Mixed-Fidelity Prototyping of User Interfaces

Jennifer N. Petrie and Kevin A. Schneider

Department of Computer Science, University of Saskatchewan,
Saskatoon, SK S7N 5C9, Canada

Abstract. We present a new technique for user interface prototyping, called mixed-fidelity prototyping. Mixed-fidelity prototyping combines and supports independent refinement of low-, medium-, and high-fidelity interface elements within a single prototype. Designers are able to investigate alternate, more innovative designs, and are able to elicit feedback from stakeholders without having to commit too early in the process. The approach encourages collaboration among a diverse group of stakeholders throughout the design process. For example, individuals who specialize in specific fidelities, such as high-fidelity components, are able to become involved earlier on in the process.

We developed a conceptual model called the Region Model and implemented a proof-of-concept system called ProtoMixer. We then demonstrated the mixed-fidelity approach by using ProtoMixer to design an example application.

1 Introduction

User interface prototyping is a process for creating mock-ups representing the user interface of the final software system. Prototypes serve as a common language between stakeholders, offering a way for designers to explore design ideas and elicit feedback from stakeholders prior to committing to designs. Prototypes aid in refining requirements and may be used as a specification for developers. Prototyping is important in arriving at a well-designed user interface.

Different fidelities of prototypes can be explored during the prototyping process: low-, medium-, and high-fidelity. Fidelity refers to how closely the prototypes resemble the final product in terms of visual appearance, interaction style, and level of detail [17]. Each fidelity of prototype uses different techniques and mediums and each is important at specific stages in the design process [14]. The commonly accepted best practice of prototyping encourages starting with low-fidelity prototypes then moving to medium- and finally to high-fidelity, refining whole prototypes at each fidelity prior to advancing to a higher-fidelity.

We have identified some shortcomings with this current best practice. Designers typically only work on one fidelity at a time and, while prototyping is often termed 'iterative', designers often only iterate within a specific fidelity. These shortcomings force designers to make decisions on some design issues earlier than desired as well as undesirably delay investigating other more pressing issues. Also, because different fidelities

are performed on different mediums and tools, there is a lack of traceability in the process and transitioning back or forth between fidelities requires significant effort. Furthermore, this practice does not encourage novel designs to be explored. Also, current practice lacks collaboration between various stakeholder groups, such as end users and software developers. These shortcomings are evident in and reinforced by the existing support tools.

To address these shortcomings we have developed the mixed-fidelity prototyping approach. Mixed-fidelity prototyping involves combining multiple fidelities within a single prototype. This allows designers to independently explore and refine individual elements within a prototype, while maintaining the element within the context of the overall design. By mixing fidelities, we aim to enhance the collaboration throughout the prototyping process by bringing together various stakeholder groups earlier on and allowing for more active participation. We utilize a large interactive display workspace in this research to further encourage collaboration.

In the remainder of this paper, we discuss related work on prototyping techniques and tools as well as collaborative large display projects. Next, we describe the mixed-fidelity prototyping approach further and include a conceptual model we developed, called the Region Model, for supporting the approach. We also provide an overview of a proof-of-concept system called ProtoMixer. Finally, an example design session is presented where ProtoMixer is used to design an example application to illustrate the approach.

2 Background

Low-fidelity prototypes are best used early in the design process when trying to understand user requirements and expectations [14]. Low-fidelity prototypes are created on physical mediums such as paper, whiteboards, or chalkboards. Freehand sketching is one of the most common techniques for low-fidelity prototyping as it allows for ideas to be left intentionally vague and informal [11] and it encourages thinking [16].

Medium-fidelity prototypes are refined versions of the low-fidelity prototypes and are created on the computer. Medium-fidelity prototypes are commonly created using multimedia design tools, interface builders, or scripting languages such as tcl/tk [12].

High-fidelity prototypes are refined versions of the medium-fidelity that typically have some level of functionality implemented and may link to some sample data. High-fidelity prototypes are computer-based prototypes that are often developed using interface builders or certain scripting languages to speed up the process. High-fidelity prototypes are particularly useful for performing user evaluations as well as for serving as a specification for developers and as a tool for marketing and stakeholder buy-in [14].

