ServiceSketch

A thesis submitted to the School of Design, Carnegie Mellon University, for the degree of Master of Design in Interaction Design

Norman Lau, MDes in Interaction Design, 2011

Suguru Ishizaki, Thesis Adviser

TABLE OF CONTENTS

ABSTRACT

ServiceSketch is a collaborative tabletop tool for service design. It was developed to address some of the challenges designers face when developing service systems, including the dynamic, intangible nature of service and the complexity of coordinating multiple stakeholders over time and space. The concept for the tool draws from literature on service design, tangible user interfaces, and co-creation. It was also informed by user research sessions with graduate design students.

The interface of ServiceSketch consists of a large multi-touch surface display that reacts to finger touches and a provided set of physical objects. Both the hardware and software development of ServiceSketch are described in this document.

ServiceSketch was evaluated with groups of graduate design students who were asked to perform small group service design activities using the tool. These sessions showed that ServiceSketch was successful in supporting common service design processes and even inspired many participants to suggest possible future developments for the tool. ServiceSketch also seemed to encourage a playful, collaborative approach to service design. The results of the project hint at the possibilities for a new breed of service design tool, one that focuses on facilitating conversations about service through an engaging, interactive medium.

INTRODUCTION

Services are everywhere. They structure our daily interactions with our environment and with other people. Examples can be drawn from health care, education, retail, transportation and plenty of other sectors. But it's relatively recently that designers have realized the active role they can play in giving form to services.

Service design isn't necessarily about any one single designed product or artifact. Rather, designers are working with customers, service providers, clients, and other stakeholders to plan and coordinate a set of service activities. Service designers might be called upon to design anything from the process of getting a coffee at the café to the patient experience at a hospital.

Perhaps more so than other design disciplines, service design is an act of facilitation. Designers must use various tools and methods to open dialog between stakeholders and themselves, working to reach a shared understanding of the situation.

Usually, the tools designers use to mediate this process are static diagrams mostly generated by the designers alone. This thesis project considers the possibilities for a new type of service design tool that captures the dynamic nature of service. I use the lens of tangible user interfaces to examine more engaging forms of interaction with design tools. I also throw focus on the co-created aspect of services and attempt to support a more collaborative approach to service modeling.

INSPIRATION

Service Design and Blueprints

The concept of service design grew out of the field of marketing. Shostack distinguished the development of services from traditional product marketing by highlighting the intangible nature of service. Unlike products, "[a] service is experienced. A service cannot be stored on a shelf, touched, tasted or tried on for size" (Shostack 1977, 73).

To address these challenges, Shostack introduced concepts that remain fundamental to service design today. The first was the understanding that people can experience the same service in different ways. For Shostack, this meant the use of techniques from psychology and sociology to identify the most common experiences of the service, or "consensus realities" (Shostack 1977, 76). Shostack also stressed the importance of tangible evidence of a service, physical clues that communicate the quality of the service to the customer (Shostack 1977, 78). Examples of tangible evidence could include the physical environment in which the service takes place or the dress of the service employees.

But perhaps one of the most lasting contributions by Shostack was the early development of service blueprinting (Shostack 1984). A service blueprint is a graphical tool to help visualize a service across time. In form, Shostack's blueprint resembles a flow chart mapping out the actions that are taken in the course of the service. A key feature is the division of the chart by the "line of visibility." Items above the line are actions that are seen by the customer and those

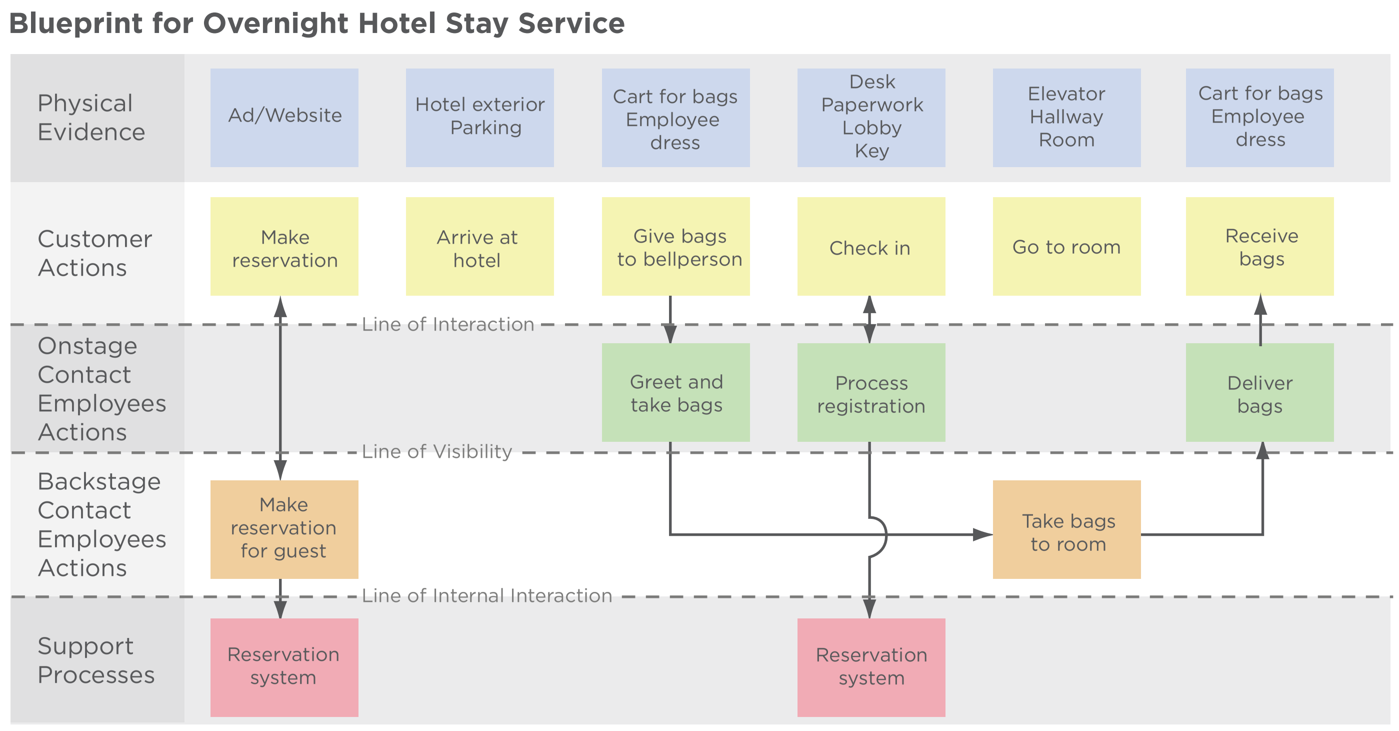

Figure 1. An example of a service blueprint. Adapated from Bitner, et al. 2008.

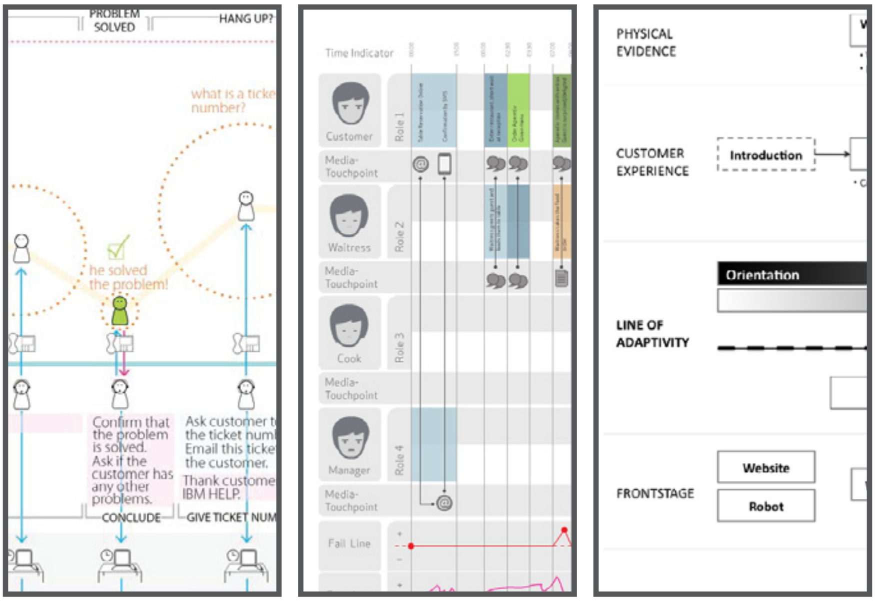

Figure 2. The service blueprint has been adapted by many service design researchers. Images from Spraragen and Chan 2008, Polaine 2009, and Lee and Forlizzi 2009.

below are not seen by the customer, but necessary for the service to operate. This simple separation fosters awareness of components of the service that may have been neglected otherwise and promotes a more holistic view of the system.

Blueprinting remains popular as a tool for service design. A later study by Bitner et al. emphasized a user-centered approach to the blueprint, a strategy familiar to designers (Bitner et al. 2008). Customer actions are prominent near the top of the diagram with employee actions mapping to them below (see Figure 1). Bitner et al. give a number of case studies in which blueprinting was used innovate services and imbue them with a customer focus.

Over time, still more iterations of the blueprint have emerged, each adapted for a particular aspect of service (see Figure 2). Spraragen and Chan used their blueprint to highlight the role emotions play in a service (Spraragen and Chan 2008). Polaine presented a re-tooled version with his Blueprint+, putting more of a focus on perception of time and media touchpoints (Polaine 2009). Lee and Forlizzi looked at modelling adaptivity and change over time with their blueprint in the context of robotic services (Lee and Forlizzi 2009).

