Enabling the Architectural Designer to Move within a Graph of Interconnected Decisions: A Case Study Dealing With a Parametric Object

Nathalie Charbonneau, Temy Tidafi
University of Montreal
Canada

Abstract

This paper examines a new way of using computers to enhance the ideation process. Our goal is to investigate methods for implementing digital tools that would help the architectural designer tackle the design process in a different way. Using these tools, the designer would be able to explore a given design space and coherently reuse prior work. This research project aims to develop a methodology that is as flexible as possible for implementing backtracking and forwarding mechanisms that would go beyond the mere notion of undo/redo (characteristic of commercial CAD packages). Instead of tackling the design process as a linear sequence of decisions, we wish to enable the architectural designer to move within a graph of interconnected decisions. We are currently working on a case study dealing with stair design. We plan to evaluate the relevance of such digital tools through working sessions involving potential users. Before doing so, we summarize the questions raised within the framework of this research project.

1. *Introduction*

Research teams have recently reflected on the undo/redo concept, developing new digital tools to give more freedom to the user. They have tackled this problem in different ways and provided a vast gamut of possible solutions. The state-of-the-art research has been reviewed in a previous paper [1]. We decided to conduct experiments based on the use of parametric objects. Such an approach allowed us to work with a well-defined problem and to structure a solution space. It also enabled us to formalize the design process as a network of interrelated decisions. While an approach based on parameters may seem rather restrictive, it enables the designer to explore a relatively broad problem space (depending on the number of parameters implemented). It also enables us to observe how the designer undertakes the exploration process. Tidafi [1] presented more details regarding this aspect.

As Chang and Woodburry [2] noted, in commercial CAD packages, an operation to be undone typically lies well down on the undo stack. When using such packages, deleting previous valuable work is often necessary. As this loss is harmful to the ideation process, we wish to avoid it. With this aim in view, we propose formalizing the design process as a graph of interconnecting various undo/redo stacks. The user would thus explore a graph of interrelated decisions, allowing him to modify previous decisions regarding a specific “chunk” of the design process without interfering with previous decisions in other aspects. He would be free to revise models by undoing and redoing operations in any logically feasible order. Furthermore, while modifying some decisions and maintaining others, the designer may develop new configurations that had not previously occurred to him. We believe that this new way of tackling the design process could potentially enhance the ideation process and result in fruitful cognitive processes.

The designer would be able to tackle the design process as follows: he could explore a network (or graph) of interrelated decisions and, by extension, a set of interrelated architectural forms. He could move backward and forward within this graph, iteratively making decisions and changing his mind. Afterwards, he could pursue the design process beyond the predefined solution space to give free rein to his creativity. We are currently implementing a digital environment to conduct experiments. The challenge here is twofold: on the one hand, we want to provide the designer with a clear mental picture of his decision network. We want him to be able to adequately structure the ideation process to be in a position to compare different options. On the other hand, we want him to be able to communicate the creative process to someone else to help his colleagues understand and follow his idea threads.

2. *Case Study*

The case study in this work considers stair design. This architectural element provides two advantages: i) the design of this kind of 3D artifact requires a wide set of decisions and ii) its configurations range from simple to complicated. Indeed, multiple types of stairs exist. The digital environment's mandate is thus to help the user to determine which type(s) of stairs would fit in the building under consideration. During the design process, the user can make decisions (and later unmake them) to explore a vast gamut of architectural solutions.

