Gooey Interfaces: An Approach for Rapidly Repurposing Digital Content

Les Nelson, Elizabeth F. Churchill, Laurent Denoue, Jonathan Helfman, Paul Murphy
FX Palo Alto Laboratory, 3400, Hillview Avenue, Bldg 4., Palo Alto, CA 94394, USA
{ nelson; churchill; denoue; helfman; murphy}@fxpal.com

ABSTRACT
With the acceleration of technological development we are reaching the point where our systems and their user interfaces become to some degree outdated 'legacy systems' as soon as they are released. This raises the question of how can we maintain, extend, override, and adapt these systems while preserving what people depend on in them? In this paper we describe an approach for dynamically restructuring user interfaces into a set of communicating processes that 1) provide methods for changing their appearance, behavior, and state; and 2) report their proposed state changes so that other processes may override their actions in updating themselves to a new state. We do this for both new and wrapped legacy user interface components, thereby allowing us to repurpose user interfaces for our evolving needs. We describe how this approach has been successfully used in rapidly creating and deploying interfaces that repurpose content for new appearances and behaviors.

Author Keywords
Content repurposing; interactive public displays.

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Commonly used interfaces such as Web browsers can become entrenched in the practices and perceptions people hold about computer use. For example, Web content designers will at times depend on the many features of existing browsers (e.g., the content types supported by a browser, underlying data representations for specifying the display properties - the tags and parameters). Browsers and the supporting languages (e.g., HTML) were to some extent designed to be device independent by allowing each implementation to choose a layout for the content that suits it. Many attempts have been made to get around layout issues when the proposed device is too far outside the norm for usable content viewing in HTML [e.g., 1]. Less effort has gone into addressing issues of adapting behavior of interactive content viewed in a new environment. One such example is proxy servers, in which modifications are made to content as it is accessed for viewing according to a set of predefined rules [e.g.,3].

Environment dependency becomes a problem if we want to create new applications that retain some or all of the representational abilities of the older software, but have new interaction requirements (e.g., defining new ways links get followed while browsing content). This problem is especially acute in our work on the Plasma Poster Network [1]. The Plasma Posters (Figure 1) are large screen, digital, interactive poster-boards designed for informal content sharing within teams, groups, organizations and communities. Through the Plasma Posters, people encounter digital content (e.g., text, Web pages, free form scribbles, images, movies) that have been posted by other community members, and can choose to engage with that content further, or not.

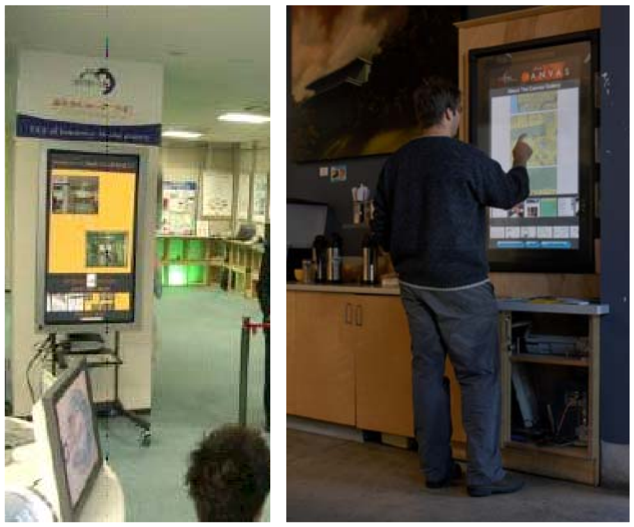
Figure 1. Plasma Posters being deployed in Japan and the U.S.