The majority of prototypes are developed using some type of support tool. One of the most widely used class of tools is Interface Builders, such as Microsoft® Visual Basic®, Borland® Delphi™, and Metrowerks™ CodeWarrior™. Interface Builders aid designers in creating and laying out interfaces by allowing for interface components to be dragged into position on the desired screen. Interface Builders may be used for high-fidelity

and, to a lesser extend, medium-fidelity prototyping. Interface builders are restrictive in terms of what designs designers can build as well as the order in which designers have to build it and they require significant time and effort to create a prototype.

Another widely used class of tools is Multimedia Design tools, which includes commercial tools such as Macromedia® Director® and Flash® as well as Apple® HyperCard® and tools from the research community such as DEMAIS [1] and Anecdote [5]. Multimedia design tools are useful in creating and demonstrating storyboards in medium-fidelity prototyping as they allow for creation of images that can represent user interface screens and components as well as for playing out navigational transitions from one screen to the next. On the negative side, the interactivity supported by multimedia design tools is very limited, usually to only basic mouse clicks, and so is support for creating functionality and tying in data.

A number of tools support freehand sketching. SILK [9], one of the first tools to support informal sketching of user interfaces, also provided support for transitioning to a higher-fidelity through automatic interface component transformation to working components. DENIM [10] is a tool aimed at supporting the early stages of web design through informal sketching and provides for creating and running designs of different levels of granularity (from sitemap to storyboards to individual pages) [10].

In recent years, researchers have shown considerable interest in attempting to bridge the gap between interface and software design through a series of workshops [6–8]. Gurantene et al. [4] argue for using high-fidelity prototypes as a bridging artifact. However, prototyping tools lack support for the diverse roles of user interface designers, graphic artists, software engineers, and end users.

Design workspaces and, more specifically, work surfaces influence collaboration [2, 15]. For instance, work surfaces help focus designers attention and aid in expressing creativity. Furthermore, workspaces provide a medium for designers to communicate through with actions such as drawing, writing notes, and gesturing to emphasize or reference previously made points. For these reasons, design teams must have suitable workspaces. We use large displays in our research as large display workspaces are conducive to collaboration, allowing for multiple people to simultaneously work directly at the surface while allowing for everyone in the room to be aware of the workspace content.

3 Mixed-Fidelity Prototyping

Mixed-fidelity prototyping involves combining multiple fidelities within a single prototype. As an example, consider having a sketched screen design that contains various sketched elements. The sketch may also contain images in place of sketched elements and also could have one or more interface elements presented as high-fidelity working components. The sketched elements or images may also be given some form of behavior similar to what they would possess at traditional higher-fidelities.

Mixed-fidelity prototyping allows designers the opportunity to focus on a specific interface issue, by exploring it at higher-fidelities and making refinements as needed. In the mean time, other aspects of the prototype may be left at a lower-fidelity, delaying decisions while allowing designers to redirect their time and efforts to the more pressing design issue(s). Also, by leaving other elements at lower-fidelities, designers are able to explore the higher-fidelity elements while keeping them within the context of more complete screen designs.

Mixed-fidelity prototyping varies from the traditional process, which limits iteration to occur within the current fidelity. Also, traditional practice does not encourage advancing to the next higher-fidelity until ideas have been refined at the current fidelity and does not encourage skipping fidelities. Finally, traditional practice discourages iterating to a higher-fidelity at the element level, rather only once the whole prototype is ready for advancement. Figure 1 depicts the limited iterative nature of the traditional prototyping process.

Fig. 1. Traditional prototyping process with limited iteration opportunities

Mixed-fidelity prototyping is a fully iterative process. Designers may advance to any higher-fidelity at any point in time as well as revert back to any earlier fidelity as desired. Also, mixed-fidelity prototyping allows iterative refinement at the element level rather than only for the overall design. Figure 2 show the iterations possible with our mixed-fidelity approach.

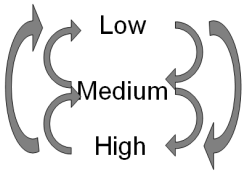

Fig. 2. Mixed-fidelity prototyping as a fully iterative process

Prior to developing our model and prototype we designed several collaborative prototyping scenarios. We used these scenarios to discover unique issues that could be addressed with mixed-fidelity prototyping. We identified the following novel or key concepts: (1) mixing elements of multiple fidelities in a single prototype, (2) transitioning between the fidelities as ideas are refined, (3) integrating domain-specific data and functionality, (4) exploring novel interactive elements, (5) comparing alternative

designs, and (6) recording the design process. These issues are not easily possible or supported under current practice and with existing tools.