The development of the service blueprint as a tool charts an interesting evolution of service design as a whole. We see the field's continuing struggle with the issues Shostack outlined early on: the intangibility of service and the multiplicity of stakeholder perspectives.

Tangibility and Tabletops

Humans have a range of tactile senses that allow rich physical experiences of the world. And yet, most of our interactions with technology are still bound to flat rectangles of space. How do we begin to break our relationships with technology away from the screen?

Ishii and Ullmer provided a vision for the movement toward tangibility in digital interfaces called "Tangible Bits" (Ishii and Ullmer 1997). Building from larger themes of ubiquitous computing, they focused on how the use of tangible user interfaces (TUIs), or interfaces that couple physical objects with digital information, might provide richer experiences.

One of their research prototypes, the metaDESK, experimented with physical analogues of typical GUI elements, like lenses in place of windows or physical objects in place of icons (Ishii and Ullmer 1997, 237). These combined with a tabletop display to create a platform for tangible user interfaces. It is early explorations like the metaDESK that helped inspire the potential of digital interactions beyond the screen.

Building on the theme of tables, Kaltenbrunner and Bencina developed an open-source software engine for multi-touch surfaces called reacTIVision (Kaltenbrunner et al. 2007). Based on computer vision, reacTIVision allows anyone with an appropriate web camera and projector to setup their own large-scale touch display with object tracking capability. The project originated as a component of the reacTable, an electronic musical instrument with a tabletop TUI (Jorda et al. 2007). But since then, others have used the software toolkit to create their own multi-touch applications and TUIs.

However, while the necessary technology to realize the vision of "Tangible Bits" becomes more advanced and widely available, many TUI projects are marked by an experimental quality. They are often treated as proof-of-concept projects, with more focus on the usability of the interface and less focus on its usefulness. By working towards a specific application, a tool for service designers, I hope to provide a compelling example of a practical use for tangible interfaces.

Co-creation and Play

Design is a collaborative activity. Designers must be able to empathize with multiple perspectives, engaging in activities of co-creation with other stakeholders. Sanders explored the idea of co-creation through the use of generative toolkits to engage everyday people in design (Sanders 2000). Each toolkit is designed to be "simple and ambiguous so that the participant can project his or her own aspirations onto the artifacts that they make" (Sanders 2000, 5). These toolkits took various forms; some were 2-D or 3-D modeling kits, others were storytelling devices. But they all shared a focus on allowing anyone to participate in the design process alongside designers.

In another take on co-creation, Habraken and Gross likened collaborative design to playing a board game (Habraken and Gross 1988). Both activities involve the cooperation of multiple people with diverging goals who have agreed to follow certain rules and constraints to proceed. The researchers explored these parallels by developing a set of concept design games. Both the development and enactment of these games, played with simple physical objects like nails, washers and clothespins, served as studies of design concepts such as physical and territorial organization.

While Habraken and Gross presented concept design games as a research tool rather than a tool to design with, others have since then applied the idea to practical design situations (Brandt and Messeter 2004; Johanssen 2006). In these cases, design games were used to facilitate collaborative design work between different stakeholders, while promoting a user-centered focus. These studies seemed to show that the structure provided by a game helped designers to generate new design possibilities. In addition, when participants entered into a shared agreement on the rules, barriers to open dialogue, such as corporate hierarchy or unequal power relations, were mitigated. Everyone's opinion had equal standing when playing the game.

In an interesting blending of Sanders' generative toolkits and the concept of design games, LEGO developed a method called LEGO Serious Play that involves the use of LEGO pieces in a structured building activity to facilitate dialogue between people within businesses and organizations (The LEGO Group 2010). The process of LEGO Serious Play is broken into three phases: the challenge, building, and sharing. The challenge is a building exercise posed by a facilitator and is formulated to spark reflection on the issues being explored. The building phase involves each of the participants creating a LEGO model that represents their own thinking on the challenge. Finally, the sharing phase has the participants tell the story of their model and explain its meaning.

There exists a fascinating analog between the activities of collaboration and play. Both benefit from rules of engagement, a shared language to encourage collaboration and inclusivity. At the same time, a playful attitude can lead to creative experimentation and a re-writing of the rules. For any collaborative design tool, the value of play shouldn't be underestimated.

Design Tools

A common thread can be found in the literature I was researching. In each topic I examined, there was a concern for tools. My service design reading gravitated towards the use of the service blueprint as a tool to focus the development of services. My exploration of tangibility revealed a variety of physical computing gadgets, tools for everything from work processes to music making. Finally, the literature about co-creation described collaborative design tools like Sanders' generative toolkits or LEGO Serious Play.

Tools are particularly interesting for design because they are the designed artifacts that enable and guide people's activities. We are often charged with designing and developing the tools for other people in their work and play. We apply our craft and knowledge of interaction to shape these tools and to hopefully shape experiences.

I saw an opportunity to combine the threads of my research by creating a tool for service design. This tool would be for designers themselves, to help them grasp the complexity of service. As an emerging field, service design provided a prime opportunity to create a novel tool. In contrast with the service blueprint, I wanted to use my research of tangible computing and co-creation to develop a more dynamic, collaborative service design tool.

EXPLORATION

For initial user research, I ran three short sessions with graduate design students. In each session, the group was asked to design a small banking service. The challenge outlined various constraints and service touchpoints. The participants were allowed to use whatever materials they wished, but I made a point of providing a number of unconventional physical objects to see how the design process would be affected. These objects included small wooden blocks and various clay figurines (see Figure 3).

I had two goals in mind for the activities. First, I wanted to get a broad idea of how designers approach the design of a service. These students were not professional service designers, but many had studied service design and had experience as practicing designers. Observation of their methods and the discussions might reveal their priorities as service designers.

Second, I deliberately brought physical objects into the design session to see if and how participants would use them as tools for design. The designer's affinity for post-its on a whiteboard speaks to a preference for tools we can grasp and change. We can easily re-arrange the post-it notes around with our hands, using spatial relationships to organize the information. I felt that the introduction of a set of three-dimensional objects might reveal some interesting directions for my exploration of a tangible service design tool.

Findings

The participants usually began by categorizing user groups and types of

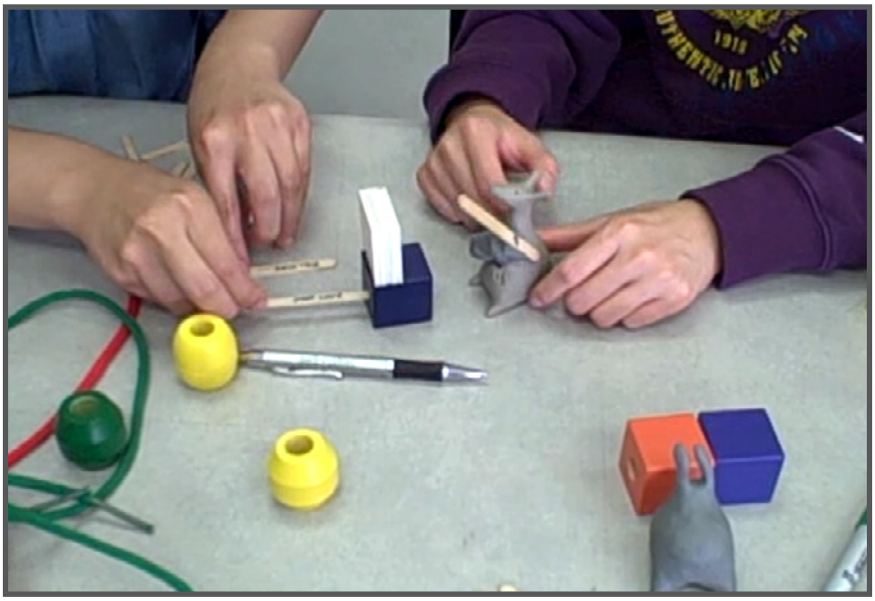

Figure 3. During my user research activities, participants modelled services using various physical objects.

services to help them to understand the situation. Most of the groups then used strategies to narrow their focus, usually by scoping the design to a particular type of user or service. Categorization and scoping are fundamental capabilities for any kind of design and service design is no exception.

The participants made heavy use of external representations both to communicate between each other and to sort through their own reflections. They used a lot of writing and diagrams to record their progress, complemented with use of the physical objects as representations for various entities or components of the service. Clearly, designers rely on externalizations of their thoughts to communicate and move forward on the design.

The clay figures and wooden blocks lent a certain playfulness to the design activities. Participants were constantly touching and moving objects as they spoke, often using gesture to help communicate what they were saying. They used the clay characters to enact design situations. There was obvious delight in some of the forms of the objects (interestingly, in all instances, the same clay penguin figure was used to represent the customer of the service). These findings showed there was definitely a richness to explore in the use of a tangible tool for service design.

MAKING

It is important to note that the ideas for my project did not arise from literature and user research alone. Even before I had formulated the specifics of my thesis as a tool for service design, I had the opportunity to experiment with the construction of a multi-touch table. The experience of building it and developing small multi-touch software applications helped me understand the table's potential for various applications and fed into the development of my thesis.

This point emphasizes the importance of making to my design process. The affordances of my physical prototype led me in certain directions for my design. Rather than a linear sequence from research to synthesis to making, I feel like there was a symbiotic relationship between the construction of my table and the theoretical development of my concept for a service design tool. Making is not only the means of creating the final product, but is method for design exploration and inspiration as well.

The construction of my prototype multi-touch table consisted of both hardware and software components.

