User Stories and Algorithms as Programming and Design Tools

Dr. Tom Elliott Spector, Oklahoma State University

Professor of Architecture, Oklahoma State University Licensed Architect

Mr. Stan Carroll, Oklahoma State University

Stan Carroll, a computational designer, has been practicing architecture for over 25 years, is an award winning public artist, and an educator/researcher. As a result of an ACADIA 2009 workshop on Grasshopper, Carroll transformed his entire design process to center on computational design and fabrication methods. Having recently completed a master degree in the Emergent Technologies Program at the Architectural Association in London in 2014, Carroll has returned to his undergraduate alma mater where he is currently an assistant professor at the School of Architecture at Oklahoma State University. He is currently teaching undergraduate design studios and Computational Foundations. In addition to his teaching and researching, he also continues his private practice pursuing large public art commissions.

Prof. John J. Phillips, Oklahoma State University

JOHN PHILLIPS, a registered engineer and associate professor of architectural engineering, practiced as a structural engineer for nine years before returning to his alma mater to teach at Oklahoma State University. He teaches undergraduate and graduate courses including Statics, Analysis I, Foundations, Timbers, Steel, Concrete, Steel II, Concrete II, Steel III, Concrete III, and in the Comprehensive Design Studio.

User Stories and Algorithms as Programming and Design tools

Introduction

A not uncommon sentiment from dissatisfied software clients is: “yes you gave me what I asked for, but it wasn’t what I wanted.” This dissatisfaction occurs primarily due to the difficulty of defining the desired experience of the software in hard performance numbers. Asking clients to specify in advance exactly how the requested software is to perform is a notoriously faulty strategy. Usually, their responses only reflect what a client has already seen, known and become comfortable with or else the responses are hopelessly vague on the order of: “Give me something I’ve never seen before.”

The obstacles for innovation in such situations should be obvious. This situation in software design has its direct analogue in architectural engineering design. The architect or engineer receives a program with space sizes and relationships, perhaps as well a statement of organizational goals, and then is expected to turn these parameters into a design concept. The large gap between the demands of basic functionality and the evolution of an artistically unified design response make the conceptual and schematic phases of the design process often appear opaque, mysterious, and less than fully rational. Data enters a black box, a design emerges. For engineers, heuristic aids for initial sizing of structural elements all-too-quickly become immutable paramaters.

In response to this problem of the subjectivity of experiential design qualities in the software industry, a visualization technique called “user stories” has evolved to replace such cumbersome elements as usability test reports and lengthy UI specifications within the agile design work environment. (Sy, 131) The agile environment forsakes comprehensive software design for shorter design “sprints,” which, when paired with frequent user feedback, results in products which gradually come into useful existence instead of all at once. (Agile Manifesto) The user stories become, in essence, the basic unit of design. We have applied this technique to a research, programming and design methodology in design studio that, when conjoined with an algorithm to test the design, produces satisfactory results.

Methods

In software design, a user story describes how a user wants to interact with the software—what he or she hopes to get out of the product as well as the experience of using the product—or with a component of the software. Typically, different sorts of users with different user goals are identified. The essential features of a good, actionable user story are:

- It adopts an end-user’s viewpoint—it enters the user’s psychology in a sustained way
- It is not a single action, but a set of connected actions activated by a goal

- It is testable: criteria for successful implementation are established to determine success or failure of the design. Those criteria for success may take the form of constraints (such as egress requirements).
- The story does not contain technical details. It is phrased in experiential and qualitative terms.
- A story has to be the right size. Too large of a story is called an "epic," which must be broken down into manageable stories. Too detailed a story is merely a scenario—an image, a single action.
- Stories must be devised that have significant design implications.