4 The Region Model for Mixed-Fidelity Prototyping

We developed a conceptual model, called the Region Model, to support mixed-fidelity prototyping. Prototypes are composed of multiple elements on the design space. Prototypes are composed using the region metaphor by overlaying regions on other regions to arrive at a desired design. Overlaid regions are related in a parent-subregion hierarchy. The root region is also referred to as the *design space*. Figure 3 illustrates how regions can be used to compose prototypes.

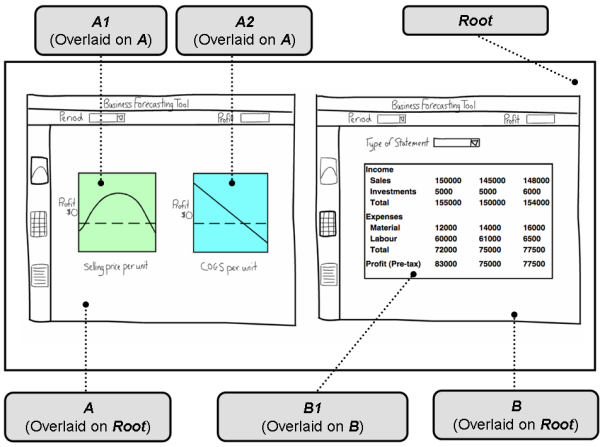

Fig. 3. Using the concept of regions to compose prototypes

Regions have both visual properties as well as associated behaviors. The visual presentation or form that a region takes is represented as *assets*, where assets may be sketches, images, or high-fidelity widgets, for example. A region have spatial and visual attributes, such as x, y, and z coordinates, width and height, as well as color and transparency. A region may be associated with a layout algorithm to automatically position its subregions. Finally, regions have a history list, which are clones of themselves over a period of time.

Behavior is represented as *scripts* that are bound to regions, where the scripts perform manipulations on regions. Scripts are made up of *commands*, which are used to perform basic manipulations on regions such as modifying their size or position.

Regions may be connected through *relationships*. Relationships specify that a change in one region's properties affect other regions. Relationships may exist for two main purposes: to bind behaviors and to indicate navigational flow.

The Region Model can be used to describe the current and historical state of the prototypes being designed as well as the design space itself and how alternative prototypes are arranged within the design space. Figure 4 is a UML Class Diagram illustrating the major concepts in the Region Model.

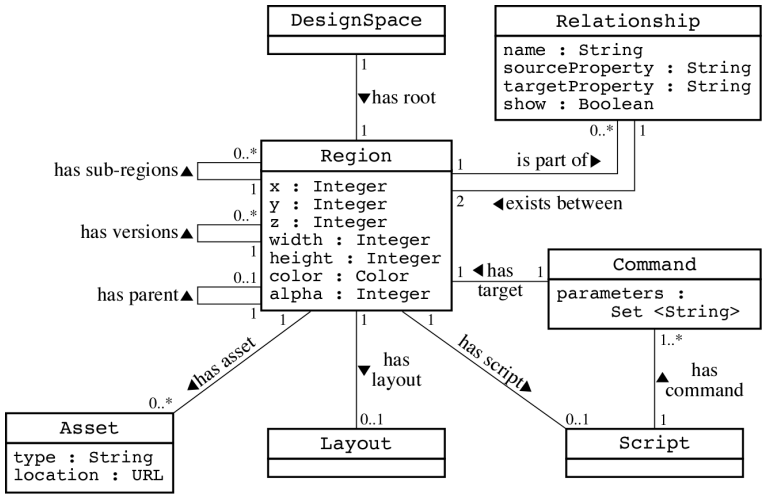

Fig. 4. Region Model illustrated as a UML Diagram

An XML[3] based notation was developed for specifying the Region Model. Below is an example XML document describing the design space in Figure 3.

```
<DesignSpace>
    <Region name="Root" x="..." y="..." ... parentRegion="null">
        <Region name="A" x="..." y="..." ...  parentRegion="Root">
            <Asset type="Image" location="..."/>
            <Region name="A1" x="..." y="..." ... parentRegion="A">
                <Asset type="Image" location="..."/> </Region>
            <Region name="A2" x="..." y="..." ... parentRegion="A">
               <Asset type="Image" location="..."/>  </Region>
        </Region>
        <Region name="B" x="..." y="..." ... parentRegion="Root">
            <Asset type="Image" location="..."/>
            <Region name="B1" x="..."  y="..." ... parentRegion="B">
                <Asset type="Image" location="..."/>  </Region>
        </Region>
    </Region>
</DesignSpace>
```