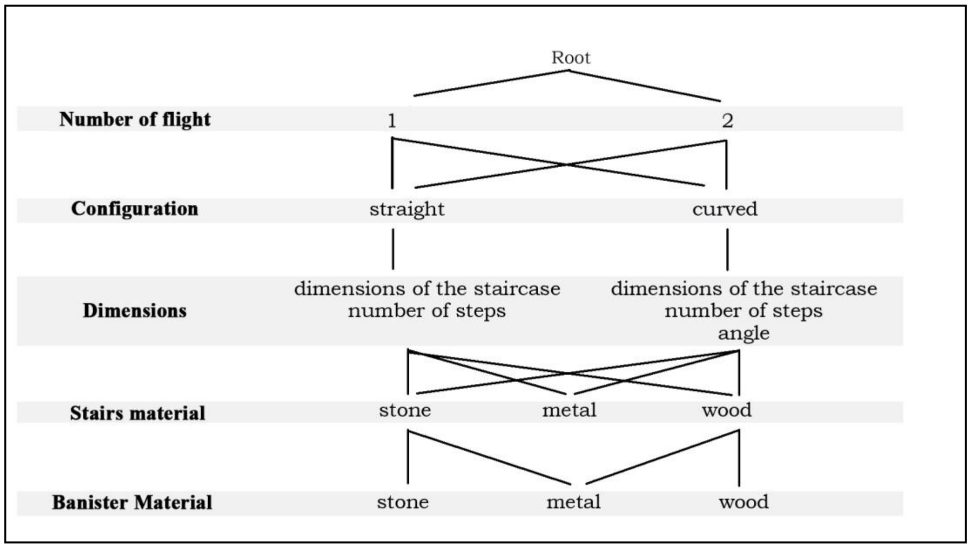
2.1. The structure of the graph

This research project tackles creative process complexity. Before implementing a tool to enable backward and forward movements, our first task was to elaborate the graph formalizing the relationship between the decisions. Within the framework of our case study considering stair design, we considered which decisions are involved and how they are interrelated. To address this issue, we formalized a first provisional version of the graph.

The graph of interconnected decisions is similar to a tree diagram. Nevertheless, it is not a tree structure, strictly speaking. When leaving a node, the user can proceed toward various other 'downstream' nodes (further down the design process). It is important to point out that a given node may be accessed from different 'upstream' nodes, i.e., some 'leaves' may be attached to several 'branches'. For instance, the user must make a choice among several materials (stone, metal or wood), as the stair configuration depends on the material used to produce the components. Let us assume that the user chooses stone. After making various decisions regarding the steps, he accesses the node considering handrail elaboration. He can access (among other nodes) the node implementing metallic handrails. Obviously, this node would still be accessible if metal had been previously selected as the base material of the stairs. Because the graph of interconnected decisions is not a tree structure, we refer to it as a branching graph.

We constructed an initial version of the graph, starting with the most basic decisions faced when designing stairs. These decisions are related to the size of the staircase, the number of steps, the shape of the stairs and the design of the risers and handrails. Up to now, we have been working exclusively with the decisions involved in designing straight stone stairs. The graph presented in figure 1 illustrates the next development stages that we are currently planning.

Figure 1: First version of the branching graph.



As we add more parameters to the system, the number of options proposed to the user will rapidly increase, and the graph will become increasingly complex. It will be of utmost importance to formalize this graph at every step of the research project to keep track of the links between elements and to avoid being overwhelmed by the increasing complexity of the structure. The challenge we face here is to design a branching graph that will not be too complex for the user or programmer to manage.

2.2. Types of nodes

Within the formalized branching graph, there exist two kinds of nodes: 'tweak nodes' and 'junction nodes'. The following sub-sections explain the differences between these two types of nodes.

'Tweak nodes'

When working with a parametric object, the design process consists of assigning values to different parameters. During a given stage of this process, the user can deal with several parameters in any order. These parameters exert effects on the 3D model that are independent of one another (i.e., staircase width and number of steps). Within the system that we are currently implementing, the parameters that are independent of one another are grouped within a 'tweak node'. Such a node enables the user to adjust the 3D model by modifying the values of any parameters belonging to the group, in whichever order the user pleases.

The user can thus evaluate the overall effect produced by changes made to several parameters and make adjustments to the architectural solution. Assigning values to these parameters constitutes a step the user makes toward his goal; it can be seen as a "chunk" of the design process or as a node within the branching graph. When leaving a 'tweak node', the system brings the user toward a 'junction node'.

'Junction nodes'

'Junction nodes' enable the user to choose which branch of the graph he wishes to move toward. He can choose the number of flights, step material, or handrail material, among others. The choice he makes brings him to a 'tweak node' displaying the appropriate parameters.

3. *Exploring the Solution Space*

At the beginning of the process, all values are assigned by default. The 3D model represents the simplest stair configuration. The user progressively assigns more values to the available parameters to define a more complex configuration. As the user makes provisional decisions regarding the object's attributes, the system assigns values to an increasing number of parameters. 3D models of temporary solutions are generated, accounting for the values assigned to some parameters and the identity of the parameters whose values remain undefined.