And it is developed in 3 steps (Figure 1):

- A **conversation** with the user types (or their surrogates) concerning the story
- A note **card**-sized written description—the story
- Criteria for **confirmation** or success. (Jeffries, 2001)

For a comprehensive studio course—the project to house a local theatre company—the methodology of user stories was combined with an algorithm written in Grasshopper and visualized in Rhino to test design solutions for seating arrangements in the thrust theatre box. First, students were tasked to interview theatre staff and patrons to develop their user stories. While each story itself is qualitative, the success criteria should be written in such a way that they can be stated in quantitative terms if proposed design solutions are to be tested prior to actual implementation. One characteristic design goal to emerge from these interviews—that the space provides an intimate theatrical experience for the theatre-going patron—was turned into quantifiable success criteria. (Figure 1)

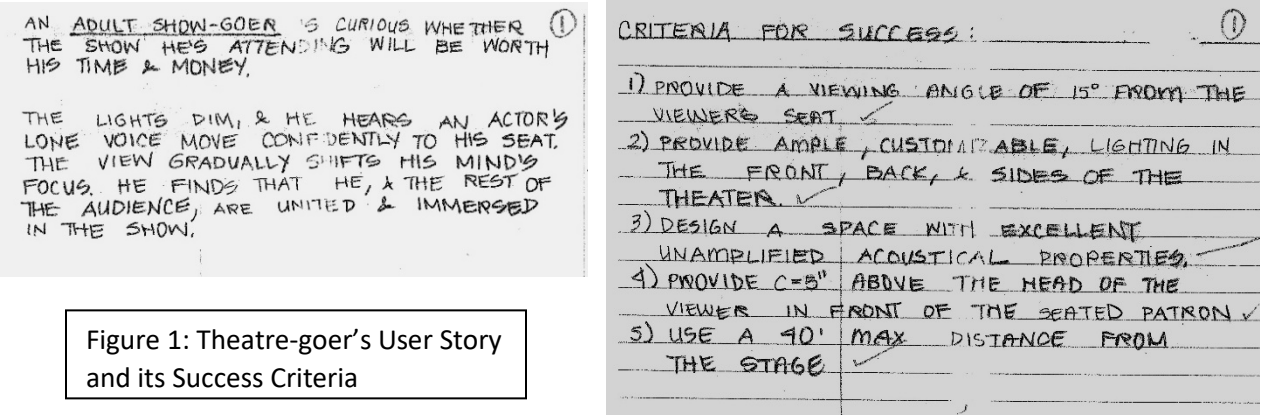


Figure 1: Theatre-goer's User Story and its Success Criteria

To sustain the desired quality of intimacy with the onstage action we isolated three quantifiable success criteria—a maximum desirable distance of seating from the stage, a maximum desirable spread, or viewing angle, for the seating, and the minimum "C" distance—defined as the height in inches between any given seatback and the seatback immediately in front. The "C" distance, ultimately, determines the rake. The algorithm to test proposed solutions was envisioned as an optimization problem requiring iterative attempts to make proposed theatre seating layouts meet the performance requirements. Each success criteria embodies trade-offs for optimization. For example, a shorter maximum allowable distance from the stage will compromise the width of the

rows or else the number of seats the theatre can accommodate. Too narrow an allowable viewing angle will impact the distance from the stage. High “C” numbers, while increasing viewing desirability, complicates providing access to the seats.

