is how they make decisions, (more) than how they design.” (P2)

The other observation by a participant related to a more profound change in the design paradigm. This participant mentioned how future prototypes could change how we design by changing how we look at each decision we make. The difference is that DReX provides a clearer description of the design intent, even if the design solution is imperfect. This would release the designer from the pressure of executing a perfectly defined solution in order to focus on the more important work of actually solving the problem. Refinement comes next, possibly by the same person or by others.
Another comment was that DReX allowed participants to see a “complete” visual history of the design process in the Design Narrative graph, in particular the thumbnails in the local collections which showed how the design progressed and the changes as they were made. By saving alternatives, participants mentioned how it would become very easy to go back and reuse or build upon past alternatives which in most cases are lost during the design process. Some participants mentioned that future prototypes would likely encourage designers to review their design process and to be more introspective about that process.

Participants were concerned about a future prototypes’ intrusion into the workplace and the implications it might have on workflows and social dynamics. These were expressed in terms of competition between designers, designers being so self-aware of the processes that their ability to perform would be reduced and overreaching and overbearing managers using this new tool to micromanage designers. Another more benevolent but nevertheless important issue mentioned by participants was procrastination; in particular how one might get lost looking at design process graphs.

“I just may be a bit more confusing then, it’s like design is already confusing process.” (P3)

“(...) so you are suddenly in front of things that you might not, you know, be willing to see. It might get more confusing but it’s very interesting.” (P3)

Participants were asked to assess their willingness to accept automatically generated feedback based on their design process. This was asked with the current prototype and as a follow-up question with a future prototype in mind which would have pattern recognition capabilities based on an artificial intelligence system. While participants were open to having a computer provide feedback they also always expressed some hesitation. The main concern was intrusive feedback that would interrupt or annoy users. User control over the time and magnitude of automatic feedback became a common request for future versions. When asked about low- and high-level feedback, participants were more enthusiastic about the possibility of having higher-level feedback than lower-level. This became more evident when paired to a system with an AI component or some pattern recognition algorithms.
Opinions were mixed about the Design Narratives, in particular regarding the backtracking edges between nodes. The main concern was the visual aspect, scale and overall usefulness of the Design Narratives as feedback. Even so, all participants saw the importance and significance of Design Narratives when reviewing the better examples in the presentation slides, which were at a better scale and proportion. To resolve these issues participants suggested several solutions: conventional zoom, semantic zoom and hierarchical structures, or simply turning them off when not in use. While these features would improve the functionality of any future prototype, the real issue remains unresolved—limited screen availability. A significant effort would be required to solve this issue using a mixture of Visual Analytics, graphic design and interaction techniques to reconcile the importance of backtracking edges and its usability.

Higher-level usability issues did arise. Propagation control was the main one expressed by participants. The current version of the prototype only updates the thumbnail of the parameter that is being edited. Grasshopper updates all parameters while running in the background (as shown in Figure 34), but these are not loaded by DReX. This was a cause of confusion when demonstrating the prototype, as participants expected all downstream thumbnails to update. Limiting propagation was a design decision made early in the prototype’s development to reduce the computational load and trim down the implementation load. Our thinking was that the core ideas we wanted to test with the prototype would not be affected by this decision. The solution, shared and liked by participants, was to include propagation and update controls that would allow designers to have greater control over the impact a parametric change would have on DReX.

Another issue mentioned was complexity and the computational load that real-world projects would impose on a future prototype. To mitigate these issues hierarchical structures would be added to increase the abstraction gradient, thereby reducing the amount of low-level objects. This however does not address the main problem that became apparent even with the simple Tower example. Once designers are given the ability to generate alternatives at their leisure, they do so. If this happens with normal pCAD systems, then future implementations will be challenging for developers and
computers with a tool like DReX, which facilitates the creation and management of alternatives.

