TAGGED AND INTERACTIVE DIAGRAMS OF DESIGN INTENT AND RATIONALE

BHARAT DAVE AND GWYL JAHN

University of Melbourne, Melbourne, Australia
b.dave@unimelb.edu.au

Abstract. This paper describes our experiments with diagrammatic representations to think about design compositions and to learn from shared accretion of design knowledge. We describe here implementation of an online prototype that, on the one hand, offers interactive functionalities to externalise thinking about design compositions in the form of diagrams and, on the other hand, which acts also as a repository of diagrams that can be dynamically interrogated to find other proximate compositional thinking and ideas related to a particular design position. Put differently, the prototype helps both notate design thinking and draw out associations between separately notated design thinking.

Keywords. Diagrams; compositional logic; design representations; knowledge accretion; reflective thinking.

1. Introduction

Although different in materials, components, governing geometry and visual character, the three objects shown below share a common interest to explore the concept of ‘series’. The presence of such shared threads across projects – either in the final material assemblies or in how they were made, may be sometimes hard to notice. But once recognised, they open doors to further fruitful experimentations and explorations in design. To foster recognition and use of such higher level generalised concepts and operations in design, we developed an online repository of diagrams that is associative, incremental and dynamic. The online prototype system helps both notate design thinking and draw out associations between separately notated design thinking. This paper describes motivations that informed the design and implementation of the prototype, its benefits in design projects, evaluation of its use and plans for future extensions.
1.1. PROTOTYPE IN ACTION

The following figure illustrates main interface components of the implemented prototype. The left-half of the figure shows an image of a fabricated model for the currently selected design project and its associated diagram with design and fabrication sequence. The right-half of the figure shows a number of image tiles of other projects in the repository. Each of these image tiles are clickable and act as pointers to access different data associated with each project (as displayed on the left half of the interface).

2. Motivations

Our work is motivated by a number of intersecting ideas and observations, many of which are noted earlier by other researchers. There is increasing acceptance and appreciation for the need to support interactive 3D modelling using both direct manipulation and symbolic languages (as in scripting and programming languages). The burgeoning interest in interactive exploration of parametric design spaces has once again fore-grounded work in extending and integrating visual programming paradigms of the past with complex interactive 3D geometry modellers. These developments enable relatively novice users to exercise much greater expressive power in a short period of time. However, it also requires that users conceptualise and understand how to operate simultaneously between different representational spaces ranging from graphical, numeric, symbolic, and others.
Although initial learning curves for such interactive modelling tools may have become less steep, users may not find it easy to express and control anything but simple spatial compositions. The ease of visual programming is often replaced by incomprehensible clutter of symbolic graphs wherein interdependencies become hard to follow or trace and distilling generalisable knowledge may become difficult if not impossible in a forest of details. Also the frequent and continuous updates of software work against possible accretion of shared practices and reusable code.

When the preceding issues with visual programming and modelling systems intersect with architectural design and representations, they become more problematic. Since the very purpose of design is to generate something that does not yet exist, iterative design emerges as the preferred mode of practice. In this context, support for version histories may be expected to encourage parallel and divergent exploration of design alternatives using scripting or programming environments. However, the significant amount of effort required for managing complexities of digital representations and operations in a design project very often propel projects along a singular developmental trajectory. These issues become apparent, albeit with varying intensities and forms, while teaching architecture students to design by scripting. Ideally, such undertakings in academic and professional settings should be accretive, exploratory, and independent of particular software environments. They should be accretive, i.e. build on others’ work rather than always start from the same low thresholds. They should be exploratory in that they foster divergent and lateral thinking rather than rapidly converging on to a solution at least in the early design stages. And they should foster higher-level compositional skills rather than be constrained by software instrumentalities.

2.1. RELATED IDEAS

Our work draws upon a number of recurring themes in development and adoption of computational approaches that respond to particular needs for design representations and modes of thinking. The notion of patterns in design appears
in literature from vernacular architecture (Alexander 1977) to formalised rule books (Durand 2000). The notion of levels of abstractions in design appears from cognitive studies of designers (Gero 1998) to pedagogical traditions in design (Fischer et al. 2000). The notion of internalising and using design moves of various grain sizes appears in the context of how designers think to how users develop models of software systems and how effectively and efficiently they can use those tools (Coyne et al. 1993, Pantazi 2008, Maleki and Woodbury 2010, Woodbury 2010). Our work builds upon and extends these ideas by developing a particular set of ideas and testing it in the form of an interactive, incremental, online catalogue of fragments of design thinking at a level of abstraction that is independent of constraints of specific languages, tools or materials.