5 ProtoMixer: Software Support for Mixed Fidelity Prototyping

ProtoMixer, shown in Figure 5, is a proof-of-concept system developed to support the mixed-fidelity prototyping approach. ProtoMixer is implemented in Java and the graphics rendering is with Java2D. ProtoMixer is intended to be an easy to use, lightweight system, much in the same manner as a basic drawing editor. All objects may be positioned anywhere on the workspace, which is the same as the canvas in a drawing editor; placement of objects is not restricted by frames and borders as with other tools like Interface Builders. Also, there are no menu bars; all system operations are either performed directly on the object or through a simple command panel (cf. Figure 6).

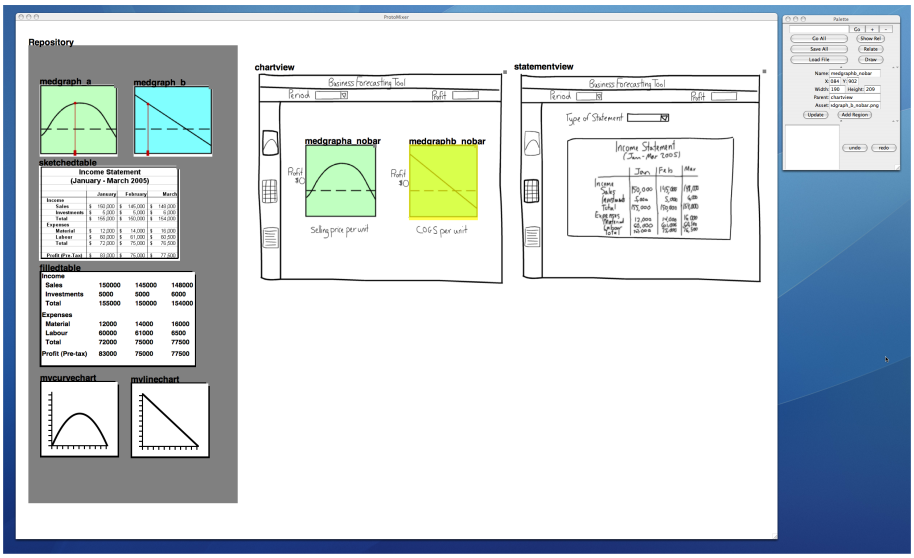

Fig. 5. Screenshot of ProtoMixer

ProtoMixer supports the integration of elements of any fidelity through the use of assets. In the current version of ProtoMixer, assets may be of these types: sketch, image, and widget, where each of these types clearly corresponds to a fidelity. These assets can then be created outside of ProtoMixer using tools and mediums that designers are accustomed to. For example, sketches can be created on paper and then be scanned as images and imported into ProtoMixer. As well, images can be created in the designers' favorite multimedia application. High-fidelity components can be coded in an Interface Builder or from scratch and be imported into our system. These components may be any of Java's pre-built widgets, such as high-level control components (like JTables and JTextFields) as well as lower-level general purpose container objects such as JPanel and JscrollPane. ProtoMixer also allows for custom-built components, as long as they extend from JComponent class. ProtoMixer also provides limited support for sketching directly within the tool.

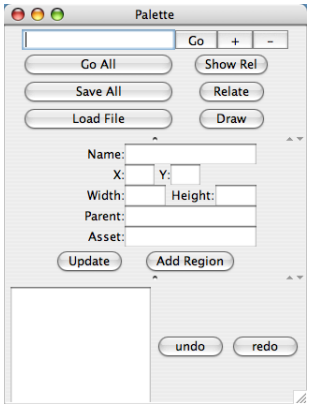

Fig. 6. Screenshot of ProtoMixer's command panel

Prototypes are composed in ProtoMixer by overlaying regions on top of other regions. Visual properties can then be adjusted for regions by specifying new values in the command panel. Then prototypes and/or their elements can be given behavior through a few different approaches: constraints, animation, and bindings. ProtoMixer supports generic binding between high-fidelity elements using the observer design pattern.

