CAPTURING PARAMETRIC DESIGN EXPLORATION PROCESS

Emperical insights from user activity and design states data

VERINA CRISTIE[1] and SAM CONDRAD JOYCE[2]
[1,2]*Singapore University of Technology and Design*
[1]*verina_cristie@mymail.sutd.edu.sg* [2]*sam_joyce@sutd.edu.sg*

Abstract. Computational design, especially parametric associative modelling tools, have opened a whole new world of possibility in design exploration. However, their now established use poses further questions regarding how they effect design process and ultimately the quality of the outcomes. Answering those questions requires a better understanding of parametric design process through empirical data. In this paper, we extend a method to systematically capture the design process into a structured data of designer's activity and design states. Analysis of design sessions reveal a unique pattern of parametric modelling and exploration strategies produced by each designer. Capability to save design process into structured design states shows potential to improve process.

Keywords. Design exploration; Parametric Design; History Recording; Version control; Conceptual Design.

1. Introduction

Parametric modelling is a common approach used effectively in many modern design practices (De Kestelier and Peters, 2013). When applied at the early exploration stage, it has the ability to increase the impact of a user's design effort (MacLeamy, 2004). Owing to its growing importance, research has been undertaken to understand and improve parametric design workflows. Davis (2013), investigated flexibility issue in parametric design workflow. Whilst (Smith, 2007), explored 'breaking' models problem while creating a major changes.

Data has been used to understand and solve shortcomings in processes and workflows of numerous fields, however design specific examples are relatively few. Meagher et al. (2013) saved user-edited simulation code using software versioning system upon compilation. Changes detected in the code could reflect user's iteration of modifying geometry and running simulations, which eventually could help to understand how users interact with simulation results in a parametric design process. Sakai and Tsunoda (2015) allowed an explicit way of saving design states by giving users the opportunity to choose a design, edit, and save their own versions within *3D WebGL* based platform. These two approaches embodied

Intelligent & Informed, Proceedings of the 24th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2019, Volume 2, 491-500.

Aish's (2000) and Woodbury (2017) idea of recording changes transaction and design alternatives for concept reuse and design exploration, or more commonly termed as versioning. Revit and similar BIM platforms save states and 'merge' multiple users models together, but the emphasis is on creating singular unified model, and not on capturing or understanding design process.

The concept of versioning is not new. First versioning system was developed to systematically save code of operating system (Rochkind, 1975. Modern versioning systems such as Git (https:git-scm.com) or SVN (https:subversion.apache.org) are ubiquitous in software developments. With a versioning system, overall progress of software development could be tracked and it is easier for programmers to collaborate. Additional feature is developed on a separate branch of the code - a different 'version', before merging the branch back to the main version again. Should code in a certain version fails, programmers would then revert to previous working version. Such availability of code saved at various stages from various projects allows for empirical analysis to improve software design. For example, D'Ambros et. al. (2008) mined software repositories to understand how software evolved, specifically by detecting hotspots (areas of code that frequently change), and measuring impact of code change to the whole system. In this manner, it is possible to predict further defects in the system.

In this paper, we are interested to understand what designers actually do while designing with parametric tools, and what that process looks like by logging designers' activity in a set design task. Designers go through different process of changing parameters 'sliders' and associative logic 'components' and 'wires' to achieve their desired geometry. Potentially, this could offer a new way to understand and improve design support. In particular, we would also like to see if giving the users the capability to capture intermediate design progress impacts design activity. In the next part, the tool developed for this research and the data captured will be explained.

2. A tool to capture parametric design activity

To record parametric design exploration states, the *Grasshopper* Plug-in *GHShot* has been used and extended. *GHShot* is a tool developed by the authors and allows for versioning of parametric model state or 'snapshot' to a cloud repository with a single button action (Cristie and Joyce, 2017, 2018). Each snapshot of parametric design exploration consists of its: (i) *Grasshopper* parametric design definition (as an *XML*), (ii) identified and named input parameters and/or performance values that designers choose to save, (iii) 3D mesh geometric representation, and (iv) designer's notes. Meta-data such as time of saving, and designer name (in case a file is modified by more than one designer) is also saved. Upon saving designers are allowed to specify if the current state is a continuation of previous state, or a branch of previous state (e.g.: making variations of design). The snapshot saved can be viewed online where progress could be observed in a photo album and tree-like visualisation. Performance values of all snapshots sent are also shown in a graph view with filtering mechanism, to allow data exploration.