When dealing with a parametric object, it is not possible to assign values to all parameters at once. The choices made at the beginning of the design process impact the parameters that will be relevant in subsequent phases of the process. Nevertheless, within our digital environment, there is no strict order for assigning values to parameters; the user is not compelled to assign values to one parameter after another, following a predetermined sequence. Within a given 'tweak node', he can assign values to parameters in any order. Furthermore, after exploring several nodes, he can access them again in any order he wishes.

Theoretically, this digital tool gives the designer the possibility of exploring a vast array of solutions. If he finds a suitable solution, he can export the model into a commercial CAD package. He can then modify the model's geometry as he wishes, using the available commands. He can generate plans, sections or elevations. The user can leave the digital environment when he chooses and pursue the design or documentation process by other means.

3.1. Types of movements

Our aim is to develop an environment that will enable the user to move as freely and intuitively as possible. During the exploration process, the system enables the user to make three kinds of moves as described below.

Movements within a 'tweak node'

When the user explores a 'tweak node', he makes adjustments to the 3D model by modifying parameter values. Each decision is recorded and stocked within a list grouping the current value of each parameter within this node. This set of values (or list of numeric values) corresponds to the user's choices at time 'T'. As the user explores the solution space, these lists are chronologically gathered within a global list. For a given chunk of the design process, the system records a first state, a second one, a third one and so forth. The decision sequences are made available for later use. By moving a slider, the user can go backward and forward within a chunk of the design process. In doing so, he gains access to other versions of the model and can compare different options. This slider can be compared to a timeline in which each position corresponds to a state of the model.

As the user explores the current node, the timeline becomes longer. As he explores more nodes within the branching graph, more timelines are recorded.

Movements within the branching graph

As stated previously, the user can move from one node toward another within the branching graph using ‘junction nodes’. So the user can move from one chunk of the design process toward another. He can move either ‘downstream’ or ‘upstream’. In short, movements within a tweak node are analogous to horizontal movements, whereas movements from one ‘junction node’ to another are analogous to vertical movements (as the branching graph is similar to a tree structure).

Exploration of a repository

When the user wishes to interrupt the exploration process, he can save the working session and name the current solution. Two items record this solution: a text file and an image. The text file is a transcript of all lists associated with the current design session. The image is the result of a rendering of the current model (at low resolution). When the user assigns a name to a given solution, the transcription and rendering processes are launched automatically.

When the user wishes to pursue a previous design session, he selects the appropriate item within the main menu, which gives him access to the ‘repository window’. This window displays a set of images associated with names. The images are presented to the user as a collection of large icons. The user can browse through the icons: he can inspect them, select one or delete one. When he summons a previously recorded solution, the associated lists are loaded. A procedure automatically assigns the values contained within the lists to the appropriate parameters, and the 3D model is displayed.

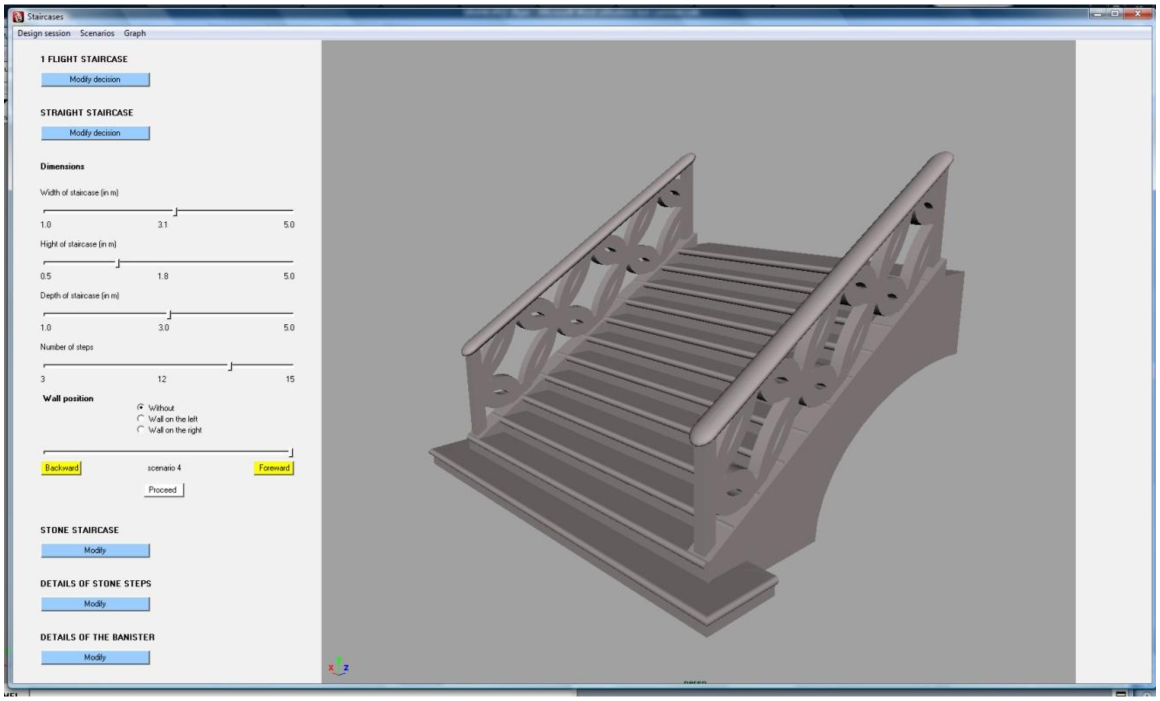
Thus, the user can return to previous work sessions. The repository items are not connected with 3D models but rather with text files. The text files contain lists of numeric values corresponding to each explored node, as well as the current positions within those lists. The system does not record merely the result of the design process but rather the process itself. When the user summons a previous solution, the system brings him to the last option considered during the previous design session. The user can move backward and forward within each previously visited node and re-enact, so to speak, the design process, though he will not necessarily explore the ‘tweak nodes’ in the same order as before. When exploring a previous design session, he can either pursue the design process or backtrack to modify previous decisions[1].