Another issue requiring deeper analysis is how to manage design moves that are more semantically complex than those currently implemented, e.g. copy/paste or merging files within the prototype. Although only one participant asked about this issue, it is clearly a problem needing to be resolved in the next iteration. Similarly, the prototype is unable to manage structural changes made to the parametric model. In principle, this only requires greater connectivity between DReX and the base pCAD system. With this increase, Actions Nodes in DReX could shift between semantically equivalent nodes as guided by the designer. A graph representation would also be developed.

Interesting and useful UI and interaction suggestions were made, including: the addition of secondary notation (Green & Petre, 1996) to allow designers to annotate Action Nodes, alternatives and edges; adding split screen capabilities to visualize multiple graphs or different sections of the same graph simultaneously; making the UI a responsive design, particularly the timeline and local collections, similar to today's websites that automatically change depending on the interface characteristics (laptop, phone or tablet); having the system automatically generate alternatives based on upper and lower bounds and set sizes; and including tools for collaboration, such as multiple graphs to monitor and interact with others.

Another suggestion was the reordering of the Design Narrative graph within the prototype based on the true chronological order in which actions took place, irrespective of backtracking. That would move the latest backtracked parameter node to the “present,” i.e. to the rightmost position on the timeline. This alternate interpretation of the graph would provide a new perspective of the Action Nodes, backtracking and alternatives. Another alternative, not mentioned by participants, is to have the graph show the true time lapse between actions. This would provide additional information to be considered in an analysis of a design process.

Based on this prototype, we have identified several key issues that need to be resolved for this tool to be effective. We have also been reassured that research in this
area is both promising and relevant to practitioners and researchers alike. Our future work is well delineated by the prototype's shortcomings and strong points. It is also empowered by the confirmation that designers, and experts in particular, hold the design process and the search for alternatives paramount and that these experts need tools to support them.
7. Conclusions and future work

The research and development of Design Narratives, DReX and running the expert panel review provided insights into something that could potentially be interesting. An exploration of these insights and conclusions of our work follows.

Many alternatives were explored during the design phase of each of these artefacts. Design Narratives and the prototype went through multiple iterations before arriving at their current state. All these alternatives created a space of possibility. From this space, common elements were identified and used to sketch the core concepts that in our view are common to all solutions.

While still preliminary, this research may over time provide the foundation for a conceptual framework, the Design Analytics framework. This will position systems such as DReX within a delimited conceptual context, providing meaning and direction. We acknowledge that without further research this first approximation may fail to coalesce or could become a fringe component of another framework. We are continuing to develop a conceptual framework based on the simplest definition, building it around a set of concepts that fit together to describe a much simpler and concrete construct.

We believe there is a void, an opportunity to develop tools similar to DReX that take advantage of increased computational power and improved pCAD tools. There is also an opportunity to learn about the design process, and design in general, with the help of these tools. This is a family of tools which can help multiple actors within the design domain, either to expand our knowledge or support design.

Existing frameworks may share elements with a new framework to be developed. Based on what we have learned during this research, the Design Analytics framework would intertwine with other areas of knowledge and fall in the intersection between Decision Support Systems (DSS) (Burstein & Holsapple, 2008) and Creativity Support Tools (Shneiderman, 2002).
From the perspective of this thesis, the Design Analytics framework is an abstraction of the DReX prototype. This conceptual framework bridges the gap that divides the acts of designing, reflecting on design and the design process. The Design Analytics framework is not a prescriptive or descriptive framework for design, or of design. Rather, it is a guide for a new set of tools intended to systematically allow designers and other stakeholders to use design process information in order to understand and improve the tools designers’ use, the design processes they traverse and, ultimately, the design solutions. Its purpose is to establish a foundation on which systems can be developed, one which is specific enough to guide designers and developers while also abstract enough to not be limiting in their conception or implementation.

The framework emerged from a collection of ideas developed while completing the first of our research studies and subsequent papers (Erhan et al., 2012) (Shireen et al., 2011) and represents an amalgamation of ideas found in the literature, including (Woodbury & Burrow, 2006) (Makkuni, 1987) (Derthick & Roth, 2001) and (Akers et al., 2009).