While *GHShot* saves design states upon designer's command and displaying them in the corresponding custom web interface, in this work an extension plug-in *GHRecorder* is developed. *GHRecorder* periodically sends records of design activity to the server automatically. Design activity is defined as all actions designers do within their *Grasshopper* document. For example: adding components, connecting components, arranging/dragging the components on the canvas, copy paste, etc. Records of these actions and the time they are done are sent to the server.

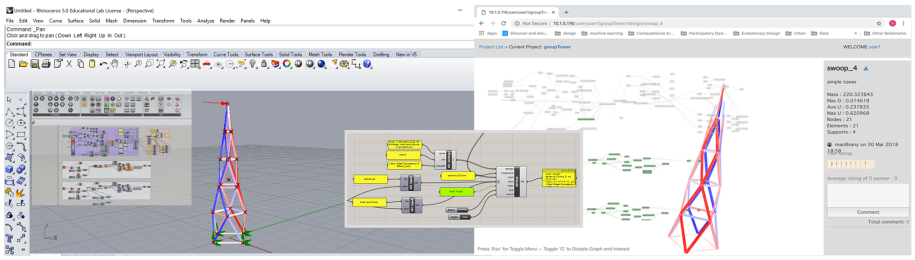

Figure 1. GHShot (center) is used to send parametric design states from designer's Grasshopper file (left) to web (right). State of parametric model (both 3D model and graph connection) is captured and changes (green colour) could be detected.

3. Experiment Design

To understand what designers do during parametric design process and if *GHShot* has effect on this process, experiments with two tasks were designed. In total, eight designers with architectural backgrounds, consisting a mix of undergraduate and postgraduate students were recruited. They have a range of experience in designing from 2 to 15 years, and *Grasshopper* experience from 1.5 years to 10 years. Five designers were to do the first task without *GHShot* and second task with *GHShot*. Three designers were to do both tasks without GHShot as the control group. For all the design tasks, *GHRecorder* logged participants' design activity in the background. In addition, there was screen recording for all the design tasks, and interviews were conducted at the end of the second design case for participants using *GHShot*.

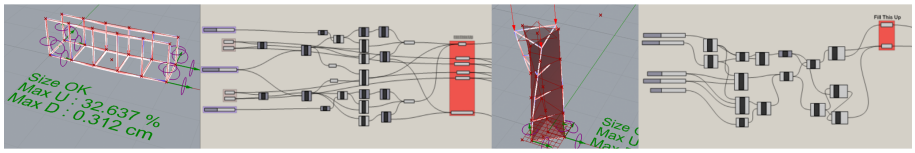

Figure 2. Parametric Bridge Experiment Setup (left): the parametric model consists of 45 components. Parametric Tower Experiment (right): the model consists of 24 components.

A parametric bridge model was used as the first design case, and a parametric tower model as the second. Relatively simple parametric structural design tasks were chosen as they provided flexibility to design aesthetically whilst being

mindful of some performance criteria. For both design cases, designers were to meet requirements on the parametric model's maximum geometric size, maximum utilisation, and maximum deflection. When any of these three requirements were not met, green labels at the side of the 3D model would turn to red (see Fig. 2).

Each participant was given one-hour design time for each of the cases, with a gap of at least three hours before doing the next case. A minimum of 40 minutes of designing was imposed, but participants were allowed to continue over the hour to complete if desired. At the end of the experiment, participants were asked to submit their final design, a design of their liking. To maintain anonymity all users were given a two letter name such as 'HD' or 'MJ'.

Between scale of 1 being non-familiar to 10 being very familiar with structural knowledge, most participants rated themselves with 5 or 6; only one rating themselves 10. Most participants have 0.5 to 1 year of experience with *Karamba*, a *Grasshopper* plugin used for analysing parametric structures, however this was not necessary in this experiment as participants were not allowed to modify the settings on *Karamba* components and the input was basic. They were only to modify the parametric model definition relating to the geometric representation.