ProtoMixer offers a variety of features, too many to discuss in this paper (see [13] for further details). For example, it supports automatic layout of prototypes and/or their elements. It also supports prototype storyboarding through animating the navigational flow between the prototype screens by either highlighting the screens in order of navigation or by laying screens out on top of one another and flashing through in sequence. Also, ProtoMixer supports logging of design activities for undoing actions and supports importing and exporting of the design space contents.

A set of commands is built in to ProtoMixer to manipulate properties of regions. For example, there are commands to scale, move, select, clone and animate regions. Manipulating properties is important in constructing prototypes as well as in managing the design space. Commands can be grouped into scripts, stored and later recalled.

ProtoMixer utilizes a large display workspace, currently running on four 30" Apple Cinema Displays® each at a resolution of 2560x1600 pixels for a total of 16 megapixels. The high-resolution workspace allows for multiple designs to be worked on and for multiple designers to participate in the design session.

6 Example Mixed Fidelity Design Session

In this section, to illustrate the mixed fidelity prototyping approach, we look at an example design session for prototyping a business forecasting tool, used by businesses for estimating their revenue and expenses and ultimately profit for some future period.

Forecasting involves estimating a number of factors and with so many factors and possible values for each factor, it is most effective to take a visual approach where users can play through different situations and have the effects visualized in charts on the fly. In this example application, adjusting the factor value (x axis value) on charts causes a shift in the corresponding chart's profit curve as well as causes the underlying financial statements to be updated. Note that this application is for illustrative purposes and the mixed-fidelity approach is intended to generalize beyond this domain.

We start by sketching designs using an external drawing application. Note that we also could have drawn the designs on paper and then scanned them in as images. We then import these sketches into ProtoMixer as images and specify them as regions' assets (done through an XML input file). All of the sketches are then displayed in ProtoMixer in the Repository Region. Refer to Figure 7 for the initial state of ProtoMixer.

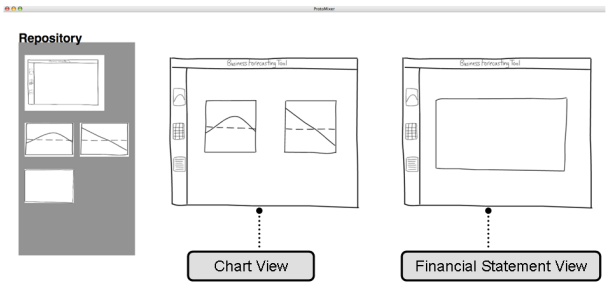

Fig. 7. Initial sketches of two views of forecasting data

(1) Mixing multiple fidelities in a single prototype We proceed with prototyping by adding further details to the sketches such as labels for the chart axes and pull-down menus for selecting cost and revenue drivers as well as the desired time period.

Next we want to give the sketched charts a more refined look, so we create medium-fidelity images of the charts using an external tool. We import these assets and overlay them on top of the sketched versions on the Chart View.

Now we turn to working on refining the table element in the Financial Statement View. We sketch out example data to put into the table and then import that new sketch. As we are confident the table will be used in the final design, we move it to high-fidelity. Specifically we create a new region and set its asset to be a JTable widget. This region is then overlaid on top of the sketched table. The prototypes created thus far are shown in Figure 8.

(2) Transitioning between the fidelities as ideas are refined After reviewing the prototypes at that point, we come up with a new design for more interactive charts which involves changing the appearance of the charts. Rather than starting from scratch, ProtoMixer allows us to turn off a layer to hide the images of the charts, thus reverting to the original sketched charts. Using ProtoMixer's built-in sketching feature, we update

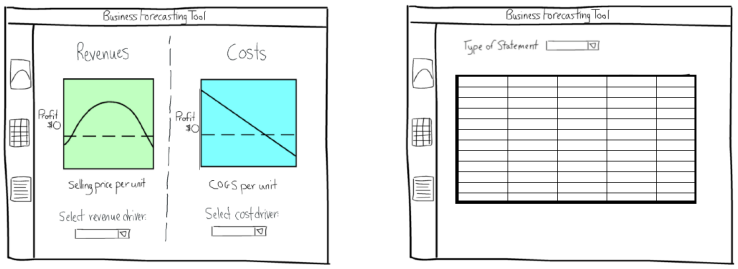

Fig. 8. Medium-fidelity charts and a high-fidelity table have been mixed on refined sketches

the sketches to include a vertical guide bar, to emphasize which x value the user has specified. Then we create and import medium-fidelity images matching these refined charts and overlay them onto the sketches. The resulting designs are shown in Figure 9.