The framework borrows from the cycle used for DReX. It is as triad of concepts working in conjunction, where re-exploration enables design space re-exploration based on analytic reasoning, analysis enables and supports reflection and analytic reasoning and feedback provides design process-centric feedback to designers and other stakeholders. These three broad interrelated concepts work together with the designer to create a feedback loop (Figure 36).
Two ideas by Makkuni (1987) stand out in helping to consolidate some of the initial ideas presented here. The first is highlighting the path that is covered and left behind, as well as the destination. This is best exemplified by thinking of the tracing paper that is left behind or discarded in architectural studios or by looking at a sketching journal as a means to understand the design process, as previously described in the motivation and background sections (section 1.1 and 2).

As architectural projects develop, the information and ideas etched onto the tracing paper tell the story of how spaces were conceptualized, defined, refined and eventually built. They also tell the story of what would never be built, i.e., the terrible ideas, and the better ideas that were never seen, as well as the better solutions that were discarded on a whim or because of unforeseen factors or issues beyond our control. The role of the tracing paper becomes that of storyteller whose subject is the design outcome.

The second idea is that the design process can be considered an artefact in itself, or more specifically, an “artefact for design” (Makkuni, 1987). McCrickard (2010) explains that artefacts in design are “...representations that express properties or captured information – can serve to inspire, represent, and manage the decisions made throughout the design process.” The design process as an artefact for design encapsulates the accumulative actions that take place from the beginning to the end of a
given design project. It becomes a purveyor of information about when, how and why problems and solutions occurred. Additionally, the artefact can be edited, saved, shared and replicated. Lastly, this “artefact for design” becomes a platform from which new design decisions can be explored, akin to reusing a discarded sheet of tracing paper to explore a new solution by sketching a new alternative over it based on what was learned previously. In doing so the design process, as an artefact, becomes another tool which designers can use to design (Makkuni, 1987).

Similar ideas, but from a more abstract perspective, are expressed by Woodbury and Burrow (2006):

...the best accounts of intent will necessarily include the decision history of design process, then narrative becomes an important part of a design system. The bet here is that telling stories of design decisions, including the decisions foregone, begets new understanding. Efficient navigation and recombination reinvigorate such narrative. To explain the telos of a design, simply replay an account of its creation. If the navigation mechanism embodies structure other than derivation, for example, plausible explanations other than those followed by the designer might be available and such explanations could just be interesting. Navigation and recombination reify story telling as a design tool.

A comprehensive evaluation of a design solution is not complete without an analysis of the design decisions and alternate paths the designer might have taken at every stage of a design process. As an artefact, the design process becomes a powerful tool worth implementing. It is not until we consider these alternative choices do we get a deeper understanding of the true value of a design outcome, be it good or bad. During each step of a design process the designer is presented with three alternate paths from which to choose, each of which falls within one of the following categories: paths that are selected; paths that are explored and then discarded; and paths that have never been explored. Once a path is selected the design moves one step closer towards completion. These decisions are single pieces of data which can be gathered and used to enhance our understanding of designer actions.
Tools developed under the Design Analytics framework can help evaluate a design process and outcome. To do so, at least two questions need to be considered. The first is what designers do, which can be evaluated by focusing on their design solutions and the actions they take while designing, i.e., by looking at the process through the accumulated sequence of actions that constitute the design process. The second question is understanding the why, the reasoning behind a designer’s decision. Though the reasons behind a designer’s decisions may elude us, creating the framework may allow us or others to establish a new foundation with which we may be able to build upon previous research to identify the why.