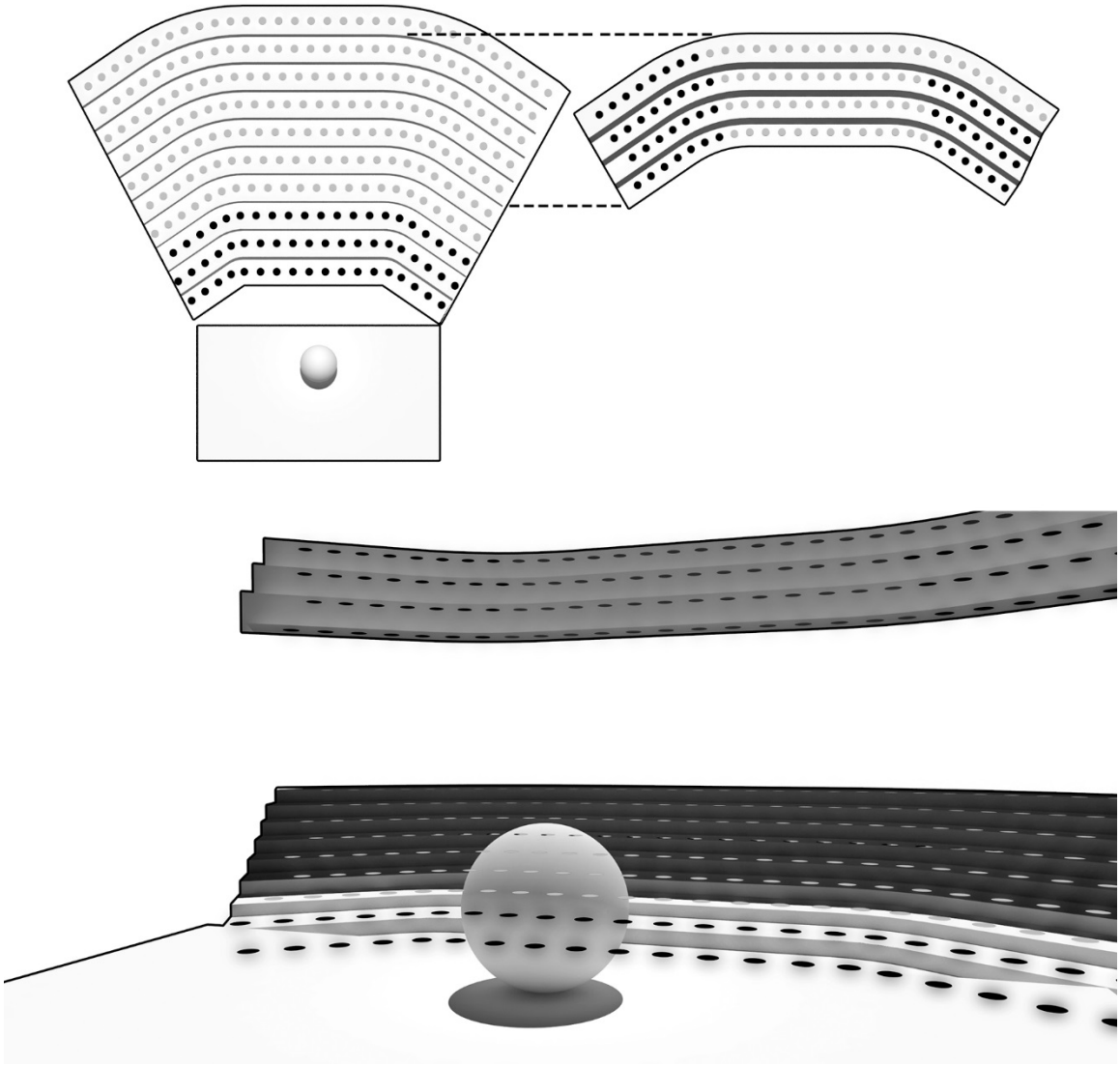


Figure 2: An image of the Rhino base model that students could adjust to reflect their proposed seating layouts.

These 3 parameters were turned into an algorithm in the Grasshopper program which was used to evaluate the geometric properties of students’ theatre box designs in Rhino (Figure 2). Theatre seats that failed to meet one of the criteria were highlighted as triangles. This information formed a feedback loop used to direct the redesign of the theatre box (Figure 3, 4).