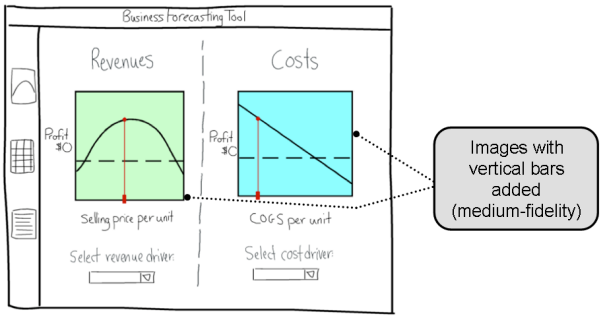

Fig. 9. 'Chart View' after adding updated images of charts

(3) Integrating domain-specific data and functionality We now make some high-fidelity updates to give the users a better appreciation for what the final software will be. First off, we add real data to the high-fidelity data component by setting the JTable's data and updating the region's asset to point to the new table. Then we insert a high-fidelity text field on top of the low-fidelity version. Next we connect the table's data to the profit field, as the profit's value is calculated based on the table data. This connection is done using the high-fidelity binding feature. From now on, modifying the table automatically updates the profit text field. The resulting design is shown in Figure 10.

(4) Exploring novel interactive elements Next we further explore the charts' behavior and interactivity. We want the x-axis of the chart to behave like a slider bar so the user can then slide through the different x-values and have the effects on the other chart visualized immediately by shifting the curve. Such a design is rather unique and is not implemented in any standard toolkit.

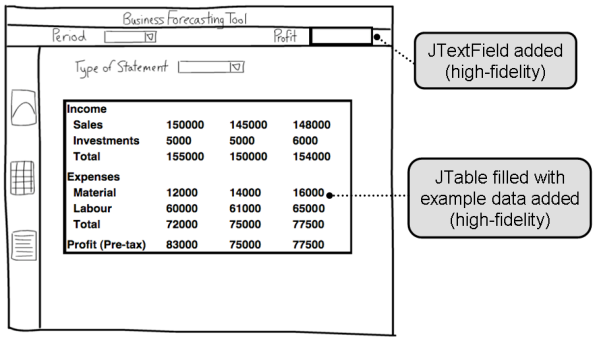

Fig. 10. 'Financial Statement View' after adding the high-fidelity text field and table with data

We start off by creating a series of medium-fidelity images that illustrate their appearance at some instance in time. Next we set these new charts as the assets of the most recent charts. We are careful to add the charts in sequential order to allow for animation of the chart's behavior. We now animate the charts by using the flash animation feature to see the resulting effects of moving sliders.

Now we need to move to a higher-fidelity design with these chart components, giving them some actual behavior. We work on the main chart area first. As Java's standard toolkit does not offer a chart widget, we implement a simple one on our own to look like the sketched charts. Then we need to use a slider widget for specifying an active x-axis value, so we make use of the JSlider from Java's standard toolkit. We overlay the chart on the sketched chart and then position the slider along the chart's x-axis.

Now that we have the key elements to create our novel chart component, we need to connect them together so a slider's value modifies the other chart's curve. We do this by binding the high-fidelity chart and slider components, again using the binding feature. Now we can see that dragging one chart's slider modifies the other chart and this can be demonstrated to other stakeholders. The resulting design is shown in Figure 11.

(5) Comparing alternative designs Now we want to explore an alternative layout for the prototype. We sketch out an alternate layout idea, which tries to incorporate the charts and financial statements in the same view so no details are hidden from the user. We import this sketch as well as a sketched version of our current layout and use the automatic layout feature to position the two designs side-by-side. We then want to annotate the two designs with pros and cons of each. We annotate them by creating two high-fidelity JTextArea widgets and type the annotations into the text areas. These annotations can then be associated with a particular design and hidden and unhidden as desired. The resulting state of ProtoMixer is shown in Figure 12.