Hardware

The primary hardware of the table includes a projector, a web camera, and a wooden box that forms the structure of the table (see Figures 4 and 5). Conceptually, the table's functionality is relatively straightforward. The piece of acrylic supported on the top of the table acts as the multi-touch surface. The projector is located inside the box, positioned so that the projected display falls on the acrylic screen from behind. The web camera

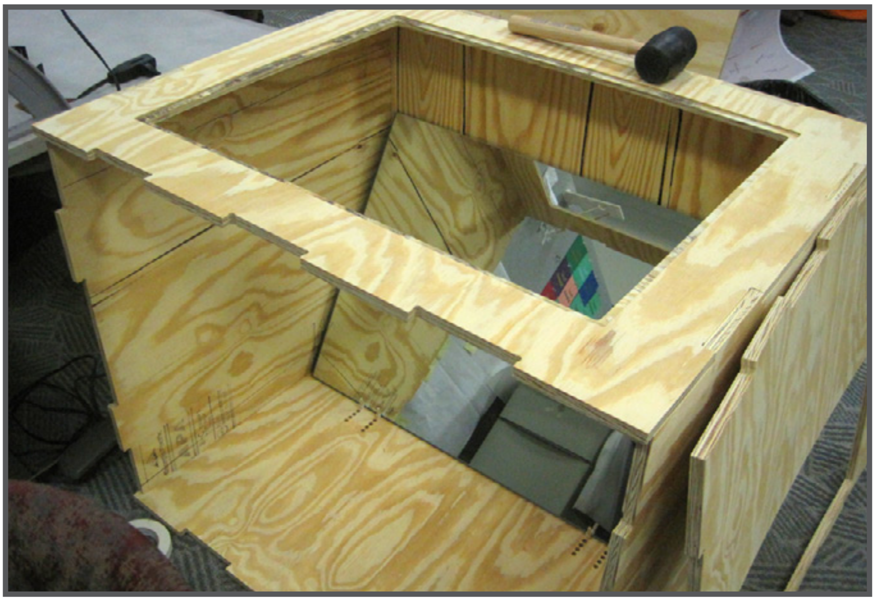

Figure 4. A view of the table as it was being constructed.

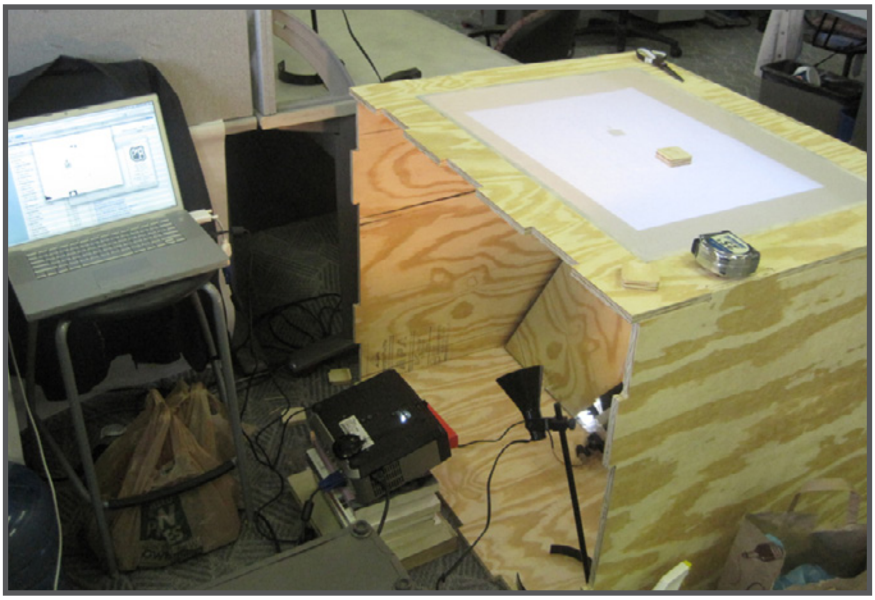

Figure 5. An initial test of the software and projector setup.

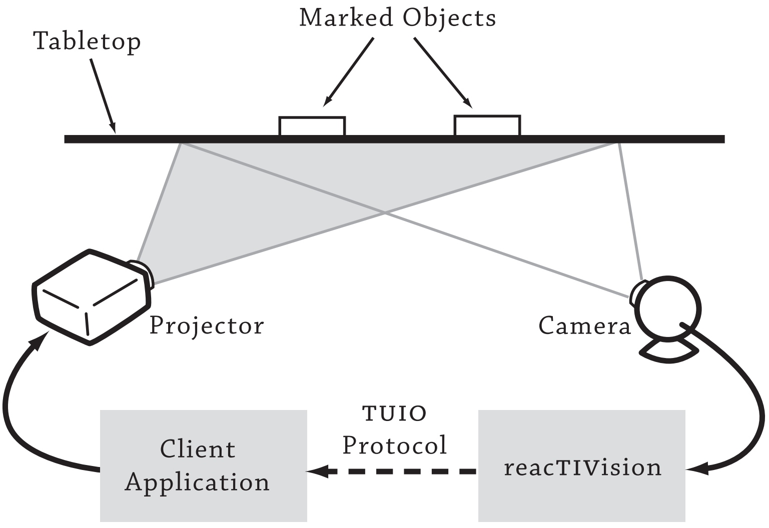

Figure 6. This diagram shows the basic data flow of the system. Adapted from tuio.org.

is also placed inside the table, pointing upwards so that it can see the underside of the surface. This allows it to record the position of any finger touches or marked objects placed on the tabletop. That information is fed to the computer and processed by the software, which updates the projector display to reflect the multi-touch input (see Figure 6).

Of course, there were a number of additional considerations when assembling the table in reality. For instance, the size and position of the projected display is determined by a number of interdependent variables, including the distance from the projector to the screen, the projector's throw ratio, the dimensions of the table, and the angle of projection. Coordinating these variables to achieve the desired display is a small technical challenge in itself. I benefited greatly from the online community of do-it-yourself multi-touch enthusiasts and the tools they have developed and made available for everyone.

Software

There are two main parts to the software system of my multi-touch table. The first is reacTIVsion, the open-source C++ project for multi-touch applications that I encountered during my literature review. Taking advantage of its availability to the public, I decided to use the reacTIVision engine to do the multi-touch sensing for my application. Essentially, reacTIVision can take video input

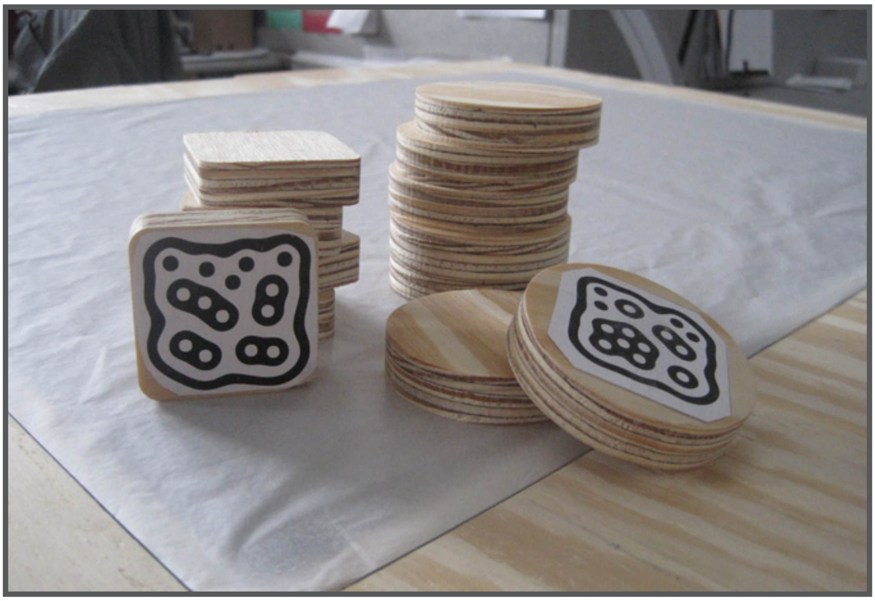

Figure 7. These symbols mark the objects that are tracked by reacTIVision. Each symbol is unique.

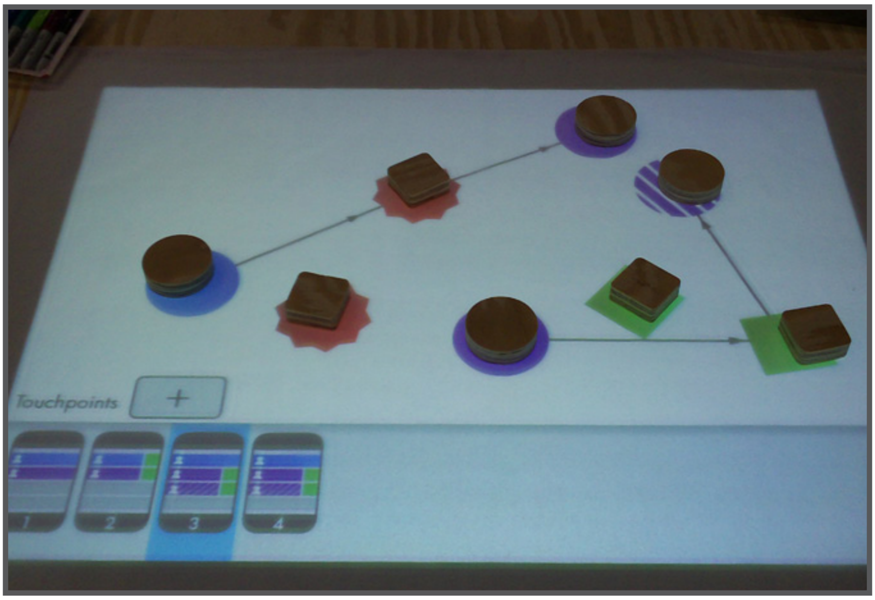

Figure 8. An early iteration of my software prototype.

