6 CONCLUSIONS

The multi-criteria design and optimization of buildings poses both an interesting technical challenge and potentially a powerful means by which to drive design toward better performance.

This paper shows a conceptual approach to supporting optimization within a single computational design system that crosses traditional discipline boundaries and can therefore address issues of optimization that are inherently multidisciplinary.

While the idea of design optimization has been discussed in the research literature, it has not been widely used in practice, mainly because tools based on optimization have not been widely available or indeed available in forms that are easily accessible to practitioners.

Therefore, one important future challenge for software developers is to make optimization tools more accessible and more easily used by practitioners, but without compromising the rigor of use that is required to achieve valid results. The anticipated increase in adoption of optimization tools has the potential to bring substantial benefits not only to architects and building engineers but also to the users and owners of buildings, and thereby address wider economic and sustainability concerns.

REFERENCES

Aish, R. (2011). DesignScript: Origins, Explanation, Illustration. Design Modeling Symposium, University of the Arts, Berlin.

Attar, R., R. Aish, J. Stam, et al. (2009). Physics-Based Generative Design. CAAD Futures.

Commission Internationale de l'Eclairage. (1994). Guide to Recommended Practice of Daylight Measurement, ed. J. D. Kendrick. Vienna: Commission Internationale de l'Eclairage.

Evins, R., S. Joyce, et al. (2012). Multi-Objective Optimisation: Getting More for Less by Design. Proceedings of the Institution of Civil Engineers, Civil Engineering Special Issue 165.

Shrubshall, C., and A. Fisher. (2011). The Practical Application of Structural Optimisation in the Design of the Louvre Abu Dhabi. In Proceedings of the International Association for Shell and Spatial Structures Symposium, London.

Whitehead, H., and B. Peters. (2008). Form and Complexity. In Space Craft: Developments in Architectural Computing, ed. D. Littlefied. RIBA Enterprises.

Williams, C. (2001). The Analytic and Numerical Definition of the Geometry of the British Museum Great Court Roof. In Mathematics and Design, eds. M. Burry, S. Datta, A. Dawson, and A. J. Rollo, 434–40. Geelong, Victoria, Australia: Deakin University.

PARALLEL DEVELOPMENT OF PARAMETRIC DESIGN MODELS USING SUBJUNCTIVE DEPENDENCY GRAPHS

ABSTRACT

Exploring problems through multiple alternatives is a key aspect of design. In this paper, we present a prototype system as an extension to existing parametric CAD tools that enables parallel generation and editing of design alternatives. The system is built on two fundamental ideas. First, use of subjunctive dependency graphs enables simultaneous work on multiple design variations. These graphs capture and reveal complex data flow across alternative parametric CAD models. Second, prototype-based modeling provides a weak notion of inheritance, enabling incremental description of differences between alternatives. The system is intended to be general enough to be used in different CAD platforms and other systems using graph-based modeling. The three basic system functions are definition of alternatives (variations) using prototype-based modeling, structural and parametric divergences of the prototypes, and interactive comparison. The goal of this research is consistent with the general qualities expected from any creativity support tools: enabling exploration and simultaneous development of variations.

Naghmi Shireen
School of Interactive Arts and Technology
Simon Fraser University

Halil Erhan
School of Interactive Arts and Technology
Simon Fraser University

David Botta
School of Interactive Arts and Technology
Simon Fraser University

Robert Woodbury
School of Interactive Arts and Technology
Simon Fraser University

1 INTRODUCTION

Exploring alternatives is a crucial part of design (Akin and Lin 1995; Cross and Dorst 1998). Whatever the media used, designers generate, compare, analyze, and reflect on multiple solutions in their search for a viable, satisfying design (Simon 1973). Computer Aided Design (CAD) systems should be expected to support these activities (Shneiderman 2007). However, because they have limitations on multiple states, their functionality falls short in helping designers. CAD largely supports building single-state models—term introduced by Terry and Mynatt (2002)—with limited support for creative processes, especially those that explore alternative solutions.

Recent parametric-CAD (P-CAD) systems present opportunities to explore alternatives. They enable rapid change of design dimensions and structure (Woodbury 2010). Using them, designers can change the values of parameters in an abstract structure to explore various resultant forms. However, P-CAD systems presently show the result of only one parametric state at a time; alternatives are lost during work (Shireen et al. 2011). Herein is an opportunity to enhance P-CAD systems, so the designer can compare and build multiple concurrent alternatives.