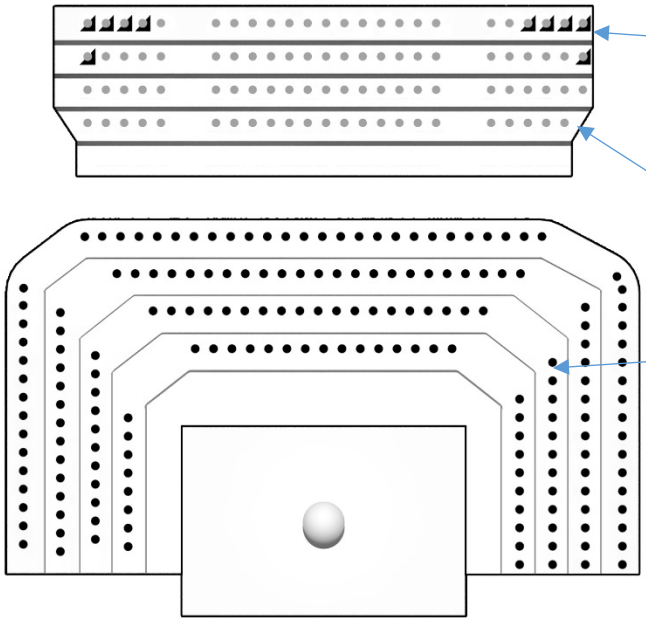


Figure 3: a tested seating configuration with graphic legend

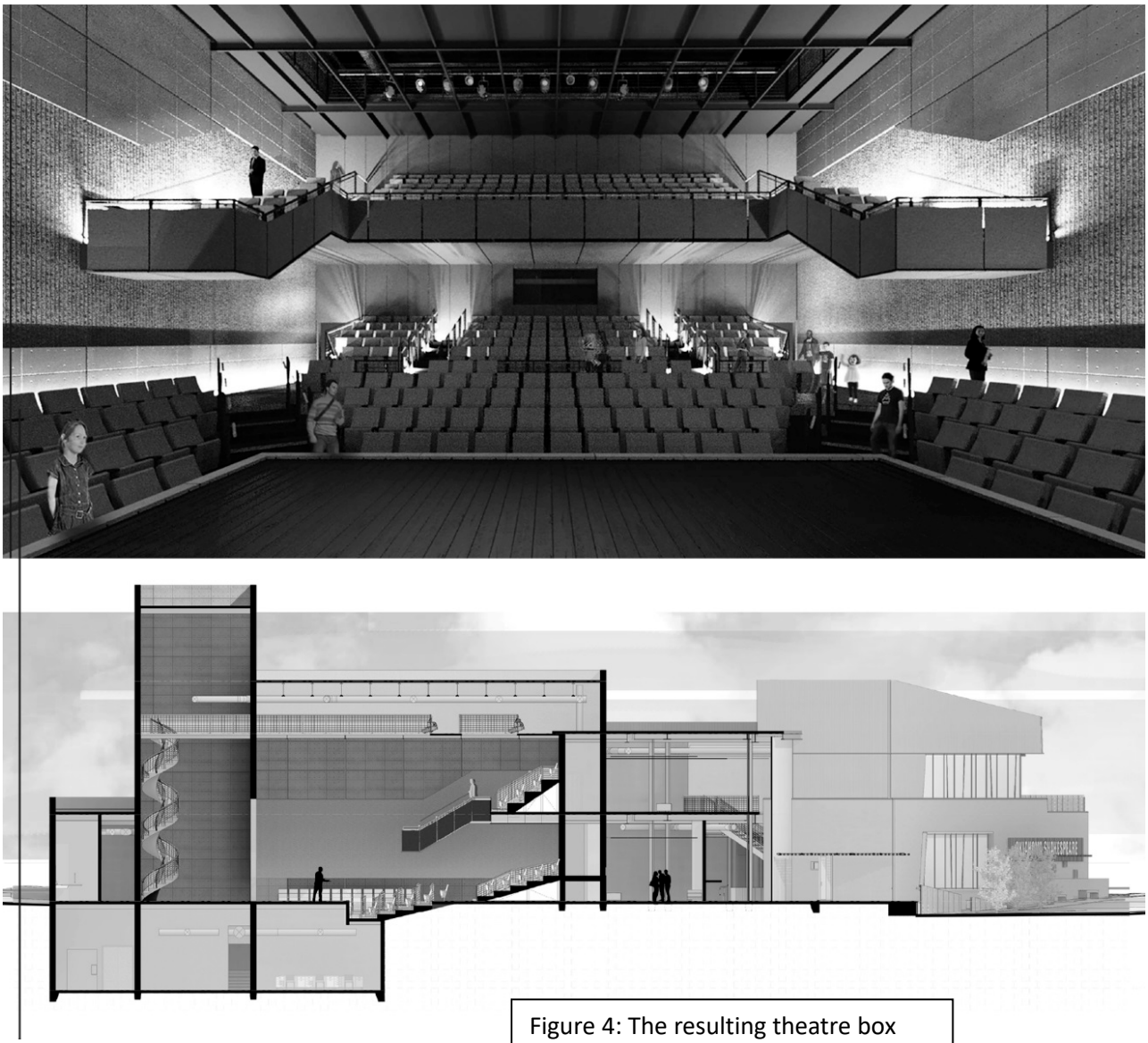


Figure 4: The resulting theatre box and section through the theatre.