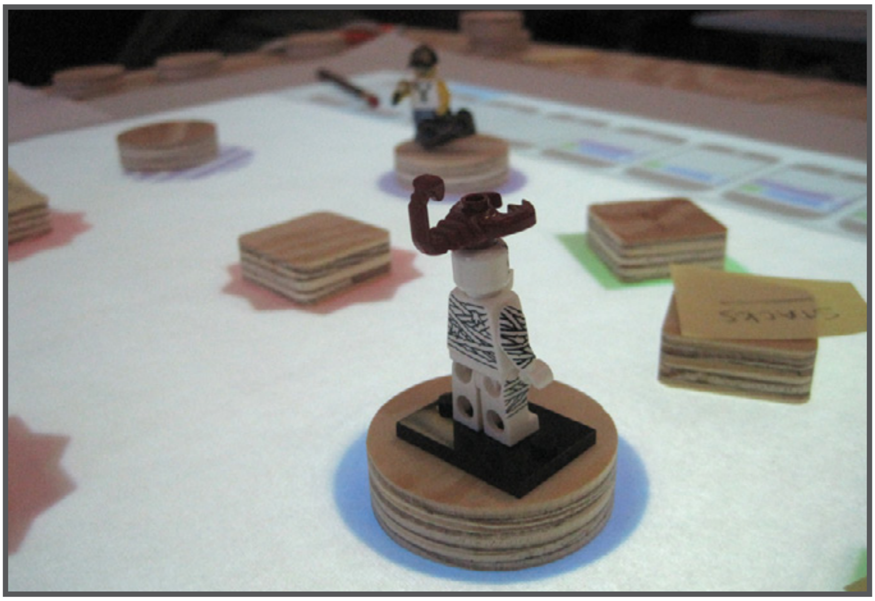

Figure 9. Development was iterative; I would test my early prototypes with designers while it was in progress. This was the results of one of those sessions.

and output the position, orientation, and identification information of any finger touches or objects marked with a particular symbol (see Figure 7).

The second software component is my own Flash-based application that controls the user interface of the table. Written in Actionscript, this code takes the multi-touch data from reacTIVision and uses it to update the display of the application. The bulk of the programming effort for this project was in the creation and testing of this application.

For a more detailed overview of the technical aspects of ServiceSketch, including the specifics of the software setup, see Appendix A.

Development

Taking cues from my initial user research sessions, the first version of my application functioned as a simple diagramming tool to help support activities like categorization and drawing relationships between entities. Incorporating some terminology from service blueprinting, I used basic shapes to represent various elements of a service including actors, actions, and tangible evidence (see Figure 8).

Development of the prototype proceeded in an iterative fashion. I built multiple versions of the prototype, starting with my first basic prototype and adding new features incrementally. Between iterations, I brought in graduate design students or other designers to use the tool in its incomplete state and gather feedback about its usefulness and usability (see Figure 9).

These evaluative sessions served as a form of participatory design. The state of the application was nebulous enough that the discussions I held with other designers helped to shape the tool by revealing the need for certain features to be incorporated. Soon, I began adding more complex functions such as text labeling, color coding, and emotion scales.

But more than simply guiding feature decisions, the dialogue with other designers helped me realize I was not only designing a multi-touch table application, but also the activity that occurred over the table. Much of the value of my application would arise from the conversations and reflections sparked by its collaborative use.

This revelation played into an interesting design question: if I was designing to facilitate a conversation rather than carry out a specific function, how much expressivity should I build into the tool? By expressivity, I mean the degree to which the tool reflects the designer's intentions. Some tools, like a whiteboard or post-it notes, are highly expressive and allow the designer to represent things however they wish. Other tools are less expressive, like a programming language that requires strict adherence to a particular syntax, but can potentially perform more efficient or complex functions.

In the end, I found that a balance had to be struck for the right amount of expressivity. I wanted the tool to be flexible and general enough that designers could use it in multiple ways to support their service design process, but I also wanted to build in enough features and constraints for the tool itself to be valuable. Navigating this balance led me to the final form of the tool, which I've named ServiceSketch.

FINAL DESIGN

ServiceSketch is a collaborative tool to help designers grasp the complexity of designing services. I imagine the tool situated in a design studio being used by a group of designers. I've focused here primarily on its use during the explorations phases of a service design.

The activity is mediated through a multi-touch application and a set of physical pieces (see Figure 10). The application interface has two main components, the stage and the timeline. The stage is where designers place the physical pieces to start to model their service. The timeline introduces a temporal element and allows designers to navigate back and forth through the service process (see Figure 11).

While the exact use of ServiceSketch is left intentionally ambiguous to encourage experimentation, I've developed a basic protocol that helps structure a session with the tool.

Step 1: Building the scene

The activity begins with building the scene on the stage. The basic building blocks of ServiceSketch are actors, objects, and touchpoints.

Actors, represented by circular pieces, can be used to describe any entity that performs actions in the service, including customers, employees, support staff and even machines. When an actor is placed on the stage, the designer can add a text label and select a color for the actor (see Figure 12). Colors can be used to distinguish customers from employees, on-site versus off-site, etc. The exact meaning of the colors is left up to the design team. Users can re-label or change the

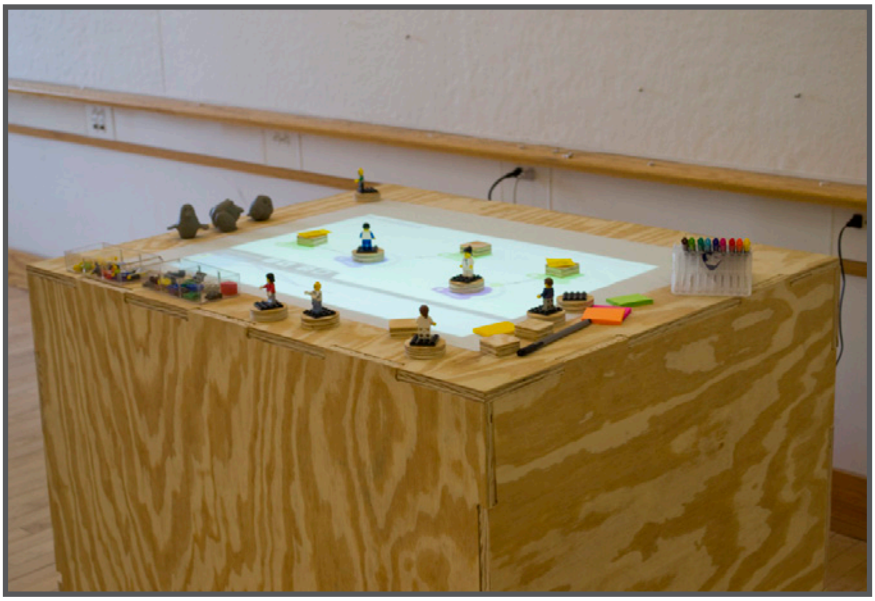

Figure 10. The completed multi-touch table, with assorted materials for construction of pieces.

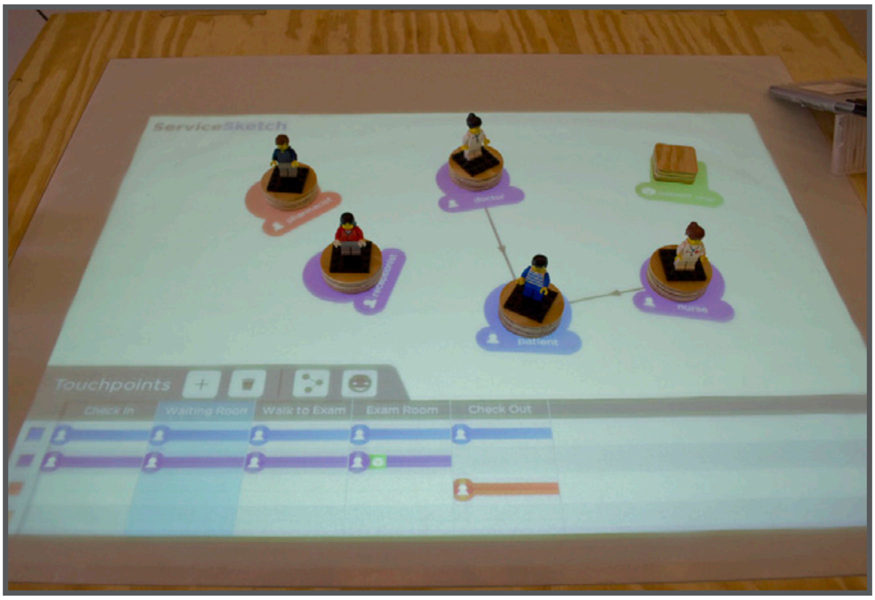

Figure 11. The main interface of ServiceSketch has a timeline (along the bottom) and a stage (where the physical pieces are placed).