P-CAD typically supports multiple modes of model construction, including scripting, and facilitates this with interactive graphical representations of the data flow, called *dependency graphs* (Aish and Woodbury 2005). Textual scripts for generating form can be hard to see and understand, so interaction with the dependency graph provides a way to grasp the relationships among the design elements. Here we focus on dependency graphs, as these expose the logical structure of a design, visually decoupled from the geometric views (where parametric relationships are not always accessible). The decoupling permits the designer to work with logical structure free from the visual noise of the resultant geometry, while still directly seeing dependency relationships.

This study is part of a research program on representing and interacting with alternatives in design. The larger goals are about modeling simultaneous independent scenarios, comparing them, and manipulating them independently or collectively (Terry et al. 2004). Lunzer and Hornbæk (2008) call such interfaces *subjunctive*, which term we use here. Our approach draws upon the *prototype* paradigm of programming, which enables the modification and evolution of individual objects, without *a priori* classification into type hierarchies (Taivalsaari 1997). Briefly, a subjunctive graph is a derived graph from another prototype dependency graph, to execute "what if" scenarios. A subjunctive graph inherits all properties (nodes and dependencies) from a prototype graph. It can override these properties, add new properties, define parametrically or structurally different solutions, and can compose properties from several prototypes. An equivalent view is that subjunctive graphs contain *subjunctive nodes* that distinguish those parts of the graph that differ. A computation path through a subjunctive graph must pick one differing part from each subjunctive node.

After describing P-CAD systems more fully, we use a realistic scenario, based on Foster and Partners' Copenhagen New Elephant House, to illustrate strategies for using dependency graphs to support exploration of alternatives.

2 DESIGN AND PARAMETRIC CAD SYSTEMS

In current practice, both experts and novices use P-CAD systems for experimental and professional purposes. The widespread adoption of P-CAD systems attests to their utility in at least some aspects of design. Yet these systems challenge both designers and developers by introducing new tasks that entail new mental models, new representations, new knowledge, new skills, and new strategies for building models (Aish and Woodbury 2005; Kasik, Buxton, and Ferguson 2005). While they enhance modeling capabilities, their pattern of use in actual design remains unclear. In the wild, they might reasonably be expected to enable exploratory processes, facilitate collaboration, support *rich history-keeping*, and be useful for both novices and experts (Shneiderman 2007). In addition, they ought to be engaging, balance effort-reward tradeoffs, provide transparency toward achieving tasks, and be expressive. Their actual success against these aspirations is limited to specific and often technically involved strategies (Woodbury 2010, Chapter 3).

In parametric design, the main view is of the constructed 3D geometry—a familiar representation for most designers. Recently, graph and script views have become central parts of the interface. Figure 1 shows that such views enable definition of parametric dependencies among model elements at a symbolic level (Woodbury 2010). When dependency graphs were first introduced into P-CAD systems, their main role was to visualize parametric dependencies, but they had limited selection capabilities. In current systems, they enable direct model creation using semantically meaningful nodes and links. In addition to the nodes representing geometric elements, other nodes can represent arbitrary data useful in design. Some node types provide access to data external to the system, thus supporting weak but easily accessible integration across tools.

Parametric models are dynamic—they change with their inputs. A well-known pattern of use is to build geometries and parameters that are not part of a design and act specifically as input-output controls. In use, models and graphs grow in both size and complexity, so model elements can be composed together into reusable components with interfaces and input and output parameters—a form of encapsulation, composition, and abstraction.

P-CAD systems inherit a single-state limitation from the conventional CAD tools on which they are largely based. This approach limits exploration and comparison of variations. Although there are studies on exploring variations using representations resembling the intended result (Hartmann 2008; Krish 2011; Marks et al. 1997; Terry 2004), to the best of our knowledge, there is little research on how symbolic representations as interactive graphs can be used for this purpose. This research explores opportunities in graph-based parametric modeling for parallel editing of multiple design variations.

3 DESIGN STRATEGIES AND SYSTEM NEEDS

Design builds on previously generated solutions and often diverges from existing solutions to further develop alternative ones. Designers adapt different strategies in the search process (Akin and Lin 1995). For example, they copy, paste, and modify to create alternatives. Our approach envisions tool support for design strategies, where the initial solutions are taken as the base alternatives and are used to derive subsequent alternatives. Essentially, each alternative is a variation of a base alternative and is accessible at the same time, and editable in parallel with the base and the other alternatives. Below we summarize some of these strategies and the need to provide support for them through a scenario of designing a stadium—a design problem that has been deeply probed through parametric models (Shepherd, Hudson, and Hines 2011).