In the following year, the capstone studio project was a combination food pantry and homeless youth shelter for a provider in a nearby urban center. This project presented the prospect of user stories from several perspectives—that of the food pantry clients, the organization's skeletal staff, homeless teens, and the volunteers that do much of the work of unloading, sorting packaging and stocking the pantry shelves.

All students were required to volunteer at the food pantry to experience the volunteers' perspective. This also gave them a chance to interview regular volunteers for their user stories. Those stories characteristically expressed a desire for a maximally efficient operation combined with the opportunity to enjoy time spent with fellow volunteers. Success criteria for the food pantry receiving and distribution operation were modeled as a straightforward efficiency equation in Grasshopper (adjusted for a frequency assessment for the different types of grocery products received and distributed) applied to a simplified plan view in Rhino. Testing the model based on the total number of linear feet required to complete the various receiving, storage and distribution operations resulted in a unitless scoring of each student's design (Figure 5). Depending on the score, the revised design could be imported back into the Revit model for incorporation into the overall design (Figure 6). Scores were posted on a whiteboard for all to see (Figure 7).

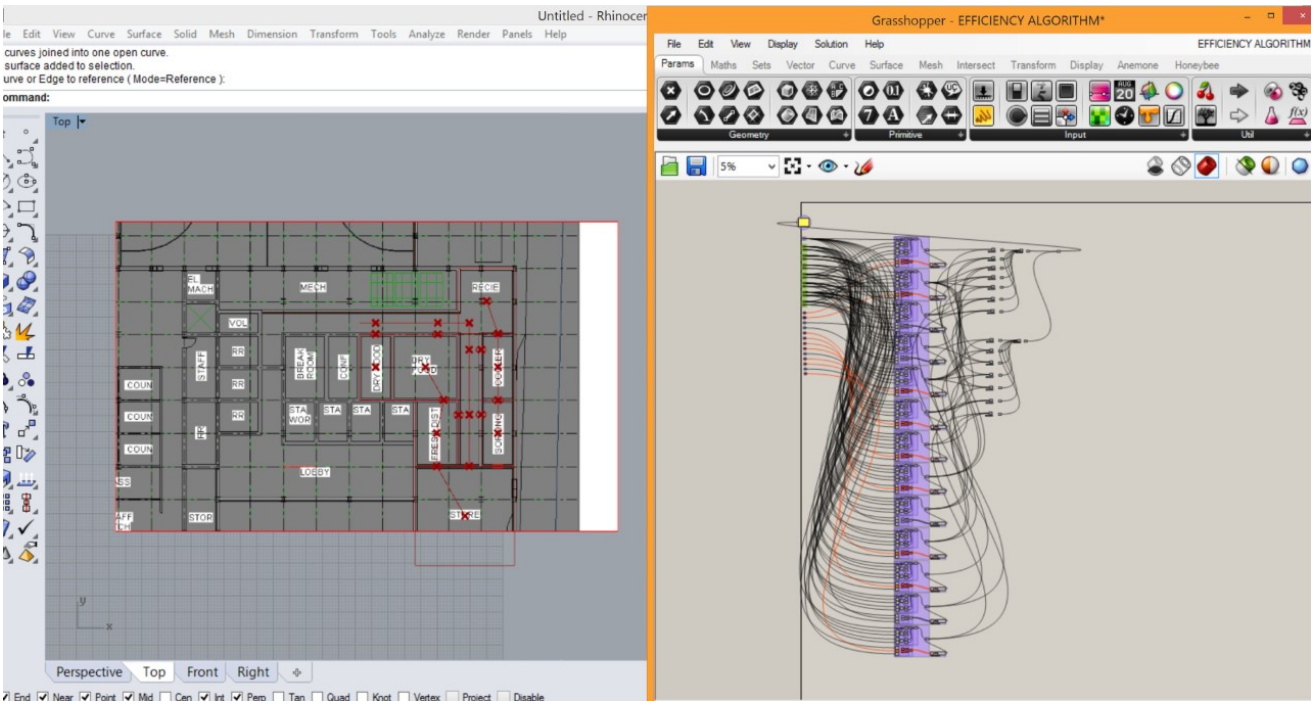


Figure 5: The Rhino model and Grasshopper algorithm

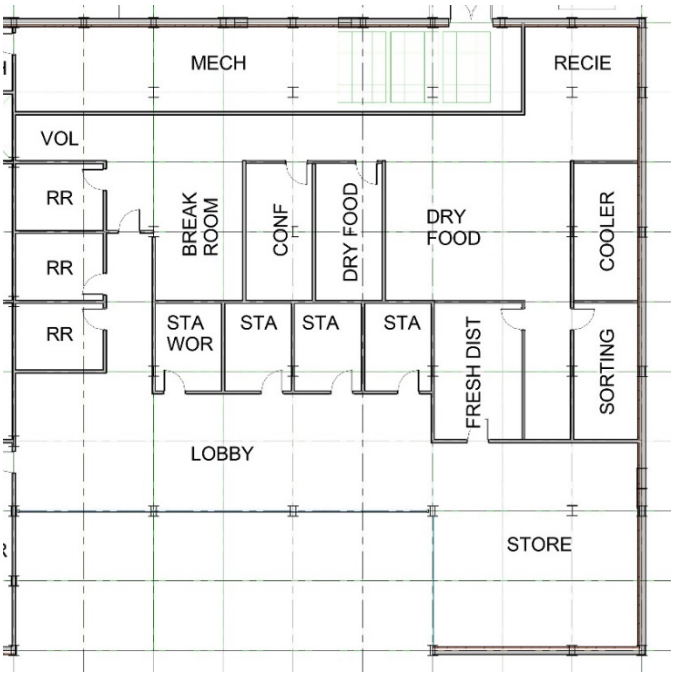


Figure 6: The food pantry area designed in Revit. The Rhino model of the Revit file for this area records the design nodes for receiving, storing and distribution of the foodstuffs. These notes are analyzed in Grasshopper to determine a score.