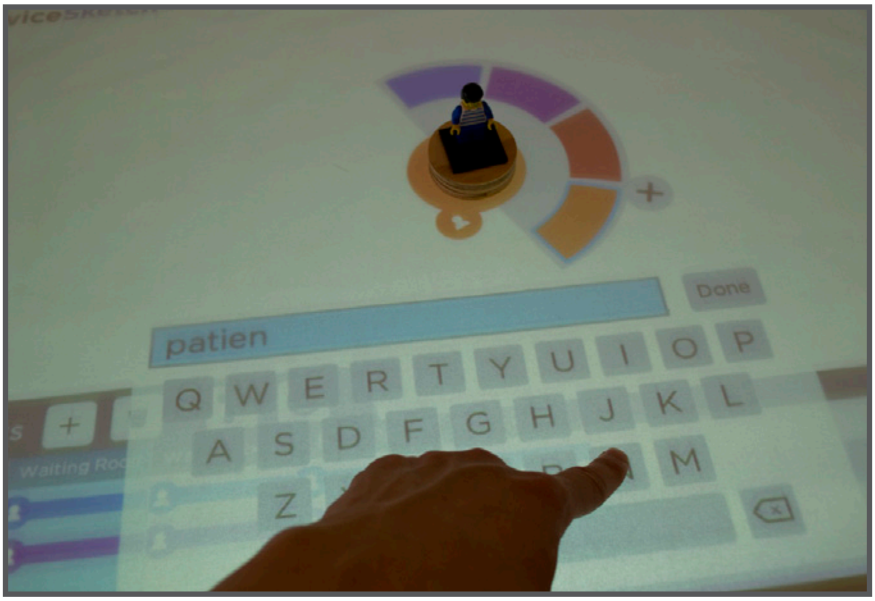

Figure 12. Actors can be labelled and color-coded. Object pieces can be similarly labelled.

actor color at any time by tapping the icon next to the actor piece.

Objects are represented by square pieces. They are green in color and can also be labeled. These are meant to represent the physical objects that take part in the system, the tangible evidence of the service. For example, the tangible evidence of a banking service might include the receipt you receive after a transaction or the attire of the employees of the bank. These objects are key to the experience of the customer since they are often the only clues that speak to the quality of the service.

Touchpoints can be found on the timeline. They represent discrete phases or situations of the service. For example, to continue the example of a bank, touchpoints might include the entrance into the bank, waiting in line, talking to the teller, etc. Two buttons on the timeline allow the user to add or remove touchpoints. Touchpoint labels can be changed by tapping on their title. Tapping anywhere else on the touchpoint will select it as the current touchpoint being viewed.

An important aspect of building the scene is actually constructing physical representations of actors and objects. For my prototype, I've used LEGO pieces to make this construction simple. Each actor piece has a small LEGO platform and designers can build custom minifigures with pieces I've provided (see Figure 13). Of

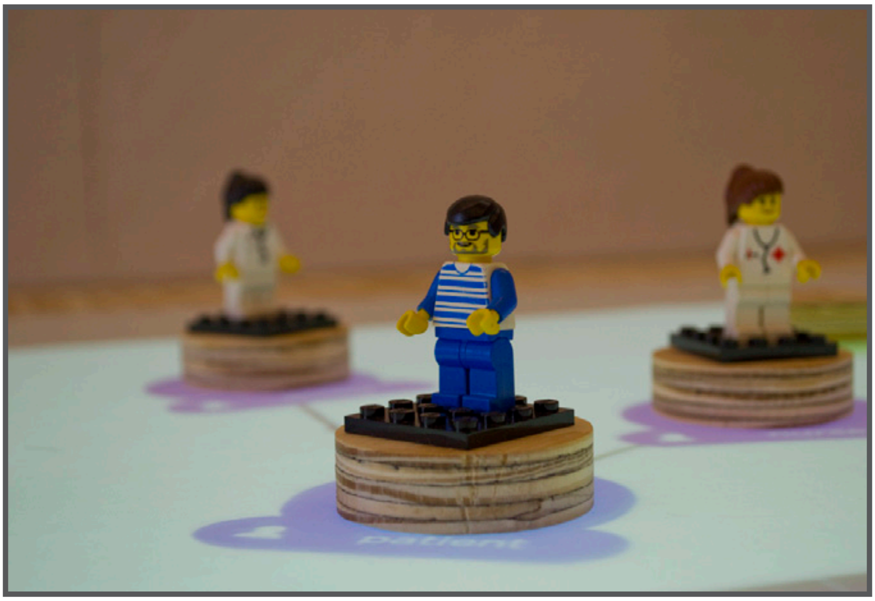

Figure 13. I used LEGO to make construction of actors pieces fun and simple.

course, these models could be built from any materials, like clay, or they could even simply be drawings on post-it notes. This creative aspect of the process is similar to Sander's use of generative toolkits. By building physical representations that everyone can see, we reveal our tacit understandings of the situation to the group.

Step 2: Relationships and Emotions

Once the group has built some of the scene, they move into the next step, creating relationships and tracking emotion. Relationships are important when we want to model a service. In some ways, a service only exists in the intangible relationships between actors. ServiceSketch uses a basic mechanism to begin to represent relationships. Lifting an actor or object piece and placing it near another will draw a digital line between them (see Figure 14). This is a directed link with a small arrow indicating direction. By tapping on the link, a text label can be added. These links can represent actions between actors and objects or other types of relationships. For example, we might draw a line in between a patient and a doctor and label it "talk" to represent the action of a check up at the hospital (see Figure 15).

When a link is drawn between two actors, this is reflected in the current touchpoint on the timeline. The timeline is divided into layers, one

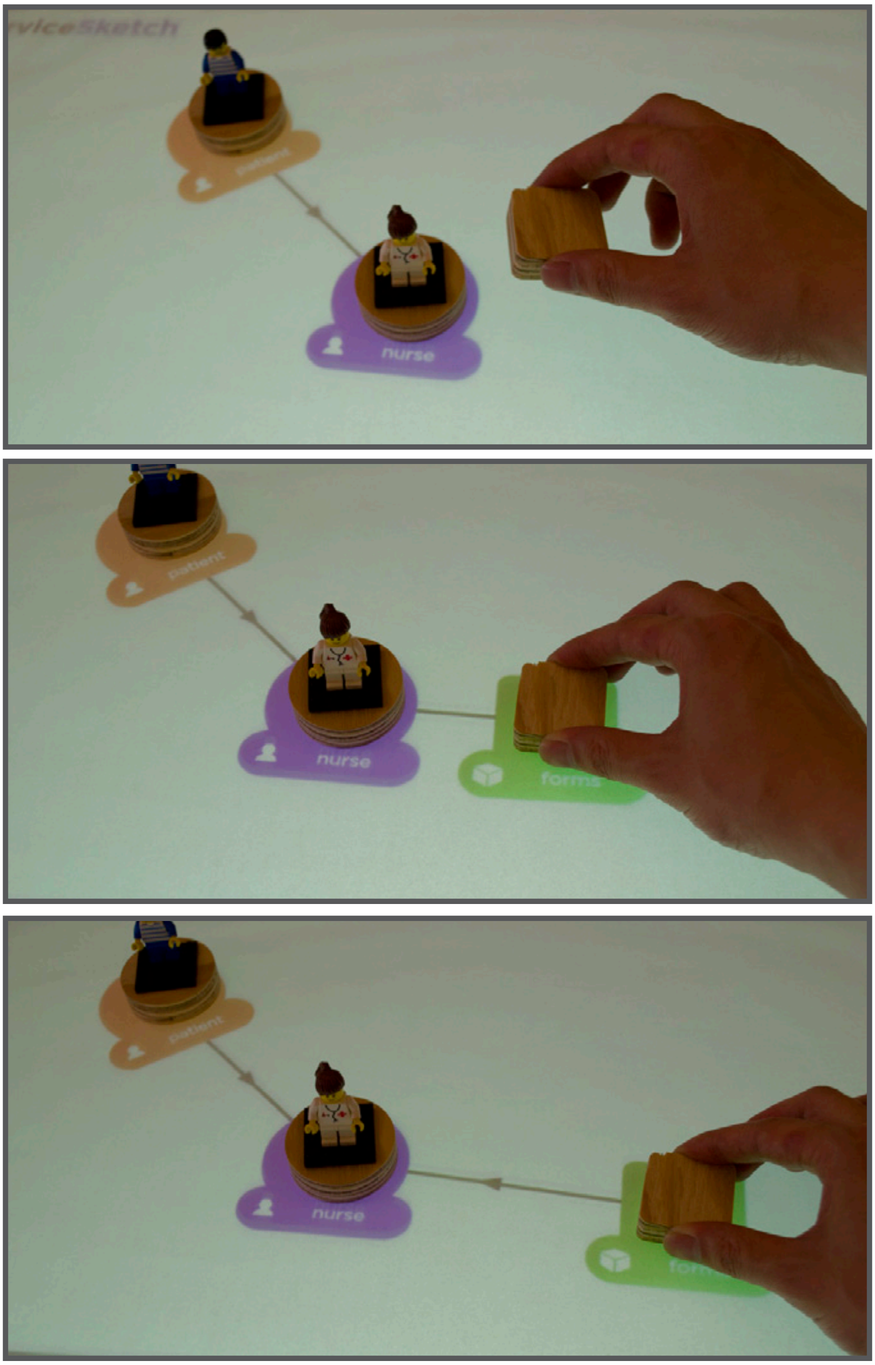

Figure 14. A link can be drawn between actor or object pieces by placing one pieces near another and then drawing it back.