6) Recording the design process Later on assume that a software developer is working on the full implementation of the chart components. As no standard component exists, the developer needs to get a thorough understanding of what this chart does and how

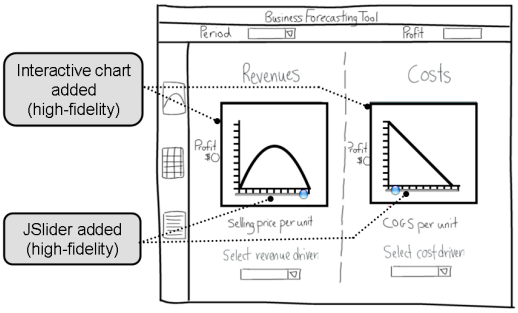

Fig. 11. 'Chart View' after composing a lightweight novel chart element

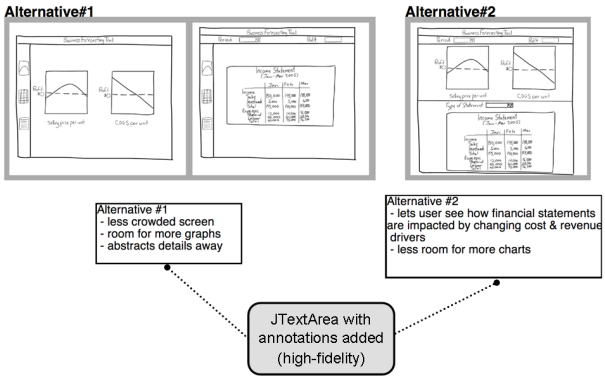

Fig. 12. Annotating the two alternate layouts with pros and cons

evolved into the current chosen design. The developer uses ProtoMixer to look back at previous versions, or more specifically, the evolution of the chart using the view versions feature. ProtoMixer then pops up a new region next to the specified chart with all of the versions. This state is shown in Figure 13.

7 Conclusion

This research presented a new approach to user interface prototyping called mixed-fidelity prototyping. Mixing fidelities allows designers the flexibility to focus on one specific aspect of a prototype at a time, by exploring that aspect in the various fidelities. In turn, our approach allows for designers to defer the exploration of less urgent issues, unlike current techniques and tools that heavily restrict designers in their workflow. For example, with Interface Builders designers are immediately forced to choose a layout and specific component types when composing a prototype. Our approach also allows individuals with expertise in a specific fidelity to be involved in that fidelity earlier on.

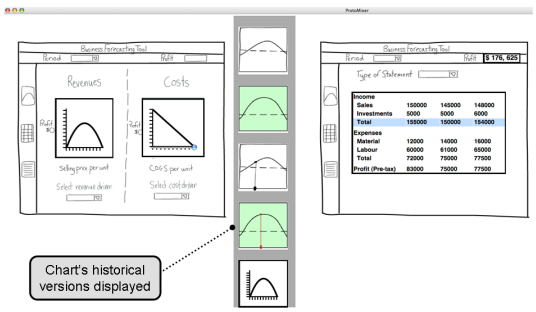

Fig. 13. Center region displays historical versions of the Revenue Chart

For example, a software developer can begin implementing a high-fidelity component for the prototype at an early stage.

Our approach also addresses some of the issues or shortcomings with the current prototyping practice: multiple fidelities may be explored at any given time, iteration may occur between any of the fidelities, user interface designers may better collaborate with each other and with other stakeholders, and potentially more innovative user interface designs may follow. Also, our approach adds continuity and traceability to the process by offering a single unified tool for prototyping in while allowing for designers to still take advantage of paper and whiteboard-based designs that they are familiar with.

ProtoMixer has several benefits over other existing prototyping tools. With ProtoMixer, prototypes can be composed of multiple fidelities, and elements are easily refined and transitioned between different fidelities. Individual elements can be tied into data and functionality, and can be executed inside prototypes. As well, traditional informal practices such as sketching and storyboarding are supported on the mediums that designers are accustomed to. Furthermore, ProtoMixer aids the designers by recording the history of the design process. Finally, ProtoMixer is designed for collaborative use on a high-resolution, large display workspace.