3.1 Branching Variations

Designers will typically branch from an initial design to try different ideas. For example, combinations of seating, circulation, and form might be explored, driven by sight lines and seating capacity in a stadium design. In parametric systems, branching is achieved by simple parametric variation, by altering dependency relationships, or by substituting, adding, or removing features. While variations are core to parametric systems, designers must explicitly and manually manage them, one at a time. In the stadium-seating example, to see the effect of the number of seats in all variations, the designer must work on each solution independently. Adding a shared feature to all variations is a challenging task involving concepts akin to those from software engineering. Design exploration by branching demands intense manual redoing and reshaping of the models.

3.2 Merging Variations

Often, designers will combine the best parts from different branches into a single model. The stadium design might use the seat layout from one variation and the roof structure from another. In current systems, merging partial solutions usually requires building a new model. The merged parts are thereby divorced from their original contexts, cutting off the possibility of further development of the merged parts by tweaking their original contexts. In current systems, after a merge, the model requires manual patching.

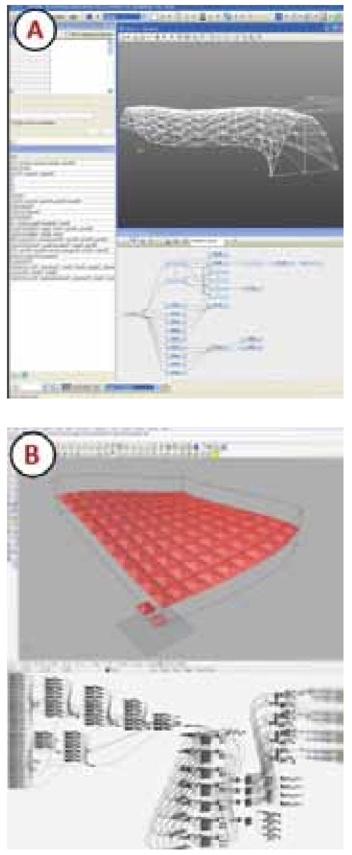


figure 1

figure 1

A curved surface structure in (a) GenerativeComponents™ and (b) Rhino Grasshopper®.

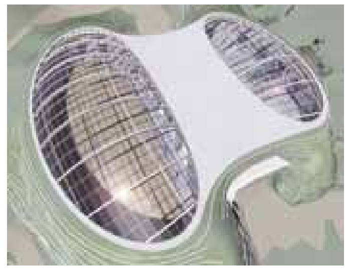


figure 2

3.3 Reusing Parts

Design often involves reusing parts or all of previously created solutions. In parametric modeling, reusing of parts not only substantially decreases the amount of time to model each variation, but also reduces chances of error as modifications are made in one place. If the reused part in different variations can be kept in the same workspace, the effect of updates on parts can be observed. In the stadium example, various seat furniture designs can be compared in each of the alternatives. The object compilation features common in most systems provide for reuse akin to functions in programming languages. That is, the ability of an object to be used in a new context is expressed through a list of arguments.

3.4 Simultaneous Manipulation

Our stadium designer might want to see the effect of the same number of seats across all stadium variations—that is, to simultaneously edit multiple variations by changing linked parameters and features. Yet, in current parametric systems, our designer must work on each variation independently, even if they share common parameters. A typical strategy in existing systems is to write a script that enumerates variations, one at a time.

4 THE PROTOTYPE: WORKING WITH SUBJUNCTIVE DEPENDENCY GRAPHS

How might a subjunctive dependency graph look and act? As mentioned, a subjunctive interface should enable setting up simultaneous independent scenarios; comparing them; and manipulating them independently or collectively. Here, we run our initial prototype through a realistic scenario and describe how it supports branching, merging, reuse, and simultaneous manipulation.

4.1 Design Scenario: Elephant House

Our scenario uses the design of Foster and Partners' Elephant House in Copenhagen (Figure 2). This project was parametrically modeled in many aspects (Woodbury 2010, Chapter 5; Peter 2008) of which we focus on the toroidal roof forms, cut by planes from two geometrically related tori. In Fosters' design process, the tori and the cutting plane could be smoothly changed to explore variations of the shapes and the space defined, while the building connecting them could be replaced by radically different geometries (Figure 3). In our scenario, we imagine three alternatives of viewing the elephant enclosures (Table 1).