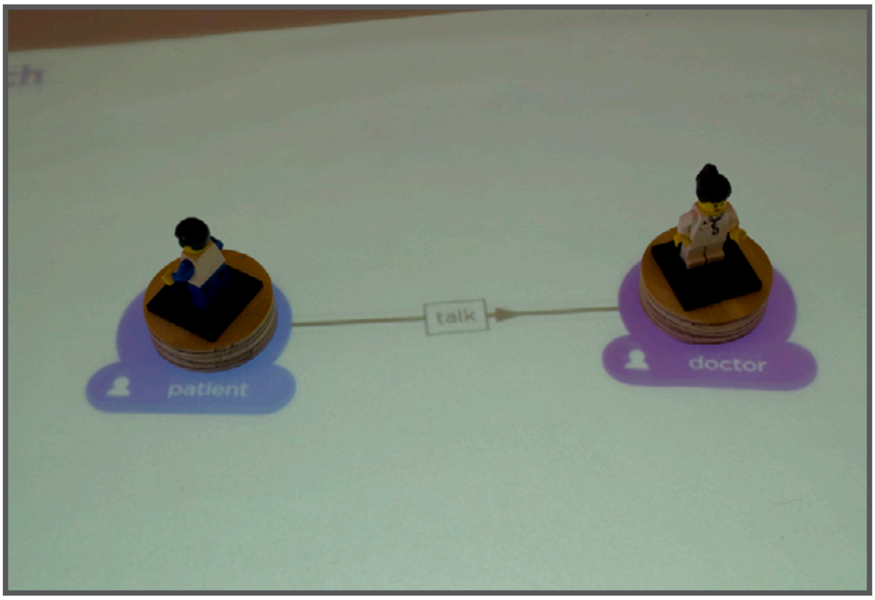

Figure 15. A link can be labelled to specify the type of relationship or action.

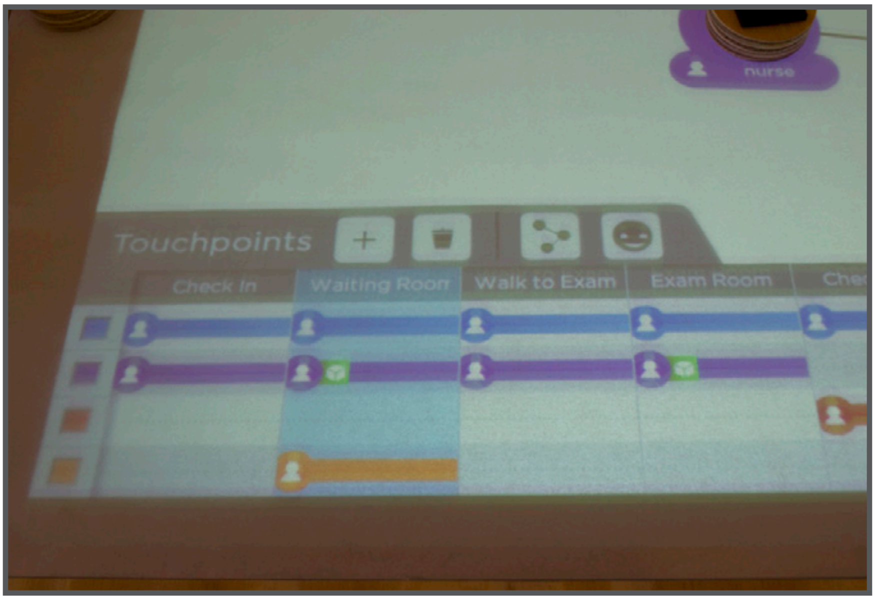

Figure 16. The timeline shows an overview of relationships created in each touchpoint.

for each of the possible actor colors. When an actor of a particular color is linked to another piece, a bar appears in its corresponding layer. If the link is with an object, a small green icon appears alongside the bar to indicate this. In effect, the timeline becomes a rough overview of the relationships of your system (see Figure 16). Once we have modeled a number of touchpoints, we can see at a glance where actors from each particular color category are more active.

By tapping on the happy face button above the timeline, we can change to an "emotion mode." In this mode, each actor is overlaid with a circular, emotional scale. By touching anywhere along the scale, we can set an emotional state for that actor (see Figure 17). The meaning of the scale is open for interpretation. It could help to represent any sort of emotion related to the service (satisfaction, frustration, anger, etc.).

The timeline also transforms in "emotion mode" to display the emotion plot. This graph reflects any changes made to individual emotional scales. Each actor color category has a line on the graph that tracks emotion values from touchpoint to touchpoint. Here again, the timeline serves as an overview by showing clearly where the emotional highs and lows of the service lie (see Figure 18).

Some might remark that the functions built into ServiceSketch to represent relationships and emotions are too simplistic. I would agree that the mechanisms used in this early prototype fall very short of capturing the complexity of these concepts. While we might imagine future iterations with more comprehensive functions, it should be remembered that ServiceSketch is not focused on quantifying and recording these qualities. Instead, these simple mechanisms serve to initiate a conversation among the group about relationships and emotions. It is through this conversation that an understanding of the richness of the service experience emerges.

Step 3: Reflection

This final step of the protocol is a reminder that the focus of ServiceSketch should not be on accurately modeling a service with the tool, but on creating a model sufficient enough to reflect upon together. Designers should not be afraid to tell stories, act out situations, and have fun with ServiceSketch. As I found from my research, games and play can provide an engaging framework for discussion.

Of course, in practice, the steps of this protocol will blend into one another. The natural flow of the design conversation will direct the use of ServiceSketch. Users will probably go through multiple cycles of building and reflection. Usually, it won't be critical to produce a firm plan or artifact for your service design through the process. Like any other sketch, ServiceSketch is about exploration and inspiration, not concrete resolutions.

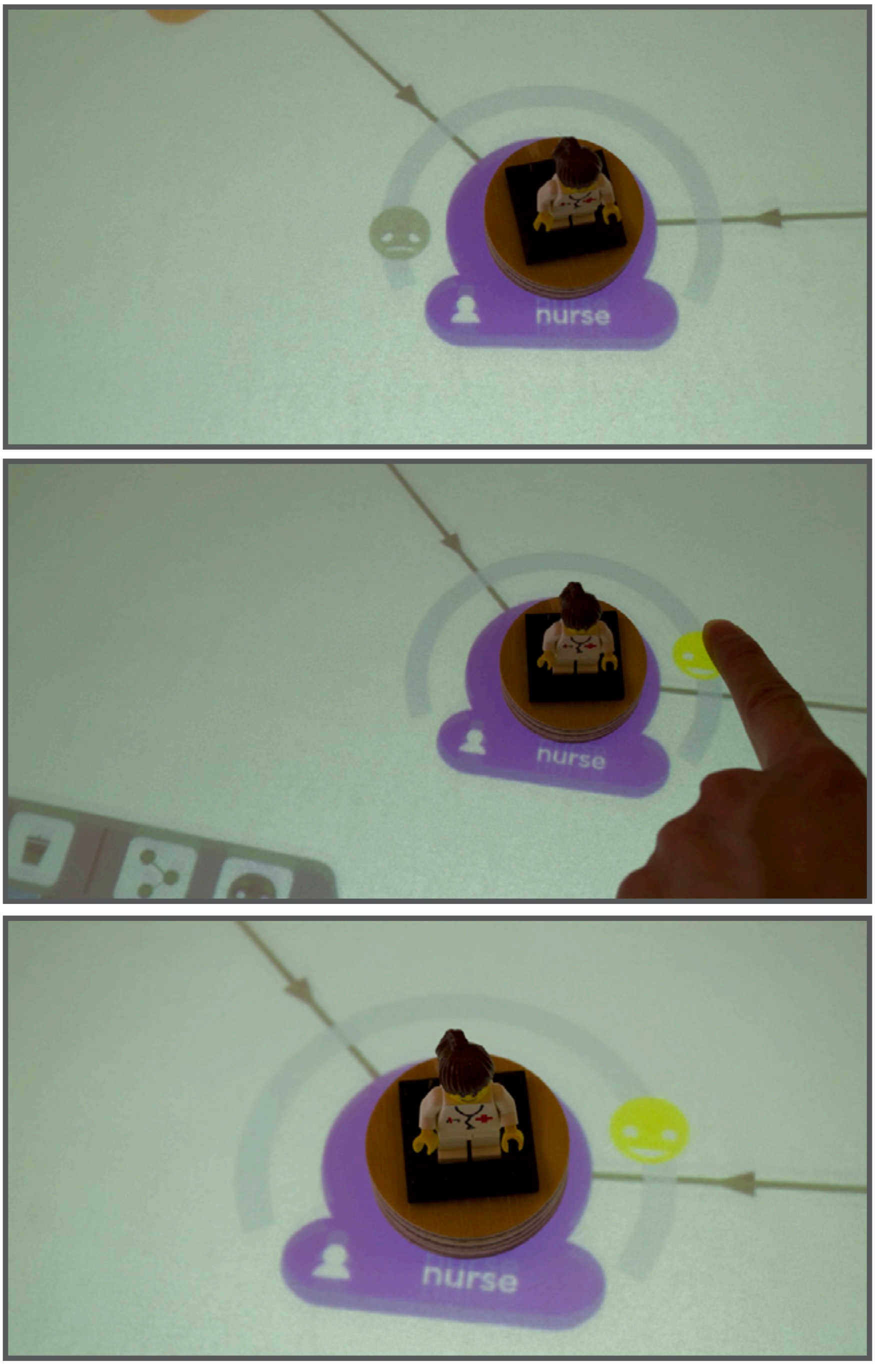

Figure 17. In "emotion mode", each actor has a simple emotional scale that can be adjusted to represent happiness, frustration, or other types of emotion.