4. Results and Discussion

4.1. VISUALISING PARTICIPANT'S ACTIVITY

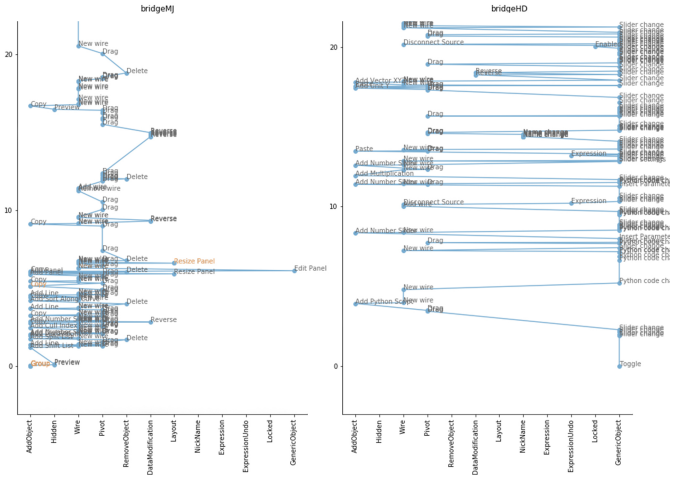

Figure 3. Activity Visualisation for first 20 minutes of bridge design session. User MJ (left) performed less activities than user HD (right). Combination of Add Component – Add Wire is prominent in MJ's while Python Code – Slider Change is prominent in HD's.

Data obtained from designers' activity records was processed and visualised to provide insights into the design process. With activity timeline visualisation, one could observe how rapid designers navigate from one action to another action. For example, in figure 3 we can see on the left a user 'MJ' who immediately changes

the model in the first 10 minutes before slowing down with more careful edits, whilst on the right an initial exploration of sliders was undertaken by user 'HD' before editing was begun.

4.2. POPULAR ACTIVITIES

Eleven unique *Grasshopper Actions* (titled by the software itself) are captured in the activity records; these titles each cover a range of similar actions. They have been grouped into 6 activity types useful for this discussion: Add, Wire, Input, Arrange, View, and Delete (see Table 1).

Table 1. Activity Type Grouping.

Activity Type	**Corresponding *Grasshopper* Actions**
Add	AddObjectAction (add components, copy and paste component)
Wire	WireAction (connect wire, remove wire, disconnect source/recipient)
	DataModificationAction (list related actions: flatten, simplify, graft, reverse)
Input	GenericObjectAction (slider change, remove parameter, code change, etc),
	ExpressionUndoAction (editing expression)
	PersistentDataAction (set boolean/line/integer, etc)
Arrange	LayoutAction (panel/number slider resize)
	NickNameAction (change of component name)
	PivotAction (component drag, component split, align components)
	ClusterPreviewDocumentAction (clustering related activity)
View	LockedAction (enable/disable a component)
	HiddenAction (preview)
Delete	RemoveObjectAction (delete component, ungroup component, cut component)

On average, designers performed 453 actions until the end of the bridge task, and 360 actions until the end of tower task. Average time spent on bridge task is 57.6 minutes, and 56.4 minutes for tower task. Minimum time spent on the task overall is 39 minutes, and maximum of 67 minutes. Aproximately 7 actions a minute. The highest number of action belongs to bridgeYY, 727 actions in 62 minutes, and lowest belongs to towers, 152 actions in 56 minutes. 'Arrange' activity type is the most popular, accounting for 37.7% of the total activities with 32.76% of this just component dragging alone. Next popular activity type is 'Wiring' at 25.21%, followed by 'Input' at 15.48%. For designers who employ coding strategy such that HD and SH, the use of sliders, which belongs to Input category, are generally higher.

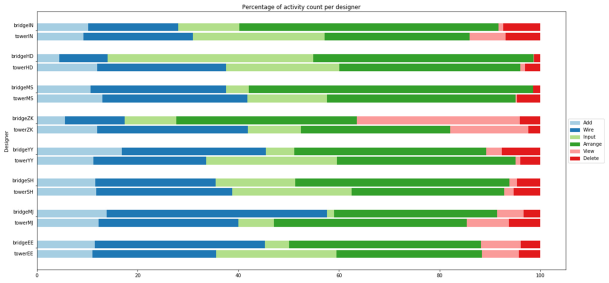

Figure 4. Percentage of Activity Count per Designer for bridge and tower parametric model.