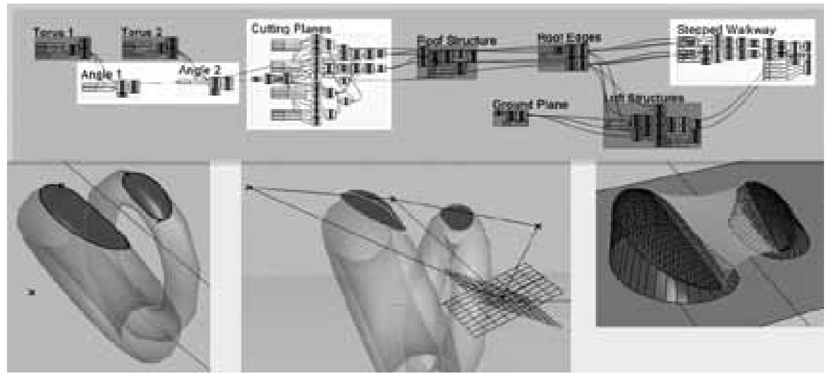


figure 3

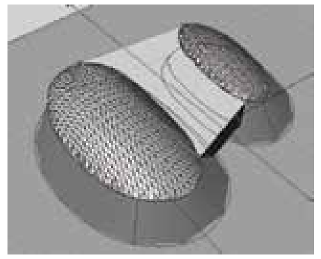
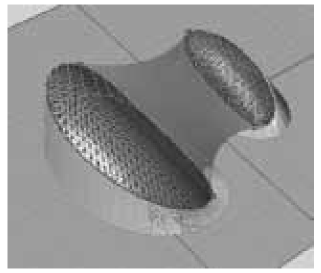
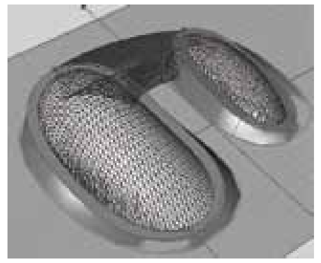
figure 2

A perspective view of Foster and Partners' Elephant House design. © Brady Peters.

figure 3

Two tori, each cut by a plane, define the large skylights for the elephants in the Elephant House. The space between provides the main accommodation for visitors.

Table 1: Three design scenarios: Picnic, Stroll, and Maze

1. In "**Picnic** with friends", the space between the enclosures comprises an elevated, open area for gatherings, with a protected area underneath;	
2. "**Stroll** up a grassy hill to discover a view" provides a stepped walkway between the enclosures that reveals the interesting roof shapes, and gently angles upward from ground level to overlook the field.	
3. "Children dash around a **maze**" comprises a gallery that overlooks both the field and a central, connecting courtyard.	

4.2 Interacting with Parallel Designs

The design of dependency graphs in our prototype is consistent with the node-link diagram design in GenerativeComponents™ and Rhino Grasshopper®, which are representative of propagation-based parametric modeling software, mainly used for designing built environments. In our prototype, the major focus is on developing a set of techniques for parallel development of alternatives; hence we intentionally left the graph layout to another study. However, for simplicity, the prototype organizes the nodes by propagation order and in nearly linear form. Where large blocks of related code are identical across alternatives, we represent these as single nodes (all systems provide tools for such node compilation).

Below we describe the prototype interfaces and how they enable parallel editing of graph-based parametric models of the three Elephant House alternatives. We demonstrate how base designs are used to create variations by applying the design strategies discussed before (branching, merging, reusing, simultaneous manipulation).

4.3 Branching Variations

Branching from a prototype design requires creating a new panel for a new subjunctive graph (equivalently for graph differences). When the user instantiates a new panel, an empty design model is created. This adds an empty geometric model view and an empty graph area in the system UI (Figure 4). When the user selects a node from the prototype design and inserts it in the new subjunctive graph, a new instance of the prototype design is created and becomes ready for further editing (see next subsection).

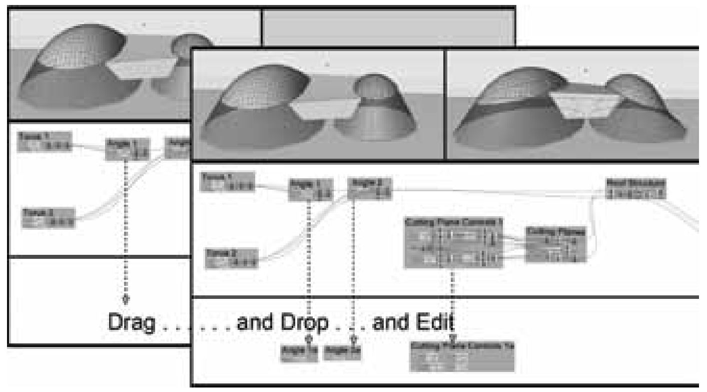


figure 4

figure 4

Once an alternative is initialized, dependency graph nodes can be dragged and dropped into it. Changing the dropped parameters changes the alternate design.