At this preliminary stage, thinking of the Design Analytics framework as analogous to a complex non-linear open system helps to provide order. The diagram below exemplifies the relations among the different elements that would constitute the framework (Figure 37). Part of the complexity of the framework is that its elements belong to different levels of abstraction, and so do the relationships between them. Non-linear systems have attractors to unite elements together, a term borrowed from complex systems to describe an entity around which other elements can congeal. The act of designing—the design process—is the analogous attractor in this system.
We have identified some of the elements that would be part of the framework. The first three, Feedback, Analysis and ReExploration, are borrowed from DReX to become higher level concepts within the framework. Feedback, Analysis and ReExploration together with design are fundamental components of the framework but as an open system that is still relatively undefined, other elements can be incorporated or even disappear. Other concepts that we believe to be important are: big data, collaboration, augmented history tools, visual analytics, multidisciplinary, event data collection, machine learning, cognition, time, parametric CAD, statistical analysis, hierarchical structures and research. Systems developed under the framework would be designed based on the relative weight given to each of these concepts. Each system designed to match the stakeholder’s requirements.

A more thorough and deeper treatment of these themes is necessary to properly define the framework but that would extend beyond the scope of this research; nevertheless we believe this first approximation will help form what could be a more formal framework.
References


Dassault Systèmes SolidWorks Corp. (2013). SolidWorks [software].


Appendix A.

Idioms of use

List of user defined strategies - idioms of use - to manage a project’s history

**Table A.1. Idioms of use table**

<table>
<thead>
<tr>
<th>System</th>
<th>Strategies to manage alternatives and work</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoCAD</td>
<td>Saving of multiple files, having multiple instances of design states within same file (side by side), using multiple layers to manage temporal versions of project.</td>
</tr>
<tr>
<td>SolidWorks</td>
<td>Files, configurations, design tables</td>
</tr>
<tr>
<td>Photoshop and Illustrator</td>
<td>files, layers, side by side comparison of solutions</td>
</tr>
<tr>
<td>Word</td>
<td>files, side by side</td>
</tr>
<tr>
<td>Grasshopper</td>
<td>Files, side by side, (states), recording of local parameter values</td>
</tr>
<tr>
<td>GC</td>
<td>Transaction files provide a partial view of the design process full of garbage and noise with no integration into the workflow</td>
</tr>
<tr>
<td>AutoDesk Chronicle</td>
<td>Provides an enhanced video recording of design processes for teaching and learning.</td>
</tr>
</tbody>
</table>
Appendix B.

Study Materials

This appendix contains the study materials used in the evaluation of the Design ReExplorer (DReX) as described in section 5.
### Page 1 of pre-questionnaire used during DReX expert panel review

<table>
<thead>
<tr>
<th>Participant No.:</th>
<th>Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Rank your overall level of expertise as a designer

*Expertise as a designer: N/A, Novice, Expert*  
*Years of experience:*  

2. Rank your level of expertise in the following design domains

<table>
<thead>
<tr>
<th>Domain</th>
<th>N/A</th>
<th>Novice</th>
<th>Expert</th>
<th>Years of experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphic Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Industrial Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interior Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Landscape Architecture</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Urban Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interaction Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UI Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Systems Architecture</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Engineer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Rank your level of agreement or disagreement with the following statements

<table>
<thead>
<tr>
<th>Statement</th>
<th>Strongly Agree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 I make conscious and deliberate design decisions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2 I value how I achieved a solution as much as the solution itself</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3 I rely more on a rational than an inspirational design process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4 I value feedback from others on how I develop a design solution</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. Rank your level of influence over the design workflows within your organization

*Influence: Not, Partial, Complete*  

5. Have you ever managed a team of designers?  
6. Have you ever managed multiple teams in different project simultaneously?  
7. Have you ever led a design firm?
8. Rank your level of expertise in CAD systems and years of experience

<table>
<thead>
<tr>
<th>CAD System</th>
<th>NA</th>
<th>Novice</th>
<th>Expert</th>
<th>Years of Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoCAD</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Rhino</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>MicroStation</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>ArchiCAD</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>form-Z</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>SketchUp</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>VectorWorks</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>

9. Rank your level of expertise in Parametric CAD systems and years of experience

<table>
<thead>
<tr>
<th>Parametric CAD System</th>
<th>NA</th>
<th>Novice</th>
<th>Expert</th>
<th>Years of Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenerativeComponents</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Revit</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>SolidWorks</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>CREO Elements/ProEngineer</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>CATIA</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Rhino - Grasshopper</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Inventor</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Design Script</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>