Activity records also showed that although generally designers work similarly in bridge and tower case, designers 'fiddled' with input parameters more in the latter (19.5% as compared to 11.9%). Bridge case has higher 'Arrange' percentage, 42.3% as compared to tower's 33.1%. During interview, this is confirmed:

designers looked at tower structures as a surface system and hence dealing with the input parameters more, while designers looked at bridge structures more as elements of points and lines thus edited the underlying component associations more.

4.3. TYPES OF ACTIVITY : MODEL FIRST, EXPLORE LATER

Overall component addition progress are plotted based on add and delete activity throughout the session (see Fig. 5). Addition has the line to go up, while deletion has the line to go down. A sudden jump on the line usually signified a large delete or copy paste behaviour. For example, an interesting pattern could be detected on the EE's bridge: copying the whole definition at the start as a backup and removing the copied components at the end. Designer YY, on the other hand, on bridgeYY, deleted the whole parametric model definition as the session started, and built a brand new definition. Both of these behaviour exemplifies Woodbury's (2010) observation of two strategies in parametric design: (1) copy and modify, and (2) throw code away. In general, designers added up to about 40 components within the one hour duration.

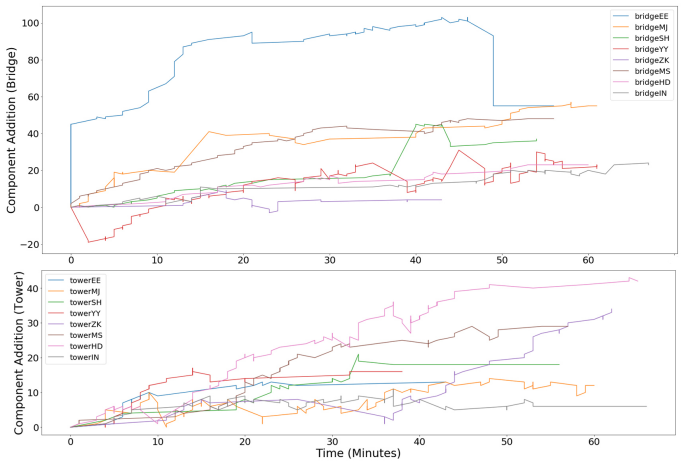

Figure 5. Component Addition Progress throughout design sessions for Bridge parametric model (top) and Tower parametric model (bottom). Addition is typically asymptotic - designers stopped adding components after a while; sliders to explore designs afterwards.

4.4. WHAT COMPONENT ADD ACTIONS FOLLOWS ANOTHER ADD?

'Component Adding Adjacency matrix' (see Fig. 6) shows which components are added after the proceeding one. The add components relationships have been grouped using Louvain Community Detection algorithm (Blondel et al., 2008) and then arranged in the matrix so that those which are used together are close. Most of the adjacent occurrence of add only happen one time, with top occurrence numbered at 6 for Add List Item and Add Number Slider.

Despite the limited data there are five groupings detected from top left diagonal to bottom right: (1) (mostly) basic arithmetic operation (light blue) - add division, add number slider, add list item, etc, (2) vector related addition (dark blue) - unit x, y, z, rotate3D, etc, (3) geometric related (light green) - add line, loft, polyline, merge, flatten tree, etc, (4) algorithm related (dark green) - such as Delaunay and Voronoi and, (5) mesh and surface related (peach)- add mesh brep, mesh surface, etc. Bottom right red colour group only consist of two data points and could not be considered as a group. This shows quite strong grouping of component usage (especially in the first three) and these categories are not the same as the existing grouping by type in the Grasshopper user interface, implying that component section could be possibly streamlined.