4.4 Cloning Parameters and Dependencies

Cloning and changing parameter values creates a variation (a new design). Cloning can be achieved by dragging and dropping an existing node from the prototype graph. Also, a new node may be inserted in the new panel; when the new node is connected with the prototype, it inherits features from the prototype. The new alternative becomes available for further editing by changing its parameters, by changing its dependencies, by substituting a node with another type of node, or by adding or removing nodes. We next describe how our prototype accomplishes each of these tasks for creating variations.

Figure 5 shows how a new variation of the Elephant House *Picnic* alternative is derived. In the new subjunctive graph panel, only the parameters that change are visible; these parameters inherit their dependencies from the original design, hence these dependencies can be hidden or shown as needed. The new alternative changes when (a) any inherited property from the base design changes, (b) a cloned parameter changes, or (c) a parameter introduced only in the new alternative changes. This strategy of reusing base nodes not only saves user time and effort, but also allows the user to manipulate both variations simultaneously by changing values of the common nodes.

4.5 Adding and Substituting Nodes

Although, in any system, adding or removing features can be done by just working on a new copy of the file, this method creates discrete instances. Comparison of variations can only be performed manually since the links are lost. In our system, users can develop variations to the existing design by adding new features or removing existing features. Using the subjunctive graph, users can substitute or replace certain features of a design. Figure 5 shows the *Stroll* and *Maze* alternatives derived from the Picnic alternative as the prototype. They inherit all properties from the prototype, except they override the tori angle and cutting plane values; they also respectively substitute the viewing bridge with different connection designs. The substitution takes place by creating new structures in the relevant subjunctive graphs, connecting them to the prototype, and removing the substituted nodes from the alternative.

4.6 Merging Variations

Design space can be explored by merging features from multiple alternate designs. Users can drag and drop into a new panel the nodes that are intended for combination.

4.7 Parallel Editing

Subjunctive dependency graphs allow users to edit alternatives in parallel. Changing the parametric value or dependency of a common node (a node common across multiple alternatives) will result in editing all dependent alternatives simultaneously. For example, in Figure 6, changing the value of *tori diameters* in the prototype results in changing all three alternatives in parallel. However,

figure 5

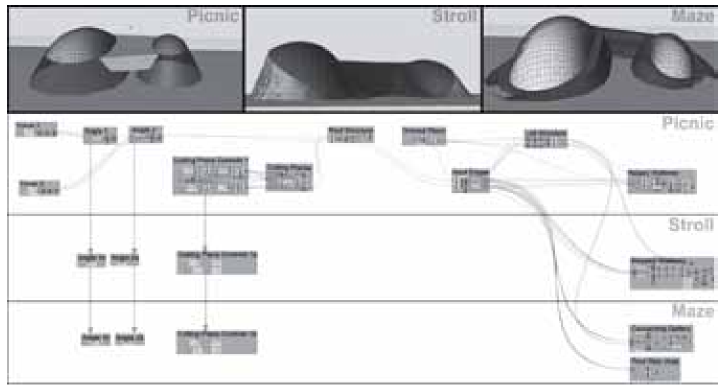


figure 5

The *Maze* and *Stroll* alternatives are derived from *Picnic*; in each, a new bridge design substitutes for the one in the *Picnic* prototype.

figure 6

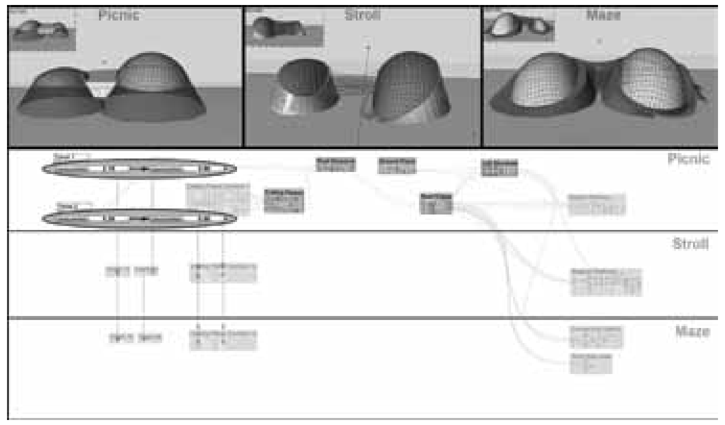


figure 6

Manipulation of tori radii in the *Picnic* prototype results in parallel changes to the derived radii in the *Stroll* and *Maze* branch alternatives.

changing tori angles in a branch alternative will change the tori configuration in that alternative only, where the tori angles are overridden and change locally (Figure 7).