4. *Graphical User Interface*

We are currently developing a digital environment to evaluate the relevance of our approach. The graphical user interface that we are implementing enables the user to modify the configuration of an architectural object (in this case, stairs). The user explores the solution space mostly by pushing buttons, selecting items and moving sliders. When he pushes a button within a ‘junction node’, it brings him to another node (either ‘tweak’ or ‘junction’) that displays other controls. We believe that sliders and radio buttons allow the user to intuitively make adjustments to the architectural solution. He can fine-tune numeric values using sliders. Every time the user interacts with the GUI, he can observe the consequences of his decisions in real time, as the 3D model is automatically updated.
The controls are displayed and organized in a simple way. We feel that it is important to keep the cognitive load as low as possible. We do not want to distract the user with a complex interface or disrupt his thought process. We wish to organize the onscreen layout so favorable conditions allow him to focus on the design process, understand the structure of the branching graph and explore a vast gamut of scenarios.

The GUI contains a permanent window (divided in two) and two temporary windows. The permanent window contains, on one side, the controls (buttons, sliders) allowing the user to change parameter values and/or move from one node to another. The other side of this window displays the current model state. The user can modify the point of view and zoom in and out.

[1] A demo of the exploration process is available. For more details, please refer to “Backtracking within a design process” on YouTube.



Width of staircase (in m) 1.0 3.1 5.0

Hight of staircase (in m) 0.5 1.8 5.0

Depth of staircase (in m) 1.0 3.0 5.0

Number of steps 3 12 15

Wall position [✓ Without] Wall on the left Wall on the right

scenario 4

Figure 2: The graphical user interface.

The ‘temporary windows’ are displayed for a short duration of time to give specific information to the user. The first temporary window displays the branching graph schematizing links between decisions. To find where the designer stands within the branching graph, he can select a menu item that displays a graphical representation of the branching graph highlighting his current position. We believe that this graphical representation can help the user get his bearings. The second temporary window allows the user to access the repository; it displays images of each previously recorded design process. As mentioned above, each image is linked to a button enabling the user to return to the corresponding work session.