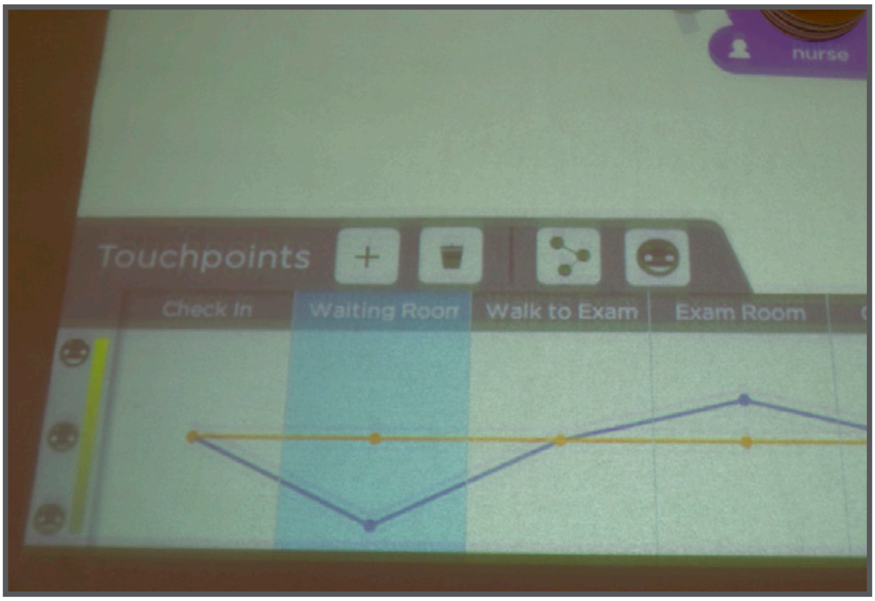

Figure 18. When in "emotion mode", the timeline changes into a plot of the emotional values set for each actor category.

EVALUATION

In order to begin evaluating ServiceSketch, it's useful to recall what issues initially led me to its design. First, I had an interest in service design and, in particular, the tools that service designers use to manage their designs. How could these tools be improved? What would a more dynamic and collaborative service design tool look like?

Second, I wanted to explore tangible user interfaces and attempt to develop my own. What kind of tangible interactions would be appropriate for my tool? What are the qualities of a usable TUI?

And last, I was interested in the effect that hands-on co-creation and play would have on the quality of interactions with my tool. After all, shouldn't design be fun?

These topics form the foundation of my design, so it's useful to evaluate ServiceSketch in relation to each of them.

ServiceSketch as a Service Design Tool

As described earlier, ServiceSketch was evaluated by groups of designers throughout its development. Each session required the groups to perform a small service design exercise with the tool. In addition, I had a number of opportunities to informally demo the tool for various professional designers and gather their feedback on ServiceSketch's usefulness as a service design tool.

Broadly, most participants remarked upon the potential they saw in ServiceSketch to help the practice of service design. In my final evaluation

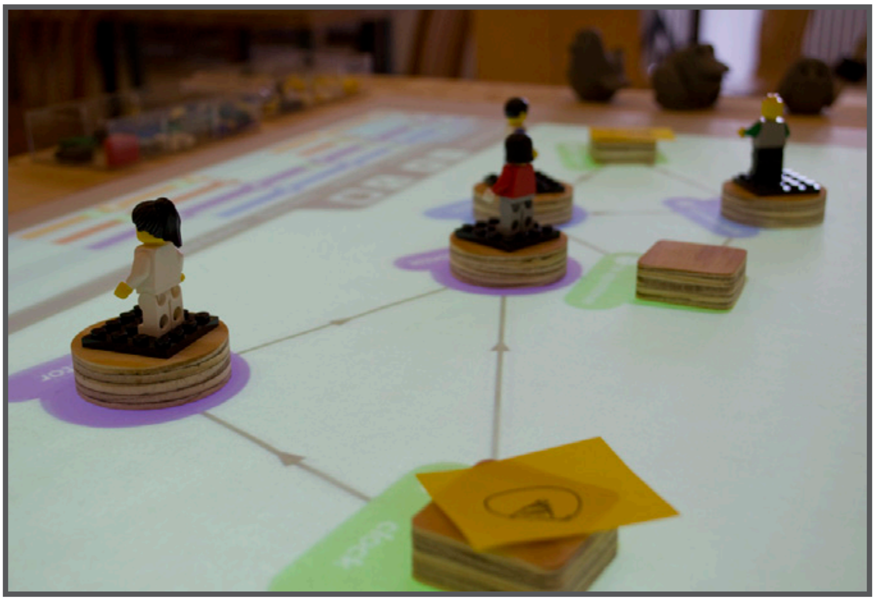
Figure 19. ServiceSketch was successful in supporting common service design tasks. This is one example service model created during a session.

sessions with other design students, I saw how the use of the tool helped lead the discussion. Creation of actors or objects on the table by one designer would stimulate comments or ideas from another designer. Comments received from participants indicated they liked working with common interactive artifact that laid all the elements in front of them. Participants also liked the ability to move back and forth through the timeline, a dynamic feature that more traditional tools lack. This feedback suggests that ServiceSketch was successful in supporting common service design activities (see Figure 19).

In fact, most designers who used the tool started suggesting more features to support other aspects of service design. For example, some asked for the ability to track financial considerations or information about the physical environment of the service. These requests were encouraging; they indicated that ServiceSketch as a basic concept resonated with most people and that it offered multiple possibilities for future development.

The request for more features also recalled my struggle with the level of expressivity built into the tool. If more features were added, the tool might be more constraining in the sense that the features would dictate its use and perhaps limit the scope of the design problem to what can be represented on the tool. If ServiceSketch were developed further, careful consideration should be given to how the tool is framing the design. Perhaps the lesson is that any tool will frame the design problem in a certain way, and so multiple tools should be used when designing.

ServiceSketch as a TUI

Developing the interactions for ServiceSketch was a challenge for me because I had never designed for multi-touch applications, let alone a large-scale multi-touch display. And because these types of TUIs are not yet widespread, there are few standard conventions to draw from. This will change with time and advancing technology, but it was engaging work attempting to develop my own conventions.

The first decisions about the interface were made before I programmed anything. Basic things, like the height of the table or the tabletop surface area, had great impact on how people interacted with it. Here we see how the making process was integral with the design process. The size of the display, in particular, came up in evaluative sessions often. My prototype had a display of about 25 inches by 20 inches. It was sufficient for the small exercises I was testing with, but participants wondered how the tool would scale for larger designs.

Since a tabletop display can only be so large before becoming cumbersome, ServiceSketch in its current form is probably more usable with a smaller group trying to model short vignettes of a larger service system. If the tool were adapted for a larger scale, strategies would have to be developed to resolve the space limitations, such as information hiding or use of hierarchy.

Since ServiceSketch uses simple wooden pieces as manipulators, it takes advantage of people's familiarity with natural, physical interactions. Everyone who walked up to ServiceSketch recognized they could grasp and move the actor and object pieces to manipulate the display (see Figure 20). However, it was challenging trying to find an intuitive interaction for other functions. For example, it took some instruction for people to understand how to draw a link between pieces. But despite the learning curve involved, most people were able to adapt to the interface quite quickly. It would seem that, especially for novel interfaces like TUIs, people are willing to invest time in learning how to use the interface if they are interested in what the tool can do for them.

ServiceSketch as Play

One of the more unexpected results from my evaluative sessions was the playfulness that ServiceSketch aroused in designers. Perhaps because of my choice to use LEGO figures, the participants of my sessions all had fun constructing actor pieces. I feel that this playfulness carried throughout the activities and led to some creative developments. In one notable instance, the participants actually used the actor pieces to represent simple personas, going so far as to name them and give them personalities. While this was done only partly seriously, I think it helped

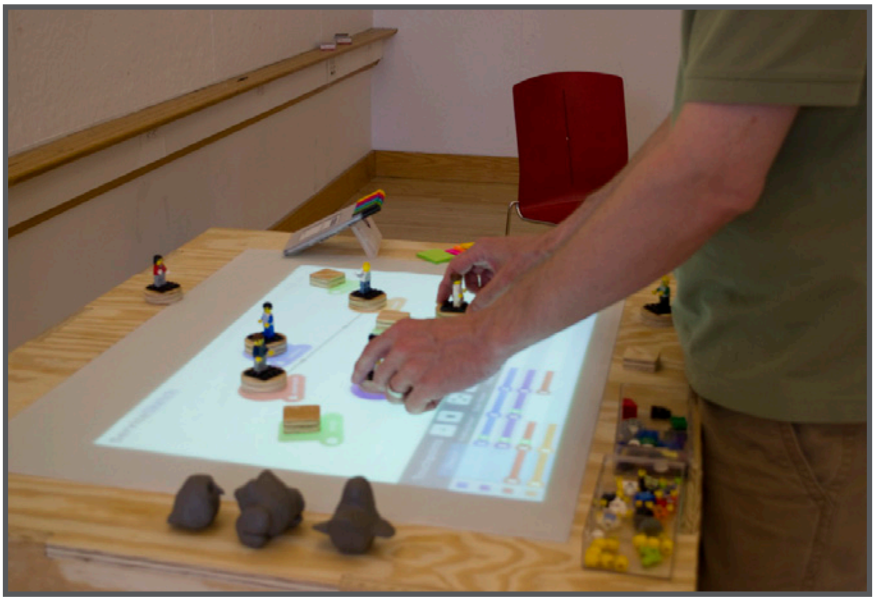
Figure 20. Everyone who used ServiceSketch seemed to become engaged with the activity, despite having to learn how to use the tangible user interface.

keep people engaged in what could have potentially been a somewhat dry activity.

These findings highlight another possible direction for the development of ServiceSketch. As one participant mentioned, it is interesting to explore service through a story. Elements like the customizable LEGO figures help support this storytelling aspect of service design. More of these elements could be built into ServiceSketch, even incorporating traditional design storytelling tools like personas and scenarios. Here we see that the potential of ServiceSketch is not only found in improving technology, but in developing the activity around the technology.