2.2. FOCAL ISSUES

An underlying interest in understanding and distilling higher level design thinking guided a number of studies cited above. These efforts have led to development of some promising tools and environments such as the visual programming environment of Grasshopper or in-place scripting supported in Generative Components. While these and other similar efforts are promising, we are interested in one particular cluster of issues: how to foster higher level compositional design knowledge without being too closely tied to a particular implementation, one that supports divergent thinking and allows associative exploration of ideas, and one that can be incrementally added to over time. Further, we wish to emphasise thinking about design compositions as a *process of making* and not just as a penultimate state description as in traditional design drawings. The above issues informed development and implementation of an online prototype that, on the one hand, offers interactive functionalities to externalise thinking about design compositions in the form of diagrams and, on the other hand, which acts also as a repository of diagrams that can be dynamically interrogated to find other proximate compositional thinking and ideas related to a particular design position. It should be added here that diagrams are but one among many other useful representations to notate design thinking – prior to, during and in retrospect.

3. Prototype architecture

The prototype system comprises a set of interconnected components. A standard WordPress installation is used as the primary backend component which supports composition of different data types including text, images, video, and others in the form of posts and pages. Diagrams are added to this collection
TAGGED DIAGRAMS OF DESIGN INTENT AND RATIONALE

673 TAGGED DIAGRAMS OF DESIGN INTENT AND RATIONALE

of media types using an interactive diagram builder implemented in Flash. This component also allows grouping of diagrams (displayed as tabbed pages) based on a simple naming convention. The diagram builder is integrated with PHP which turns text strings embedded within diagrams into *tags* similar to how they are used in the standard WordPress metadata. The automatic tagging of text chunks in diagrams allows specific instances of various data and also entire posts to be associated with each other. The front end is managed using a combination of PHP scripts and JQuery functions to generate dynamic CSS/HTML code that formats and renders user-level information display.

4. Prototype functionalities

The prototype system revolves around three kinds of users with different privileges. The ‘reader’ role allows anyone, anywhere on the net to browse and query different information chunks including diagrams. The ‘creator’ role enhances ‘reader’ privileges with creation and modification of diagrams and posts generated by a registered user. The ‘admin’ role has all the privileges including moderation and publishing of diagrams and posts.

4.1. DIAGRAM CREATION

The interactive diagram builder supports a small number of named container nodes (represented as rounded rectangles): data, function, object, image and fabricate. Data nodes contain parameter(s) or variable(s) that are used by other nodes. Function nodes represent a series of operations, possibly using data received from and passed to other nodes. Object nodes represent repeatable elements, e.g. a panel or a rib, with variable properties. Image nodes point to locally stored or network-accessible image/video data. Fabricate nodes represent specialised chunks of operations to translate generated geometry to suit the needs of specific fabrication technologies and tools, e.g. laser cutter, 3D

Figure 4. System architecture.
printing, etc. Each of these nodes has two connectors at top left and bottom left corners which represent input and output to/from nodes respectively.

To create a new diagram, nodes are dragged out from the toolbar onto the blank canvas. The flow of information between nodes is represented by drawing a connection from output (bottom left) connector of one node to input (upper left) connector of another node. The nodes can be annotated using free form text which is displayed inside the node rectangle. The nodes can be repositioned manually anywhere on canvas; the connecting edges between nodes are maintained by the system.

4.2. USE SCENARIOS

Our prototype implementation supports declaration of compositional intents prior to, during and retrospective to a design project. This is facilitated by interactive composition of a series of diagrams by the user, each of which may be associated with other design diagrams via ‘tags’. The dynamic repository of diagrams conceptually resembles ‘tag clouds’ of design intent and rationale. The uses and benefits of the implemented interactive prototype are briefly described and illustrated next.

4.2.1. Modular thinking

The use of diagrams fosters modular thinking about design moves in terms of required design elements and operations on them (Figure 6, left). It encourages a degree of higher level meta-designing instead of being caught up with all the
details simultaneously at a small grain size. The explicit recognition of input and output data reinforces the notion that design compositions are materialised outcomes of a temporal assembly process and that \textit{correct sequence} matters (in the sense that each distinct sequence may lead to unique outcomes).

![Figure 6. Left: Sequenced and modular design moves. Right: Tabbed idea histories.](image)

4.2.2. Idea histories

A series of diagrams supported by our prototype enables occasional look back and (re)tracing of paths followed and perhaps discarded. The use of multiple, alternate diagrams (Figure 6, right) encourages what-if explorations in design and provides a concise record of design history that mitigates against early idea fixation and encourages reflective assessment.