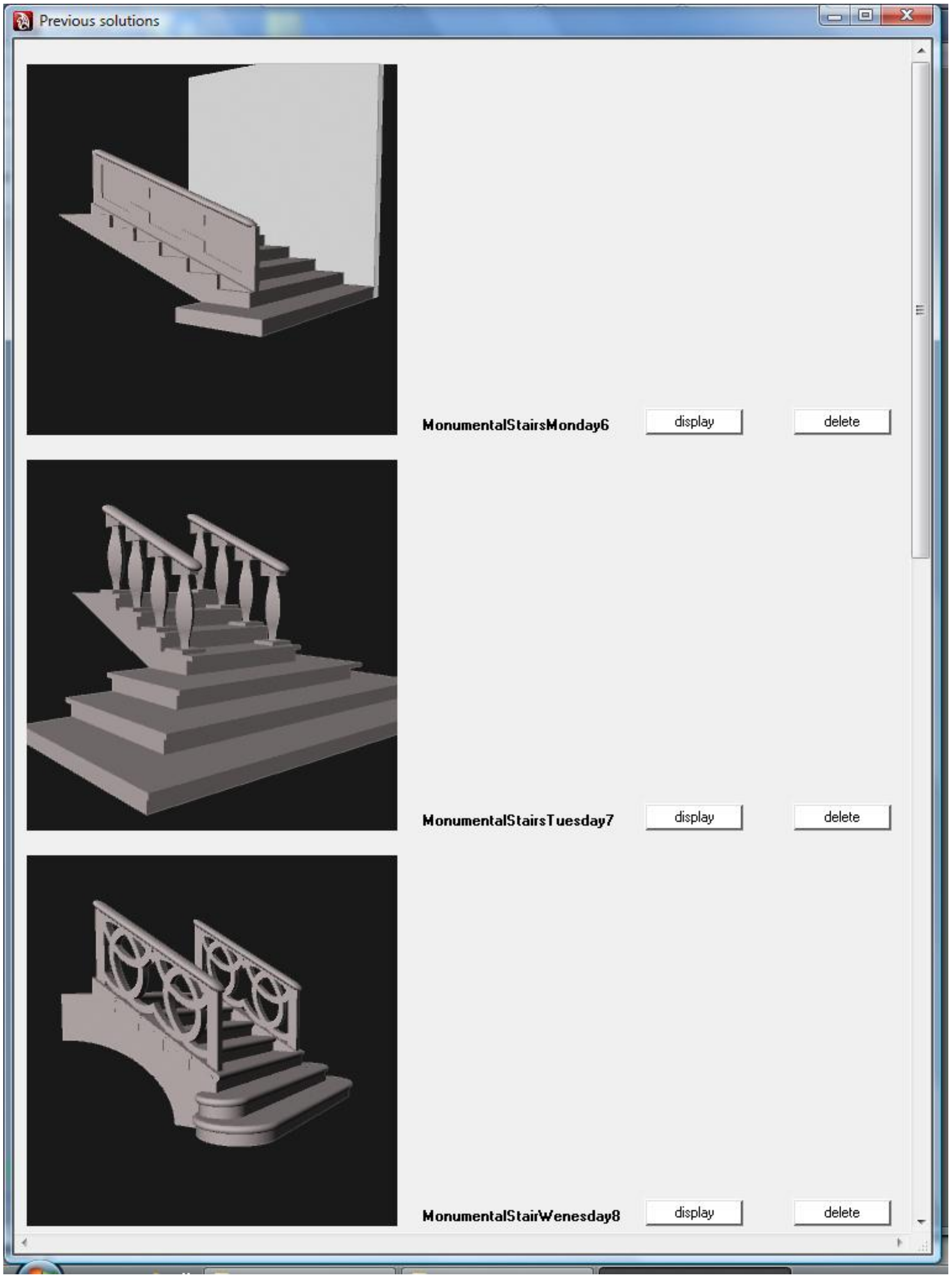


Figure 3: Example of a repository customized by the user

5. System Structure

Every time the user interacts with the GUI, new choices are retrieved from the controls, and the 3D model is updated. As the user explores the design space, the global list (in which lists of parameter values are chronologically gathered) is updated every time a decision is made.

Because the system is based on lists of numeric values, it records and stocks the results of the designer's decisions without overloading the computer's memory. We use lists because we need a 'containing device' that is as flexible as possible. Indeed, when the user begins the design process, it is impossible to know beforehand how many items will be stored; this depends on the length of the exploration process (if the user explores the solution space meticulously or if he is satisfied with a mere overview).

The system is based on object-oriented programming. It uses class inheritance extensively, as 'complex' stairs are based on more generic ones. When generating straight stone stairs, the system calls three classes. It first calls the 'straight stairs' class, which inherits from the 'stone stairs' class. It then calls the 'stone stairs' class, which inherits itself from the 'stairs' class, which is the most generic class.

Each type of stairs contains several components (including steps, handrails, risers, and balustrades). A class corresponding to a given type of stairs has several methods that use different classes for implementing components. The components (steps, handrails, risers) are themselves classes that are inherited from other classes. The class 'step with rounded riser' is inherited from a more generic class named 'step'.