4.8 Selective Manipulation and Modes of Interaction

In subjunctive graphs, the user can choose to view all variations at the same time or can switch between variations by selecting the corresponding geometric view or panel. This features aims at enabling users to focus on particular or overall aspects of design as parametric changes are applied. There are two modes of interaction. The first mode is node selection and brushing across workspaces. In this mode, the user selects a node, and in turn the interface highlights the selected node, the related nodes, and the edges in all subjunctive graphs (panels) (Figure 8).

The interaction takes place at the alternative level, where users can select an individual alternative; all the nodes relating or being used in that alternative get highlighted, and nodes relating other alternatives become transparent (Figure 9). Users can then further select nodes inside the highlighted panel to study only that alternative. During both of the interaction modes, if user makes a change to a node, that change propagates to all linked alternatives simultaneously.

Subjunctive dependency graphs also allow users to enlarge one single alternative at any time (Figure 10). During this interaction, all the nodes relating to the selected alternative are displayed by covering the workspace. All other panels and geometric views disappear. In this mode, if the user makes changes to any node, it will only affect the current alternative until other alternatives are displayed.

The system features mentioned in this paper are not exhaustive. We selected the important ones, and we leave a more detailed description of the prototype to another paper. However, we believe that these demonstrate the capabilities of subjunctive graphs in parametric design modeling.

figure 7

Manipulation of tori angles and cutting planes in the derived graphs changes local states.

figure 8

Single-node selection shows the immediate dependencies between the selected nodes and others.

figure 7

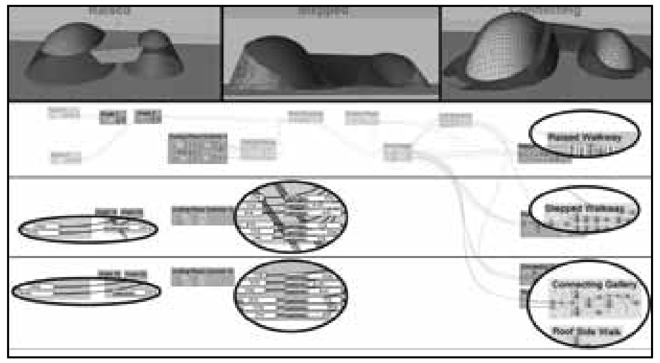
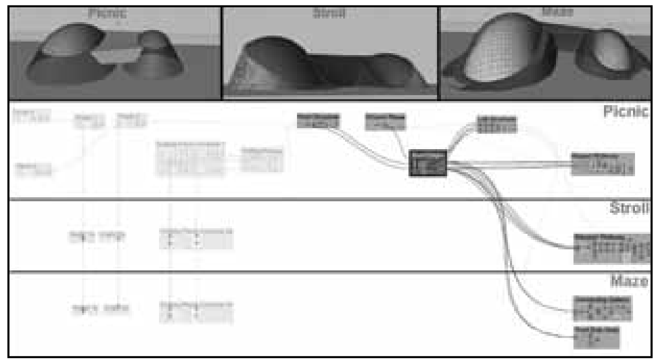


figure 8



5 RELATED WORK

There are few systems that experiment with parallel exploration of alternatives. The "subjunctive interfaces" introduce interaction techniques for exploratory analysis of scenarios side-by-side and editing those scenarios in parallel (Lunzer and Hornbæk 2008). A fully developed subjunctive interface would support selecting multiple values of interest for multiple parameters, and executing all combinations of values in parallel to produce variations respectively corresponding to each set of parameters. Moreover, a designer could choose to edit values in single variation or in all variations simultaneously, as required. The concept of subjunctive interfaces has been demonstrated in three domains: information access, real-time simulation, and document design. Our approach extends the conceptual ground of subjunctive interfaces through dependency graphs, while providing concrete user interface mechanisms for parallel development, comparison, and revision of graph-based parametric CAD models.

Terry et al. (2004) propose *parallel pies* and *side views* for exploring alternate chains of commands for manipulation of pixel-based images. Alternatives may be invoked before, during, or after the execution of a command. Subjuntive dependency graphs extend the work of Terry et al. into the area of dynamic structures. Hartmann et al.'s *Juxtapose* (2008) enables exploration of interaction design alternatives by showing side-by-side live graphical user interfaces, while enabling runtime code editing of shared parameters.

The modeling software called SolidWorks© exemplifies parallel exploration of scenarios through side-by-side calculations in spreadsheets. Its *Design Tables* associate feature parameters (columns) with parameter values (rows). Both design tables and subjunctive dependency graphs face similar issues as the number of alternatives grows.