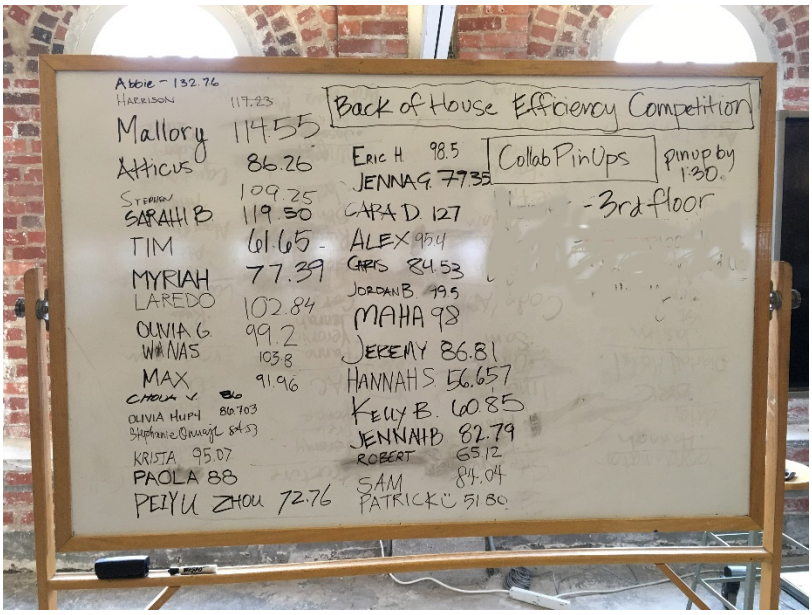
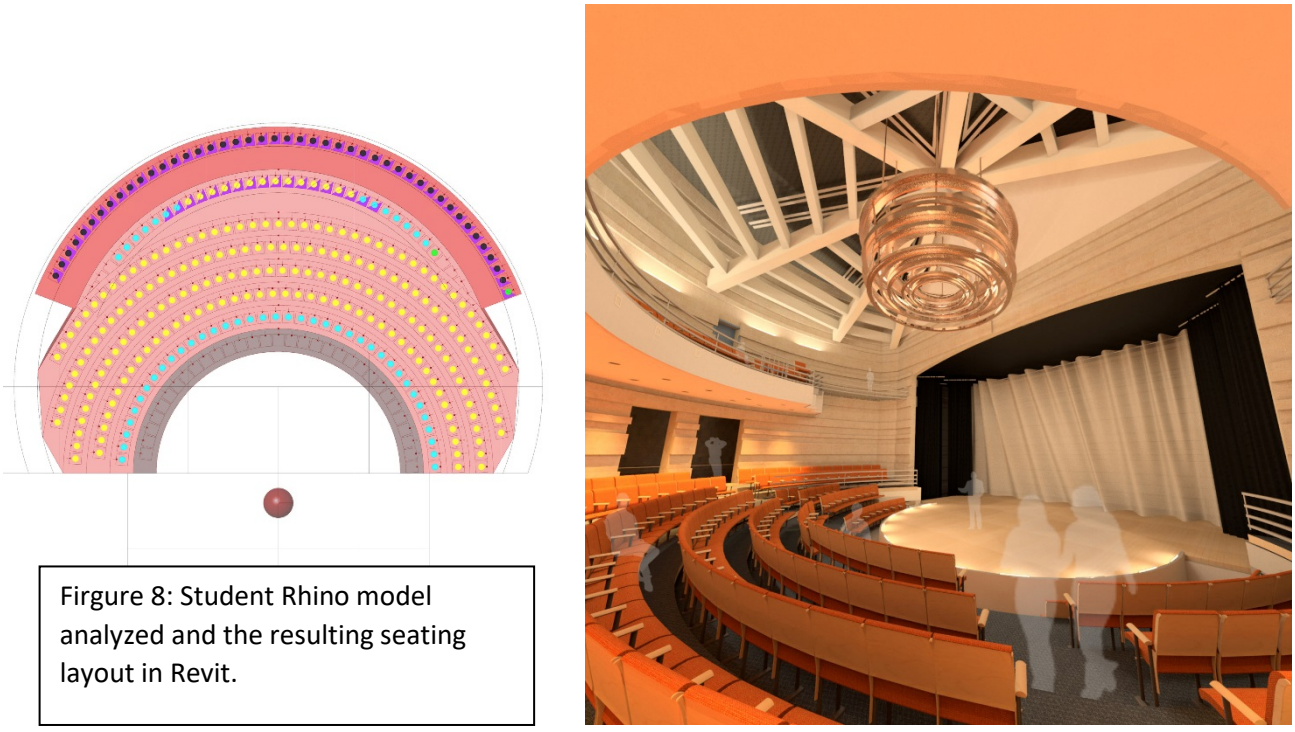


Figure 7: the efficiency scoring

Efficiency scores ranged from Abbie's high of 132 to Patrick's low of 51 (Lower is better). Anything below 100 was considered adequately efficient, though several students with adequate scores repeatedly sought to improve their scores through redesign.

Results

User stories were first tried in studio solely as an alternative to the standard research and programming methods. Students reported that once the programming phase was complete, however, the user stories provided them little incentive to revisit their designs. But once the user stories were combined with the algorithm-based feedback mechanism the next year, the link between the user stories and the end design was made tangible. A well-designed algorithm has certain advantages over frequent user feedback envisioned in the agile work method. For the design of the theatre box, the algorithm highlighted for elimination the bad seats. Design critics did not have to argue with students that certain seating configurations were poor, or difficult, or resulted in an excessive number of undesirable seats. Instead, the Rhino model conveyed this information. Furthermore, this process allowed students to both quickly try out different seating configurations and to visualize the impact of their decisions on audience experience. The timely feedback provided by the algorithm tied to the Rhino model resulted in uniformly improved theatre seating configurations compared to similar projects in years past (Figure 8).



Firgure 8: Student Rhino model analyzed and the resulting seating layout in Revit.