Obviously, the values assigned to these classes' attributes are based on decisions made by the user. These values are retrieved from the graphical user interface, transferred to the main classes and transferred again to the classes for implementing components.

5.1. Tools

To implement a parametric object, we use Autodesk Maya's geometric kernel and the object-oriented programming language Python. We choose Python because the implementation process seems easier than programming plug-ins using C++. Because we are not professional programmers, an interpreted language allows us to iteratively write the code and immediately test the prototype.

Our approach intensively uses Python's functionalities and syntax and invokes Autodesk Maya's command only at the end of the generative process. All classes invoking Maya Embedded Language (MEL) commands are considered 'geometrical classes' and are grouped within a package. These classes are generic and reusable within the framework of other research projects dealing with 3D models. Furthermore, we can replace Autodesk Maya with another commercial CAD package if, later on, we deem it necessary to use a different geometric kernel.
To implement the graphical user interface, we were planning to use Python's Tkinter module, but there are conflicts between this module and Autodesk Maya. Many developers seem to have trouble working with both simultaneously. We thus have to work with MEL, at least for the time being. All classes related to the GUI are grouped within a package. This package, as well as the package containing the geometrical classes, is reusable.
The system contains several modules that are independent of one another. Within the framework of this project, we want to always be able to modify our strategy, rather than being 'held captive' by a given tool or language. During the next stages of the project, this flexibility will allow us to program the GUI with a more effective language (i.e., Java) and use a database to store the parameter values.

The Unified Modeling Language (UML) is a valuable tool that helps us structure our reasoning regarding system implementation. It provides graphical representations of two system aspects: i) The composition of each class: What are the attributes and what are the methods? ii) Links existing between classes: Which are grouped within a package? Which are inherited from which other ones? Which classes call which others? By elaborating and consulting these diagrams, communication can be greatly facilitated among research team members.

5.2. Customization

The user can customize the system in several ways. On the one hand, he can generate a repository of previously recorded design processes. Within this repository, he can add new items or delete those he will no longer use. On the other hand, he can modify the range of acceptable dimensions for any numeric value (such as maximum riser height or maximum number of steps between two landings). When the designer does not follow the previously specified restrictions, the system displays a warning message, and he is compelled to enter another numeric value.
The only way the user can act upon the object's geometry is by interacting with the controls displayed within the panels of the graphical user interface. He can thus modify parameter values. The GUI does not allow him to add new parameters or new types of stairs. As a matter of fact, this kind of restriction is a drawback inherent to most digital environments based on parametric objects.

A higher level of customization would involve modifying the programming code. Most architects do not wish to write programming code. This aversion explains the relative success of programming interfaces, such as that included in Rhino [3]. Nevertheless, one may wonder how much longer architects will choose to stay far removed from the broad field of programming.

Obviously, an architectural designer would not need (or wish) to implement a whole CAD package. He can surely, however, draw advantages from existing geometric kernels and explore how a programming language could help him communicate with them. As early as 1997, Oppenheimer [4] insisted on the importance of mastering programming languages to avoid being confined to commercial CAD packages. This researcher suggested that an architectural designer not actively trying to use digital tools amounts to a renunciation: «If you think of some new way to do or look at things and the software can't do it, you're stuck. So a lot of people think "Well, I guess it's a dumb idea, or it's unnecessary" ». So, architects often make do with restrictive tools. As Rushkoff [5] puts it, «Program or be programmed! ».

We endeavor to extensively document classes, attributes and methods. Inserting comments within the programming code not only facilitates communication among research team members but can also enable the user to modify the plug-in. It does not seem far-fetched to believe that some users may want to modify both parameter values and the parameters themselves. Python is a relatively simple language. Furthermore, it is an interpreted language that allows the novice developer to iteratively implement and test prototypes. In our opinion, Autodesk Maya provides the novice programmer with a good environment.

Nevertheless, a person interested in modifying the code would have to become acquainted with the philosophy behind object-oriented programming. This training could be perceived as an investment, as knowing how to program could prove useful. Clearly, this aspect goes beyond the scope of this paper: we do not plan to try to evaluate to what extent the architectural designer is willing to customize a system using Python, but we deemed it important to mention that doing so is possible.