Enhanced history mechanisms enable users to revisit old commands and edit them to ask "'what if' questions (Derthick and Roth 2001) and explore design space (Woodbury et al. 2000; Klemmer et al. 2002).

figure 9

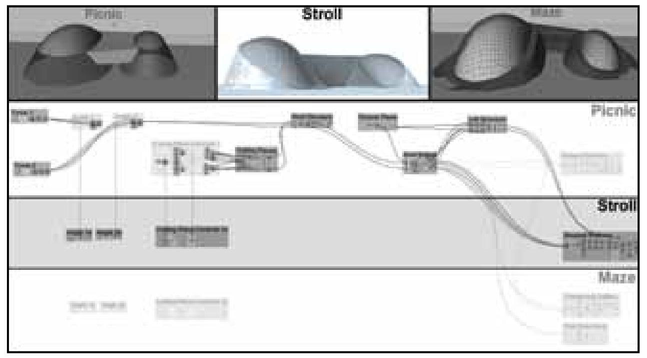


figure 10

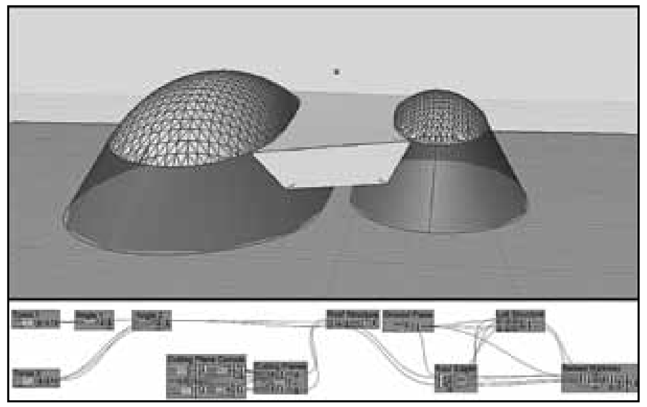


figure 9

Selection of an alternative highlights related nodes across all panels.

figure 10

Focusing on a single alternative hides all panels from the workspace and displays only the related geometry and graph.

Because expert designers defer their decisions in parametric systems, and often replay history to edit, refine, and explore design features (Woodbury 2010), we expect that enhanced history and subjunctive dependency graphs are deeply related, and together facilitate decision making and explanation.

6 DISCUSSION AND FUTURE DIRECTION

Parallel exploration of multiple design alternatives in a parametric CAD system is an under-researched area. Addressing this domain, we are exploring how dependency graphs in a parametric system can be used to develop parallel design solutions. We have introduced a prototype-based system to create design variations and subjunctive graphs to interact with them. The prototype allows its users to edit single or multiple alternatives at the same time.

To represent and work on parallel alternatives, multiple side-by-side views are made available in the same workspace. The dependency graphs shows what is different in each alternative. For clarity, each panel shows only what is different from the other panels (parametric changes, new nodes, new dependencies).

In an earlier study (Shireen et al. 2012), we found not only that participants preferred paneled graph display over small multiples to compare alternatives, but also that the study opened further questions, such as being able to move panels to better compare them. Comparing derived alternatives to each other (rather than each to the base) is difficult, especially if they have significantly different dependencies. While our prototype, based on the earlier study, enables switching between viewing a single alternative to viewing several simultaneously, the effectiveness of this strategy has yet to be verified.

Our goals include developing more advanced prototypes, and conducting studies with realistic design models.

ACKNOWLEDGMENTS

This research was partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grants Program and the Graphics, Animation and New Media Network of Centres of Excellence.

REFERENCES

Aish, R., and R. Woodbury. (2005). Multi-Level Interaction in Parametric Design. In SmartGraphics, 5th International Symposium, SG2005, 151–62, eds. A. Butz, B. Fisher, A. Kruger, and P. Oliver. LNCS 3638. Springer.

Akin, Ö., and C. Lin. (1995). Design Protocol Data and Novel Design Decisions. DesignStudies 16 (2): 211–236.

Cross, N., and K. Dorst. (1998). Co-evolution of Problem and Solution Spaces in Creative Design: Observations from an Empirical Study. Computational Models of Creative Design IV, eds. J. Gero and M. L. Maher. University of Sydney, NSW, Australia.

GenerativeComponentsTM (GC). (2011). Bentleys System Inc.

Green, T. R. G., and M. Petre. (1996). Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. Journal of Visual Languages and Computing 7(2): 131–74.

Hartmann, B., L. Yu, A. Allison, Y. Yang, and S. R. Klemmer. (2008). Design as Exploration: Creating Interface Alternatives through Parallel Authoring and Runtime Tuning. Proceedings of UIST, ACM.

Kasik, D., W. Buxton, and D. R. Ferguson. (2005). Ten CAD Challenges. IEEE Computer Graphics and Applications 25 (2): 81–92.