A secondary aspect of this research was to enhance the collaborative nature of prototyping by utilizing a large display workspace. Previous work has shown the benefits of using large displays in collaborative settings and in design settings in particular. Large displays seem ideal for this domain of prototyping, a visually rich, team-based domain. While we did not formally evaluate the benefits of using a large display workspace, it is evident that exploring multiple designs at once, whether it be exploring alternatives or exploring more than one screen design at once, requires more resolution than available on traditional-sized displays. Using tiled high-resolution displays to create a large-display workspace is conducive to our Regions Model and supports our approach well.

Preliminary use of the prototype has been positive, however, a next step in this research is to perform a field study to further evaluate mixed-fidelity prototyping using Pro-

toMixer. It may be interesting to compare the results of expert versus novice designers, and to evaluate the usefulness of ProtoMixer as a single-user versus a collaborative tool.

References

1. BAILEY, B. P., KONSTAN, J. A., AND CARLIS, J. V. Demais: designing multimedia applications with interactive storyboards. In *Proceedings of the ninth ACM international conference on Multimedia* (2001), ACM Press, pp. 241–250.
2. BLY, S. A. A use of drawing surfaces in different collaborative settings. In *Proceedings of the ACM conference on Computer-supported cooperative work* (1988), ACM Press, pp. 250–256.
3. BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C. M., MALER, E., AND YERGEAU, F. Extensible markup language (XML) 1.0 (third edition). W3 Recomendation available at: http://www.w3.org/TR/2004/REC-xml-20040204 - Accessed on November 3, 2005.
4. GUNARATNE, J., HWONG, B., NELSON, C., AND RUDORFER, A. Using evolutionary prototypes to formalize product requirements. In *Bridging the Gap II Workshop at ICSE* (2004).
5. HARADA, K., TANAKA, E., OGAWA, R., AND HARA, Y. Anecdote: a multimedia storyboarding system with seamless authoring support. In *Proceedings of the fourth ACM international conference on Multimedia* (1996), ACM Press, pp. 341–351.
6. JOHN, B. E., BASS, L., KAZMAN, R., AND CHEN, E. Identifying gaps between hci, software engineering and design, and boundary objects to bridge them. In *Workshop at CHI'04* (2004).
7. KAZMAN, R., BASS, L., AND BOSCH, J. Bridging the gaps between software engineering and human-computer interaction. In *Workshop at ICSE'03* (2003).
8. KAZMAN, R., BASS, L., AND JOHN, B. E. Bridging the gaps ii: Bridging the gaps between software engineering and human-computer interaction. In *Workshop at ICSE'04* (2004).
9. LANDAY, J. A., AND MYERS, B. A. Interactive sketching for the early stages of user interface design. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1995), ACM Press/Addison-Wesley Publishing Co., pp. 43–50.
10. LIN, J., NEWMAN, M. W., HONG, J. I., AND LANDAY, J. A. Denim: finding a tighter fit between tools and practice for web site design. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2000), ACM Press, pp. 510–517.
11. LIN, J., THOMSEN, M., AND LANDAY, J. A. A visual language for sketching large and complex interactive designs. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2002), ACM Press, pp. 307–314.
12. OUSTERHOUT, J. K. *Tcl and the Tk toolkit*. Addison-Wesley Longman Publishing Co., Inc., 1994.
13. PETRIE, J. Mixed fidelity prototyping of user interfaces. Master's thesis, University of Saskatchewan, 2006.
14. RUDD, J., STERN, K., AND ISENSEE, S. Low vs. high-fidelity prototyping debate. *interactions 3*, 1 (1996), 76–85.
15. TANG, J. C., AND LEIFER, L. J. A framework for understanding the workspace activity of design teams. In *Proceedings of Conference on Computer-supported cooperative work* (1988), ACM Press, pp. 244–249.
16. TVERSKY, B. What does drawing reveal about thinking? In *Proceedings of Visual and spatial reasoning in design* (1999), pp. 93–101.
17. WALKER, M., TAKAYAMA, L., AND LANDAY, J. A. High-fidelity or low-fidelity, paper or computer medium? In *Proceedings of the Human Factors and Ergonomics Society 46th Annual Meeting* (2002), pp. 661–665.