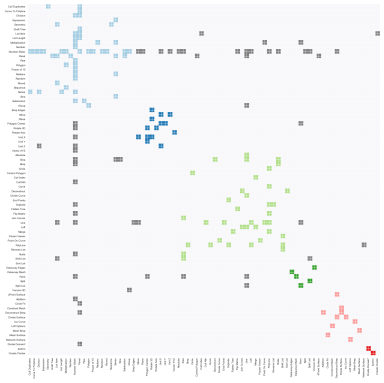

Figure 6. Add Components used in design sessions grouped by their occurences. Component addition on X axis is followed by corresponding component addition on Y axis.

5. An informed comparison: design footprints, design iterations, and GHShot

To have a more detailed look at design activity of each designer in each case, activity is visualized in form of heatmap (see Fig. 7). Types of activity are spread along the Y axis, with timeline in minutes on the X axis. Designers performing more actions have more intense activity footprints. E.g: bridgeYY as compared to towerSH. It is also obvious that Arrange activity has high intensity across all designer (as mentioned in 4.2 earlier), and generally are related to Wire and Add actions. The main strategy of adding component and wiring them at the beginning followed by more intensive use of sliders are adopted by most participants, for example designers EE, YY, and MS. For designer SH and HD, a scripting strategy is employed from the beginning of the session, hence the intensity of Input related activity (Slider Change) higher, spanning from beginning to the end of the session. It is also observable that it is possible to detect similar usage patterns by users between tasks. For example, bridgeSH and towerSH show low footprints and have Input activity spanning through the session intermittently.

Activity footprints further confirmed that all users designed in stages or bursts. Episodes of high activity usually are followed by easily identified gaps of no

modelling activity. During this period of modelling inactivity, designers reassess their design progress by observing the output geometry and further decide what to do next. Designers mentioned that the use of *GHShot* allowed them to look at overall design progress through the snapshots visualisation in the web during this period, as compared to only looking at immediate previous iteration without *GHShot*.

Figure 7. Activtiy Heatmap for each user for bridge case (left) and tower case (right). The vertical lines represent the time where the snapshots were taken.

Further, these episodes were broadly identified by two main action types: *Add-Wire-Arrange* - that is developing associative logic which signifies a creation activity, and *Input* - mainly consisting of slider activity (exploring options). A gap preceding *Add-Wire-Arrange* activity signifies an iteration of exploring a certain design concept (major iteration), while a gap preceding *Input* activity signifies an iteration of parameter exploration within that concept (minor iteration). It is observed that the use of *GHShot* does not modify these two types of iteration, but rather it becomes a clearer indication when these iterations happen. *GHShot* is used periodically in a longer interval to save progress throughout the design session (major iteration), and *GHShot* is also used in a shorter interval, with burst of snapshots (minor iteration). TowerMJ (see Fig. 7), is an exception as she forgot to use *GHShot* and only remember to send snapshots towards the end of session.

In terms of number of iterations and performance outcome, there is no difference observed for designers with or without *GHShot* (see Fig. 8). For both bridge and tower case, generally designers performed three to four iterations within the span one hour, with faster designer reached five and slower designer two. Designers mentioned that they become more aware of the performance of

the design with the graph view available in the web. One gave feedback that the performance graph in the web viewer of *GHShot* challenged them to make the most efficient structure (see Fig. 9).

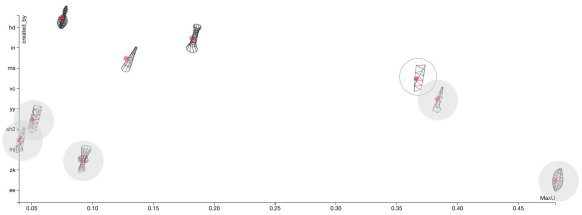

Figure 8. Final tower design from each designers. Towers completed using GHShot are shaded in gray, original design in a white circle. Six designers reduced utilisation value of the original structure.