Klemmer, S. R., M. Thomsen, E. Phelps-Goodman, R. Lee, and J. A. Landay. (2002). Where Do Web Sites Come From?: Capturing and Interacting with Design History. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Changing Our world, Changing Ourselves, 1–8. ACM.

Krish, S. (2011). A Practical Generative Design Method. Computer-Aided Design, 43: 88–100.

Lunzer, A., and K. Hornbæk. (2008). Subjunctive Interfaces: Extending Applications to Support Parallel Setup, Viewing and Control of Alternative Scenarios. ACM Transactions. Computer-Human Interaction 14 (4), Article 17, 44 pages.

Marks, J., B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, and T. Kang. (1997). Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97, 389–400. ACM Press and Addison-Wesley Publishing Co.

Michael T., E. D. Mynatt, K. Nakakoji, and Y. Yamamoto. (2004). Variation in Element and Action: Supporting Simultaneous Development of Alternative Solutions. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04), 711–18. New York: ACM.

Peters, B., (2008). The Copenhagen Elephant House: A Case Study of Digital Design Processes. Proceedings of the 28th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA), 134–41.

Shepherd, P., R. Hudson, and D. Hines. (2011). Aviva Stadium: A Parametric Success. International Journal of Architectural Computing 9 (2): 167–86.

Shireen, N., H. I. Erhan, R. Sanchez, J. Popovic, B. Riecke, and R. F. Woodbury. (2011). Design Space Exploration in Parametric Systems: Analyzing the Effects of Goal Specificity and Method Specificity on Design Solutions. Proceedings of the 8th ACM Conference on Creativity and Cognition, 249–258. New York.

Shireen, N., H. I. Erhan, L. Bartram, and R. F. Woodbury. (2012). Visualizing Parallel Design Alternatives of Parametric Graph-Based CAD Models. Internal Technical Report, Computational Design Group, School of Interactive Arts and Technology, Simon Fraser University, Canada.

Shneiderman, B. (2007). Creativity Support Tools: Accelerating Discovery and Innovation. Communication of ACM 50 (12): 20–32.

Simon, H. (1973). The Structure of Ill-Structured Problems. Artificial Intelligence 4: 181–203

SolidWorks. (2010). Dassault Systèmes SolidWorks Corp. http://www.solidworks.com/.

Taivalsaari, A. (1997). Classes Versus Prototypes: Some Philosophical and Historical Observations. Journal of Object-Oriented Programming 10 (7): 44–50.

Terry, M., and E. D. Mynatt. (2002). Recognizing Creative Needs in User Interface Design. In Proceedings of 4th Conference on Creativity and Cognition, 38–44. New York: ACM.

Woodbury, R., S. Datta, and A. Burrow. (2000). Erasure in Design Space Exploration. Artificial Intelligence in Design 2000: 521–44.

Woodbury, R. F. (2010). Elements of Parametric Design. Routledge.

SYNTHESIZING DESIGN PERFORMANCE: AN EVOLUTIONARY APPROACH TO MULTIDISCIPLINARY DESIGN SEARCH

David Jason Gerber,
USC School of Architecture

Shih-Hsin Eve Lin,
USC School of Architecture

ABSTRACT

Design is a goal-oriented decision-making activity. Design is ill defined and requires synthetic approaches to weighing and understanding tradeoffs amongst soft and hard objectives, and imprecise and/or computationally explicit criteria and goals. In this regard, designers in contemporary practice face a crisis of sorts. How do we achieve performance under large degrees of uncertainty and limited design cycle time? How do we better design for integrating performance? Fundamentally, design teams are typically given neither enough time nor the best tools to design explore, generate design alternatives, and then evolve solution quality to search for best fit through expansive design solution spaces. Given the complex criteria for defining performance in architecture, our research approach experiments upon an evolutionary and integrative computational strategy to expand the solution space of a design problem as well as presort and qualify candidate designs. We present technology and methodology that supports rapid development of design problem solution spaces in which the objectives of three design domains have multidirectional impact on each other. The research describes the use of an evolutionary approach in which a genetic algorithm is used as a means to automate the design alternative population as well as to facilitate multidisciplinary design domain optimization. The paper provides a technical description of the prototype design, one that integrates associative parametric modeling with an energy use intensity evaluation and with a financial pro forma. The initial results of the research are presented and analyzed including impacts on design process; impacts on design uncertainty and design cycle latency; and the affordances for "designing in" performance and managing project complexity. A summary discussion is developed that describes a future cloud implementation and the future extensions into other domains, scales, tectonic, and system detail.