Next Steps

I see multiple branches of possible future development for ServiceSketch. At the feature level, any of the many additions that were requested by the evaluating designers could be developed. Many of these features would serve to address the different aspects of service design. Some examples include the ability to track costs and other financial data, a way to draw or designate the physical spaces and environment that the service takes place in, or a mechanism for navigating multiple or alternate timelines of the service.

At a more systemic level, ServiceSketch could be considered the start of an entire framework for service design software. The current iteration doesn't include a way to store out the data built by the designers, but this could be developed, for example, by implementing a save feature that creates some kind of open data file format. Once we have this format, numerous possibilities are raised. The file could be opened in other popular diagramming programs like Omnigraffle to be further refined into a final model. Or perhaps the file could be opened in other new service design programs that are designed for examining different aspects of service. This could be the beginning of a standard grammar for service design data that could be easily passed back and forth between designers and other stakeholders.

But, as has been made clear throughout this document, the technical capabilities of ServiceSketch are not its only important features. Future work on the tool should also consider the possibilities for the activity occurring over the table. ServiceSketch, as conceived of in this paper, focuses primarily on its use as a design tool by designers, but it could easily be adapted for other activities like storytelling or teaching. In the end, ServiceSketch could help facilitate any type of conversation, not just those about service design.

CONCLUSION

Even in this early prototype version, we have seen that ServiceSketch successfully addresses many aspects of service design. The tool provides a physical, communal artifact that helps focus the development of a service. It helps to facilitate conversations between its users through an engaging, accessible communication medium. And it hints at a wide range of possible future developments for service design tools.

Tools facilitate activity. They need to have the right balance of constraint and freedom, providing enough structure for clear channels of action, without being overly prescriptive and losing the richness achieved in ambiguity. In fact, this is true for any designed artifact, not just tools. It is the best designs that people are able to adapt into their lives and make their own.

REFERENCES

Bitner, M. J., Ostrom, A. L., Morgan, F. N. 2008. “Service Blueprinting: A Practical Technique for Service Innovation.” *California Management Review* 50(3):66-94.

Brandt, E., Messeter, J. 2004. “Facilitating Collaboration Through Design Games.” In *Proceedings of the Participatory Design Conference*, Toronto, Ontario, Canada.

Habraken, N. J., Gross, M. D. 1988. “Concept Design Games.” *Design Studies* 9(3):150-158.

Ishii, H., Ullmer, B. 1997. “Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms”. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 97)*, Altanta, GA, USA.

Johansson, M. 2006. “Design Games: Reinstalling the Designer in Collaborative Design.” In *Proceedings of Design Research Society*, Lisbon, Portugal.

Jorda, S., Geiger, G., Alonso, A., Kaltenbrunner, M. 2007. “The reacTable: Exploring the Synergy between Live Music Performance and Tabletop Tangible Interfaces”. In *Proceedings of the First International Conference on Tangible and Embedded Interaction (TEI 07)*, Baton Rouge, Louisiana, USA.

Kaltenbrunner, M., Bencina, R. 2007. “reacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction”. In *Proceedings of the First International Conference on Tangible and Embedded Interaction (TEI 07)*, Baton Rouge, Louisiana, USA.

Lee, M. K., Folizzi, J. 2009. “Designing Adaptive Robotic Services.” In *Proceedings of IASDR 09*, New York, NY, USA.

The LEGO Group. 2010. “LEGO Serious Play Open Source: An Introduction to LEGO Serious Play.” Accessed Mar 22, 2011. http://www.seriousplay.com/

Polaine, A. 2009. “Blueprint+: Developing a Tool for Service Design.” *Service Design Network Conference*, Madeira, Portugal. Accessed March 18, 2011. http://www.slideshare.net/apolaine/blueprint-developing-a-tool-for-service-design

Sanders, E. B.-N. 2000. “Generative Tools for Codesigning.” *Collaborative Design*, London: Springer-Verlag.

Shostack, G. L. 1977. “Breaking Free From Product Marketing.” *Journal of Marketing* 41:73-80.

Shostack, G. L. 1984. “Designing Services That Deliver.” *Harvard Business Review (January-February)*:133-139.

Spraragen, S., Chan, C. 2008. “Service Blueprinting: When Customer Satisfaction Numbers Aren’t Enough.” *International DMI Education Conference*, Cergy-Pointoise, France.

APPENDIX A

Technical Specifications

A lot of technical information for this project was drawn from online resources created by a large DIY multi-touch community.

Primarily, I received guidance from the work of the open source project, reacTIVision (http://reactivision.sourceforge.net/). Coming out of the Universitat Pompeu Fabra in Barcelona, Spain, reacTIVision was developed as the computer vision technology supporting the development of the reacTable, a multi-touch tabletop music synthesizer. The software works by using the TUIO protocol (http://www.tuio.org/), also created by the reacTIVision developers.

According to the TUIO creators:

"The TUIO protocol allows the transmission of an abstract description of interactive surfaces, including touch events and tangible object states. This protocol encodes control data from a tracker application (e.g. based on computer vision) and sends it to any client application that is capable of decoding the protocol."

There were two main components to building my project: the hardware construction of the table itself and the development and integration of my client software application.

Hardware

The main physical components of the system are a camera, a projector, a screen for the surface display, and the frame of the table. After deciding upon an appropriate size for my screen, I calculated how much distance the projector would need to project the required display dimen-

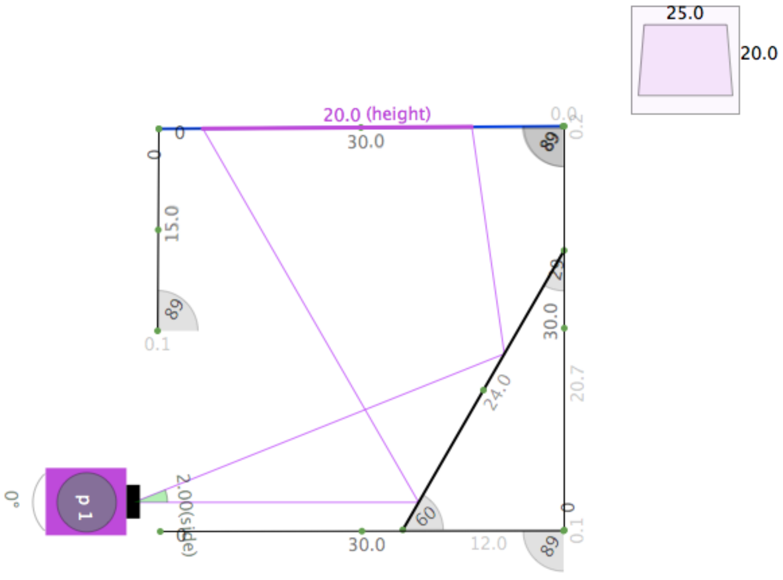

Figure 21. A rough side-view schematic of the table generated with SimProj.

sions. To accommodate long projector throw lengths, a mirror is placed at an angle inside the table. This allows the projection to be projected horizontally and reflected up vertically towards the screen.

To calculate the correct dimensions, I used SimProj (http://benjamin.kuperberg.fr/lab/?p=4), an excellent simulator created specifically for the design of multi-touch table setups (see Figure 21).

Once I had a rough idea of the dimensions for the frame (and after a few quick tests with an actual projector), I made some drawings for the frame of the table. I chose to make the box out of plywood so that I could cut exact pieces on a CNC router. The frame was designed to be pressure fit, with no need for screws or nails. I also cut small shapes that would act as the objects for interacting with the table.

The screen was simply a piece of clear acrylic, laser cut to the appropriate size. A few sheets of vellum are placed over the acrylic in the final setup, creating a translucent surface. This allows the camera to see finger touches and objects placed on the surface of the table, while at the same time providing a surface for the projector to display on.

The camera selected was a PS3 eye. From my research done on the web, I found that this was a low-cost camera that did relatively well in computer vision applications and was popular in the DIY multi-touch community. I modified the camera to only detect infrared light and set up infrared light emitters inside the table. This way, the projected image doesn't interfere with the finger and object detection by the camera.

Software

The two main parts of the software system are reacTIVision and my client application, ServiceSketch.

reacTIVision comes as a fully functional from the web site. It allows for various settings to customize for video dimensions, lighting conditions, etc. It was relatively easy to setup and worked well with the PS3 web camera. Source code is also available if any modifications need to be made to the software.

The client application took the majority of my programming effort. As explained on the TUIO web site, the client can be programmed in a variety of languages, as long as it can understand the TUIO protocol. There exist libraries for working in C++, Java, Processing, and others.

I decided to program my application in Flash so that I could take advantage of its graphics tools and animation capabilities. Unfortunately, Action-script 3 doesn't natively support UDP, which is what TUIO is ultimately built upon.

To get around this, I used udpflashlc-bridge (http://gkaindl.com/software/udp-flashlc-bridge) which bridges UDP input to Flash through AS3's LocalConnection functionality. Combining this with the TUIO AS3 library (http://bubblebird.at/tuioflash/tuio-as3-library/), Flash can understand the TUIO protocol and work with reacTIVision. This process is the method of communication suggested on the TUIO web site.

Note that other languages have more direct means for reading UDP and might provide different options for the client application.

With the proper software connections in place, I was free to program my actual client application. In order to ease development, the TUIO web site provides a TUIO simulator that generates the same inputs that an actual table would provide. This allowed me to develop my client application before I even had a working table.