The end result from the food pantry design exercise: students with poor efficiency scores were motivated to revise their designs according to the priorities established for efficiency within the algorithm. Furthermore, the algorithm itself was able to be reviewed by the client for modification. For example, the algorithm weighed the storage operation for certain food types more heavily than for others that were not accessed as often. When the relative weights were

discussed with the client, adjustments were able to be made to better reflect the operations of an average week.

Testing the proposed design against the criteria set up in the algorithm allows a feedback loop to be established from programming (in the form of user stories) through the design proposal, testing the proposal (in the Rhino model in conjunction with Grasshopper) to, finally, identification of designs that do not meet, in whole or in part, the success criteria established in the user story. We are encouraged by these results and intend to continue incorporating this technique into future studios. If the user stories and their associated success criteria are developed in consultation with clients, then the resulting designs can emerge from the black box as explicit responses to the user stories.

Conclusions and Further Study

User stories are not without their detractors in the agile design community, but we believe that there are aspects to the employ of user stories within the agile concept we have yet to employ that may well help engineering and architecture students with the always-vexing problems of scheduling and time management, and of assigning responsibilities within a group project. (Sauro, 2016, Hudson, 2013, Wright and McCarthy, 2010, Nielson, 2001) Here we have in mind in particular the concept of sprints. If design tasks can be clearly delineated, or "chunked,"—either by professors, or better yet, within the team—then creating interim deadlines for task completion and assigning responsibilities for sub-tasks can be greatly enhanced. We could anticipate that, since sub-tasks within engineering and architecture do not lend themselves to be as fully componentized as they do within software design, time will need to be made for synthesis and reconciliation of the various elements. In addition to the concept of sprints, the organization of large of amounts of programming data collected in direct consultation with users in "affinity diagrams" is another relevant strategy of the agile work environment with potential applicability to architectural design.(Beyer, 31) An affinity diagram has some of the characteristics of the familiar bubble diagram, but rather than address space adjacencies, it enables designers to bring together use concepts that might not otherwise be obvious. It also differs from the bubble diagram in that it is not the result of classification according to already-programmed use categories. Instead, it is the result of observations that themselves tend to create categories.

With these successes and ideas for future improvement in mind, however, it should be noted that the context out of which software design emerges differs in significant ways from that of architectural engineering. The agile environment is specifically conceived to both bracket ongoing change so that individual features of the overall product can be designed and to parcel out responsibility so that the development team is only responsible at any given time to reach a state of completion for individual design tasks drawn from a pre-existing backlog of tasks, that when completed, will resemble the shape of the final product. (Beyer, 6). It's as though we could produce the theatre box or the food distribution operation as a discreet plug-in component of the overall building. But of course if things are not quite this neat in software and HCI design, they are doubly messy when it comes to the design of buildings. This difference in context may help explain the paucity of scholarly writing in this subject. As long as the user stories worked with

here are understood as providing a process to optimize individual features, but no roadmap for how all the various features of a building are to integrate, then the technique has not exceeded its applicability for architectural engineering.

References

Agile Manifesto http://agilemanifesto.org/principles.html

Beyer, Hugh. *User-Centered Agile Methods.* San Rafael, CA: Morgan & Claypool, 2010.

Cohn, Mike. *User Stories Applied for Agile Software Development.* Boston: Addison-Wesley. 2004.

Hudson, William "User Stories Don't Help Users: Introducing Persona Stories," *Interactions,* November-December 2013.

Jeffries, Ron. (2001). "Essential XP: Card, Conversation and Confirmation." *XP Magazine.*

Nielson,, Jakob "First Rule of Usability? Don't Listen to Users," 2001, https://www.nngroup.com/articles/first-rule-of-usability-dont-listen-to-users/

Sauro, Jeff. "The Challenges and Opportunities of Measuring the User Experience," *Journal of Usability Studies,* v12 n2, 2016. 1-7

Sy, Desiree. "Adapting Usability Investigations for Agile User-Centered Design," *Journal of Usability Studies,* v2n3, 2007 112-132.

Wright, Peter and McCarthy, John. *Experience-Centered Design: Designers, Users, and Communities in Dialogue,* Morgan & Claypool, 2010.