6. *Prototyping and Testing*

6.1. Questions brought up within the framework of the research project

The ability to move within a graph of interrelated decisions, and to do so in a structured way, may have a positive influence on the designer's behavior. We can surmise that it could help the novice designer get into the habit of developing various scenarios. He would thus become accustomed to comparing different options. In the next phase of the project, we will verify this assumption; we will evaluate to what extent being presented with alternative solutions encompassed within emergent shapes helps a designer during the creative process and encourages discussion among designers.

Now that a first prototype has been implemented, we will organize working sessions involving potential users. We plan to ask several students from the School of Architecture of University of Montreal to use the prototype and observe how they respond to it. We will ask them to give their opinions regarding the relevance of the digital environment, to evaluate the quality of the generated solutions, and to express their concerns or dissatisfaction with the system. Their comments will help us evaluate the aspects liable to impact the user's propensity to use the implemented tool. During the consultation process, we will turn our attention to the following topics.

Using backtracking devices and customizing the tool

We will observe how novice designers use the digital tool. We will determine whether they are likely to experiment with the three types of backtracking processes (backtracking within a node, backtracking toward previously explored nodes and browsing the repository of previously recorded solutions). Furthermore, we will try to answer the following question: Are they likely to customize the system and, if so, in which ways?

Enhancing or hindering the creative process

What benefits are provided by movements within the graph? To what extent does the graph enhance the user's thought process? When decision sequences are played backward, is the designer induced to explore more pathways or options? When the sequences are played forward, does this help the user re-establish the cognitive context of previous sessions? With this digital tool, there is obviously a trade-off between the freedom of movements within the solution space and the restrictions inherent to parametric objects.

We wish to verify whether the advantages of the approach outweigh the restrictions. To what extent does the user feel that using a parametric object enhances or hinders his creativity? This part of the evaluation process will be based on 'metacognition' [6, 7] where users will be asked to reflect on their own cognitive processes.

Managing complexity

When exploring this kind of digital environment, the user must have a clear idea of where he stands within the decision network, what he is doing and where he wants to go. Developing this mental picture could imply a heavy cognitive load. This is the cornerstone of this research project: How can we lighten this cognitive load? To what level of complexity does the user succeed in obtaining a clear mental picture of the graph structure? By what means can we go beyond this limit?

Dealing with the last phase of the design process

Does the user get lost before successfully concluding the design process? If he finds an architectural solution that suits him, does he export it to Autodesk Maya to make adjustments? Does he insert it within the 3D model of the building he is currently designing? Once exported, the model does not maintain its parametric properties. It could not be called an 'intelligent object', an item that is able to communicate with other building components. After assessing the global results, the user must return to the digital environment if he wants to make new adjustments. Is this back-and-forth movement between two pieces of software perceived as disruptive?

To develop a better understanding of these topics, potential users will test the system. More questions will undoubtedly arise during these tests. Users' comments will be documented, and preliminary answers will be recorded in writing during the consultation process. It will be important to keep in mind both questions and tentative answers while developing the next prototype, as this piece of software will be useful if i) it does not distract the user from the design process, ii) it enhances the user's thought process, and iii) the user feels that he has achieved his goal and developed one or more interesting architectural solution(s).

6.2. Observing potential users

We will test the digital environment and assess its usefulness using a two-step procedure. During the first phase, we will conduct tests in which we will ask a user to think aloud, describe the exploratory process and explain the perceived potential effects and drawbacks. During the second phase, we will use a method called 'co-discovery' [8]. We will test the digital tool again with the same user, asking him to bring a teammate. We will observe the pair to answer the following questions: Does the first user re-enact previous design processes to communicate them to his colleague? Do they explore new pathways together?

Observing designers while they navigate within the solution space will be revealing. An observation and analysis of their behavior may help us better understand to what extent the digital environment is useful in its current form and what should be modified to achieve 'better performance'. We will then be able to develop new strategies to provide the designer with improved digital tools. The research process, just like the design process, is cyclical: the researcher develops a prototype, runs tests and makes observations that lead him to restructure the problem, develop new approaches and test them anew. Programmers and designers must work together to bring the process to a satisfactory conclusion.

6.3. Future developments

The consultation process will help us plan the next development phase. We will ask the potential users their opinions regarding the following matters.