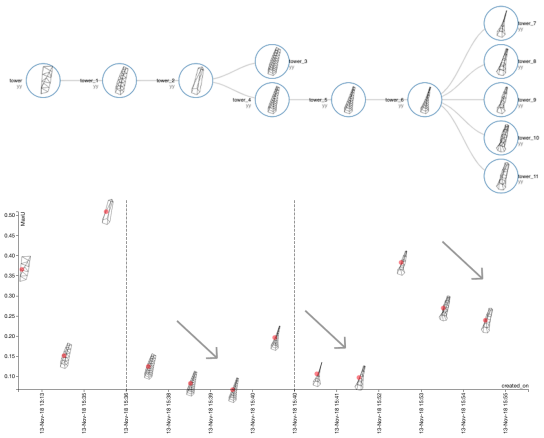

Figure 9. TowerYY design progression. Top: Tree Graph showing different concepts the designer explored at each branching point. Bottom: Continuous effort to improve performance at each iteration.

6. Conclusion and Future Work

In this work, we have demonstrated how capturing designers' actions within *Grasshopper* could reveal user behaviour, including iterations of modelling and exploration action preference during design process. Data captured serves as design footprint unique to a user, but in aggregate telling on how people interact with parametric software. It has shed light on different and shared strategies designers adopt in parametric modelling and design explorations. Grouping consecutive add actions has shown different modelling types performed by designers. Footprint of data collected had shown potentially focus areas of

intervention in improving parametric modelling. For example, if on average 37.7% of designers' time during the session is spent on arranging components this is something that could benefit from better automated tools and/or support.

Capturing design states with *GHShot* exhibits two patterns of recording: periodic snapshots to save design progress and burst snapshots to save outcome by fiddling with input parameters. Designers gave positive feedback on the ease of use of the tool and how the tool gave them capability to look at the overall design progress and design variations and not having to worry of missing previously developed but unsaved parametric model as long as they remember to click to send to server periodically.

Design process can be better understood by capturing user activity and design states empirically. This is the first step before improving the process in the long run. For our further work, we look forward to repeat similar design sessions with different design cases for more participants as it is still inconclusive to say if a certain process would yield a better actual design-result performance due to open-ended design case with limited number of participants in current experiment.

References

Aish, R.: 2000, Collaborative Design using Long Transactions and "Change Merge", *Proceedings of eCAADe 18*.

Blondel, V.D., Guillaume, J.-L., Lambiotte, R. and Lefebvre, E.: 2008, Fast unfolding of communities in large networks, *Journal of statistical mechanics: theory and experiment*, **10**, P10008.

Cristie, V. and Joyce, S.C.: 2017, Capturing And Visualising Parametric Design Flow Through Interactive Web Versioning Snapshots, *Proceedings of International Association for Shell and Spatial Structures Symposium*.

Davis, D.: 2013, *Modelled on software engineering: Flexible parametric models in the practice of architecture.*, Ph.D. Thesis, RMIT.

D'Ambros, M., Gall, H., Lanza, M. and Pinzger, M. 2008, Analysing Software Repositories to Understand Software Evolution, *in* T. Mens and S. Demeyer (eds.), *Software Evolution*, Springer, 37-67.

X. De Kestelier and B. Peters (eds.): 2013, *Computation Works: The Building of Algorithmic Thought*, Wiley.

MacLeamy, P.: 2004, Collaboration, Integrated Information, and the Project Lifecycle in Building Design and Construction and Operation, *The Construction Users Roundtable*, WP-1202.

Meagher, M., Park, A. and Nembrini, J.: 2013, Analysing The Performance-Based Computational Design Process: A Data Study, . *In Rethinking Prototyping: Proceedings of 4th Design Modelling Symposium*, 359-375.

Rochkind, M.J.: 1975, The source code control system, *IEEE transactions on Software Engineering*, **4**, 364-370.

Sakai, Y. and Tsunoda, D.: 2015, Decentralized Version Control and Mass Collective Collaboration in design-A Case Study of a Web Application Utilizing the Diff Algorithm and Automated Design Generation, *Proceedings of eCAADe 33*, 207-214.

Smith, R.: 2007, "Technical Notes from experiences and studies in using Parametric and BIM architectural software" . Available from <http://www.vbtllc.com/images/VBTTechnicalNotes.pdf> (accessed 26 November 2018).

Woodbury, R.: 2010, *Elements of Parametric Design*, Routledge.

Woodbury, R., Mohiuddin, A., Cichy, M. and Mueller, V.: 2017, Interactive design galleries: A general approach to interacting with design alternatives, *Design Studies*, **52**, 40-72.