10. Rank your level of expertise in 3D modeling software

<table>
<thead>
<tr>
<th>3D Modeling Software</th>
<th>NA</th>
<th>Novice</th>
<th>Expert</th>
<th>Years of Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maya</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>3D Max</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Cinema 4D</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>

11. Rank your level of expertise in other design software

<table>
<thead>
<tr>
<th>Design Software</th>
<th>NA</th>
<th>Novice</th>
<th>Expert</th>
<th>Years of Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhotoShop</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Illustrator</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Coreldraw</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Indesign</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>AfterEffects</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>InkScape</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>
Questions used during the interview process

Q&A

1. Do you see this prototype - and the ideas behind it - affecting your design practice, and why?

2. What are the potential - Pros and Cons - of a fully functional system built around these ideas being integrated into your daily design workflow?

   Collaboration / Management / Learning / Tutorials / Design Patterns

3. Regarding the design process. Do you see any value in viewing & understanding the design process, your own or that of others, and how that is relevant to your design practice?

   Suggest Alternatives / Anticipate Errors / AI / Heuristics

4. Beyond the design studio, do you see this prototype being used somewhere else?

   Research / Teaching / Other Domains - Business

5. What prototype features are of most and least interest to you?

6. Does it provide feedback, analysis and the capacity to re-explore?

7. Do you have any suggestions to improve the
   a) The conceptual framework
   b) The prototype (beyond fixing the UI and bugs)

8. Do you have any ideas for future implementations
**Consent formed signed by participants before starting study**

---

**Informed Consent Form**  
Interactive Design Process Narratives

**Principal Investigator:** Dr. Hail Erhan, Assistant Professor, SIAT, SFU, [successor information]

**Collaborator:** Dr. Robert Woodbury, Professor, SIAT, SFU, [successor information]

**Research Assistants:** Nahomi Shireen, Rodolfo Sanchez, Jelena Popovic, and Sinisa Kolaric, [successor information], SIAT, SFU, [successor information]

**Goals of Study:** This study is designed to investigate interactive parametric design systems for supporting design space exploration. Our goal is to improve existing interfaces and propose novel ones to support design generation and selection of alternative solutions. We are interested in understanding the potential of a proposed framework and a prototype that may guide us to develop better user interfaces for these purposes.

**Study Procedures:** You will be asked to view a presentation and prototype for 40 min. This will be followed by a 20 minute interview in which you will be asked about the concepts and prototype. An audio recording of the session will be made.

**Risks of Study:** There are no foreseeable physical or psychological risks for participating in this study. Nor are there no risks associated with the distribution of data. However, you may withdraw from the study at any time during and after the study by notifying one of the investigators.

**Confidentiality:** All personal and identifying data will be kept confidential and only the investigators of this study will have access to the data. All records from the study will be kept secure either in paper or electronic format.

**Contact for information about the study:** If you have any questions or desire future information with respect to the study (e.g. research results), you may contact Dr. Erhan at [successor information]

**Contact for concerns or complains about the study:** If you have any concerns or complains about your treatment or rights as a research subject, you may contact the Hal Weinberg, Director, Office of Research Ethics, [successor information]. You may reference the application number of this study [2011WXXXX]

**Consent:**  
Your participation in this study is entirely voluntary and you may refuse to participate or withdraw from the study at any time. Refusal to participate or withdrawal after agreeing to participate will have no adverse effects on your grades or evaluation in the course or classroom. Your signature below indicates that you have received a copy of this form for your own records and that you consent to participate in this study.

---

**Participant Signature and Date**

---

**Printed Name of the Participant**

---

**Signature of Investigator or Research Assistant**

---
Appendix C.

Study code

This appendix contains Grasshopper code to generate the Tower example and the corresponding 3D model as shown in Rhino during the evaluation of the Design ReExplorer (DReX) as described in section 6.

Grasshopper Code

3D CAD model of Tower in Rhino