- In which directions should we expand the branching graph?
- Would it be helpful to allow the user to act on the branching graph within a working session? That is, when the user considers that he will no longer use a given part of the branching graph and when he deems it unlikely that a given node (or set of nodes) will be visited anew, would it be useful if he could delete it?
- Would it be advisable to work with several small graphs rather than with one large graph?
- Would adding a 'random' button within the 'tweak nodes' enhance the ideation process? When the user presses such a button, the system would automatically apply randomly generated values to the parameters. It would verify whether the values are in accordance with the acceptable ranges and, if not, it would launch the procedure anew.

7. Conclusion

This paper examined a type of digital environment that enables backtracking within a working session involving parametric objects. The proposed approach is based on using parametric objects, as doing so allows the research team to structure a solution space and organize the designer's decisions as a graph. This approach is also based on an extensive use of lists within the programming code. These lists are used as 'containing devices' to store sequences of decisions (symbolized by numerical values). Within the framework of this approach, we attempt to develop a methodology as flexible as possible that can be applied to different scenarios.

The usefulness of such an environment (i.e., its likelihood of enhancing the user's thought process) undoubtedly lies with the developer's ability. The most important challenge in implementing this kind of system is optimally organizing the relationships between classes, whereas the main challenge in developing the GUI is intelligibly representing the graph structure and enabling the user to navigate as freely as possible. It is impossible to know beforehand to what extent the functionalities and representations at the user's disposal will correspond to his cognitive mechanisms. As Waterworth [9] observed, the validity of a GUI is mainly determined by a user's unconscious reactions. From this undeniable fact, we can presume that the digital environment will prove suitable for only some of the potential users for whom it is intended.

It is thus of the utmost importance to implement multiple trial versions and validate them through working sessions involving potential users. These tests will be performed throughout the development process of the digital environment. It is essential to develop the tool not only in accordance with the users for whom it is intended, but also in collaboration with them. We have much to learn from potential users. As Dumas and Redish [10] noted, watching users is both inspiring and humbling.

Our research project is a work in progress. At present, the implementation of the first prototype raises more questions than it answers regarding the relevance of backtracking mechanisms. Our next paper will examine the results from the consulting process and the provisional conclusions obtained from this experiment.

Acknowledgements

This research is funded through the research project GRAND NCE Graphics, Animation and New Media, Canadian Networks of Centres of Excellence.

References

Tidafi, T., Charbonneau, N. and Khalili Araghi, S., Backtracking decisions within a design process: a way of enhancing the designer's thought process and creativity, in: Leclercq, P. Heylighen, A and Martin G., eds., Designing Together: Proceedings of CAAD Futures 2011, ULg, Liege, 2011, pp. 573-587.

Chang, W. and Woodburry, R., Undo reinterpreted, in: Klinger, K. eds., Connecting: Crossroads of Digital Discourse, Proceedings of the 2003 Annual Conference of the Association for Computer Aided Design In Architecture, Indianapolis, 2003, pp. 19-27.

Davis, D., Burry, J. and Burry, Untangling Parametric Schemata: Enhancing Collaboration through Modular Programming, Leclercq, P. Heylighen, A and Martin G., eds., Designing Together: Proceedings of CAAD Futures 2011, ULg, Liege, 2011, pp. 55-68.

Oppenheimer, T., The computer delusion, The Atlantic Monthly, 1997, 280 (1), pp.45-62.

Rushkoff, D., Program or be programmed: ten commands for a digital age, Soft Skull,Berkeley, 2011.

Bouffard, T., Système de soi et métacognition, in : Lafortune, L., Mongeau, P. and Pallascio, R. eds, Métacognition et compétences réflexives, Éditions Logiques, Montréal, 1998, pp. 203-222.

Minier, P., La métacognition selon une approche constructiviste sociale, in : Lafortune, L., Mongeau, P. and Pallascio, R. eds, Métacognition et compétences réflexives, Éditions Logiques, Montréal, 1998, pp. 261-280.

Kennedy, S., Using video in the BNR usability lab. ACM SIGCHI Bulletin, 1989, 21(2), pp. 92-95.

Waterworth, J., Conscience, action et conception de l'espace virtuel: relier les technologies de l'information, l'esprit et la créativité, in : Borillo, M. and Goulette, J.-P., eds., Cognition et création: explorations cognitives des processus de conception, Mardaga. Sprimont, 2002, pp. 119-152.

Dumas, J. S. and Redish, J., A practical guide to usability testing, Intellect, Portland, 1999.