4.2.3. Proximate ideas

The tagged associations between diagrams bring to fore proximate design ideas, a difficult thing to achieve in analogue and discrete representations. The tags operate at two levels. When used as a search term in the main interface (as in Figure 7), the system highlights all projects that are similarly tagged. When used from within diagrams by clicking on a string (for example, ‘series’ as in Figure 1), all similarly tagged projects are highlighted.

![Figure 7. Query and display of proximate design ideas.](image)
4.2.4. Grain size of representations

Diagrams enforce choices about grain size of representations and thus help reinforce thinking about when and how to generalise or disambiguate specific details. This has played out in our system in two ways. Since the diagram builder provides only a limited canvas, users become selective about what to describe in diagrams with brevity and clarity. At other times, users develop a working script first, draw the diagram, and then update their scripts as the next reflective step.

4.2.5. Recombinant fragments

Diagrams as fragments of design intents and operations make possible creative puzzle-making in which fragments may be recombined to form new design compositions on the fly. This is enabled by tagged searches (Figure 8) and the possibility to combine parts of other compositions or their design logics to generate novel design outcomes.

Figure 8. Pointers to local and remote data.

4.3. EVALUATION

The prototype system has been used most recently by a group of 20 students, many with no prior exposure to computational thinking or experience in scripting. The students faced two simultaneous demands: learning to script while also developing design ideas to be generated using scripts, and then fabricated and assembled. The students were asked to use interactive diagramming system as part of their project development process.

The feedback from a representative student group offers interesting insights about similarities and differences between traditional designerly ways of thinking and computationally supported explorations, and how the use of diagrams supported by our system mediates between the two modes.

Most students generated diagrams towards the end of their projects rather than as an integral part of design exploration from the beginning of their projects. This can be ascribed to the need to know what is possible (using
scripting) before deciding on what to design. As one student put it: “… I was new to Grasshopper ... I didn’t know and needed to work out what I didn’t know”. As a result, she generated the diagram at the end, more as “a description of process” rather than as a “planning tool”. However, having generated a series of diagrams, she “…understood a lot more” about both the range of design possibilities she could explore and also alternative computational procedures through which to realise them. This retrospective abstraction of her design project also highlighted the fact that what she explored initially in terms of design possibilities was limited by the handful of scripting functions she learnt which, in turn, limited her design ambitions. The use of diagrams allowed her to step back from these limitations and reframe her design intentions in a way that was independent of what she knew or what the function set of a particular scripting environment supported. The need to articulate and represent diagrams within a limited space – unlike the infinite scripting canvas – forced her to rephrase in “clear and abstract terms” the object of her composition and the sequence in which it was to be generated.

An experienced design student employed clusters of coloured nodes in his script which were then used as building blocks of diagrams he generated. This correspondence between a particular set of scripted nodes and their abstracted descriptions in diagram allowed him to better understand and explore the notion of generic types and specific instances.

A student with limited design and scripting experience noted: “… in the beginning, I have no idea what I am doing or why”. The use of diagrams and the free form text embedded in diagram nodes gave him “goals ... a direction” to pursue, helped by looking at others’ pseudo-code and diagrams.

Interestingly, another student commented that diagrams allowed him to reflect on similarities and differences between two different operations, such as lofting and extrusion, which may result in similar design outcomes but require different intermediate information and processing steps.

Finally, the use of diagrams forced students to articulate and communicate stage wise planning and making of material objects in ways that are substantially different from traditional design drawings since the latter act more as state descriptions of final outcomes and do not reveal processes of making and assembling components.

5. Future extensions

The growing numbers of projects in which the system is used and feedback received suggest a number of future extensions. One such significant extension will support embedding of diagrams within diagrams. It will help represent and foster thinking about design operations at multiple levels of details
and grain sizes, by collapsing or unpacking diagram nodes. A related extension will allow diagram nodes to directly point to pseudo-code of specialised components or generalised procedures/functions. Further, it will be useful to add suggestive pointers in the form of contextual pop-ups with freeform text strings as they are added to describe diagram nodes. Such pointers could suggest other visual examples, pseudo-code fragments, specific operations, packaged components or plug-ins. The common thread running through these future extensions is our interest in fostering design exploration by supporting divergent paths at as many points in the early design process as possible. We also wish to cultivate higher level thinking about design that is independent of specific programming environments.

Our prototype system is accessible to anyone from anywhere on the net whereas the ability to create and modify diagrams is moderated and restricted. Until now, the system has been used locally in advanced graduate design subjects; we plan to make it available to others in near future.

Acknowledgements

This project has benefited from the work of a number of students in the Faculty of Architecture, Building and Planning, University of Melbourne during 2008–2011. The prototype is accessible online at http://scripts.crida.net/gh/dda11.

References