A CASE STUDY FOR THE DESIGN OF DIGITAL BULLETIN BOARDS

In over a year of successful deployment in an office setting, we have refined and evaluated display requirements for the posters:

1. *Main Capability.* When posted content and associated metadata (e.g., posting community member's name and picture, comments about the posting) is displayed on a Plasma Poster it must presented in large a large format where text is legible and textual hyperlinks are easy to select by touch.
2. *Navigation.* Users must have simple one finger controls for browsing and navigating postings. When no interactions are occurring, content display automatically advances after a customizable interval of time, which should be extended each time a user touches or otherwise interacts with the Poster.
3. *Interaction with Content.* Standard interactions with content should be preserved, provided these interactions do not disrupt the user's reading and navigation experience.
4. *Appearance.* To be attractive and attracting as a digital bulletin board distinct from workplace computers, we make the look and feel of the GUI as different as possible from a typical computer desktop without contradicting our users' intuitions about how typical interactions should occur.
5. *Adaptation.* We further require that the graphical user interface (GUI) architecture be flexible enough to allow us to change the look and feel rapidly, and if possible, to allow customizations with little or no rebuilding of the executable code.

We have implemented and evaluated a number of Plasma Poster interfaces to address, explore, and refine our approach.

A Web Only Implementation

The first prototype of the PlasmaPoster interface was implemented in Java and JavaScript. A single Web page was divided in HTML frames for posting title, main content and metadata, and a Java applet, which advanced the other frames to the next posting and accessed content using Java Remote Method Invocation (RMI) to communicate with the PlasmaPoster Network. This first prototype had some problems in keeping control over the displayed content:

- Some web pages opened new windows, altering our interface significantly.
- Some web pages detected they were being loaded into an HTML frame and automatically reloaded themselves in the top frame, completely removing our interface.
- The Browser security model caused problems for actions such as printing between frames hosted in different domains.
- We could not easily implement our own event logging for arbitrary user actions on Web page content.

An Ad Hoc, Standalone Display Application

In response to these issues, we developed a new prototype using Visual Basic (VB) and the Internet Explorer Web Browser Control to display the postings. Similar to what the applet did before, selectable buttons were added to let users navigate in the content. Printing was now easily introduced because directly hosting the web browser control in our application gave us full control of the interface actions (e.g., inhibiting the default print dialog boxes that cluttered up the public display interface). More importantly, using the VB web browser control allowed us to trap all events. We could now handle new windows opened with Javascript by some pages. As a consequence, any hypertext link followed opened a new browser window, which indicated to users that they are no longer navigating inside the posted content, but are instead navigating the Web.

This implementation was still not right. The Visual Basic prototype satisfied the display, sequencing, and operational requirements, but it was still not flexible enough to quickly experiment with new interfaces. For example, to change a button's location or style, we had to modify the Visual Basic code, apply a new style, and repeat the same process for all buttons to get a consistent look and feel. People's pictures were also displayed using the Visual Basic PictureBox control. Unfortunately, the PictureBox does not support animated Gifs. We were thus 're-inventing the wheel' already implemented in browsers and style sheets.

The Best of Both Worlds: A Layered Approach

On the one hand, a pure Web page solution would not allow us to maintain control over the representation of pages displayed in the main window, log user actions, or print without a dialog box. On the other, a 'proprietary' solution that gives greater control is less flexible and can not keep up with the evolution of capabilities for showing new content types.

We have thus implemented a layered architecture using two nested Visual Basic Web browser controls (see Figure 2). The first control displays the Plasma Poster brand, clocks, background, poster metadata, and buttons to control the slide show sequencing and poster operations. The second control is layered on top of the first and displays the poster content so it appears to be visually nested within the first control. We used HTML and Javascript as much as possible, but we used Visual Basic code to monitor and control the actions of the display controls, facilitate communication between them, and retrieve data from the Plasma Poster network (as it is awkward doing this using HTML and Javascript).

The layered approach provides the benefits of Web implementations:

- Because the user interface is described in an HTML file, non-programmers can quickly write new interfaces.
- Using a single HTML file for the entire user interface also allows us to use a single large background image to implement a different "skin" for each Plasma Poster display or, possibly for each author, or each poster. (i.e. imagine a food background in the kitchen and a more formal background in the visitor lounge).
- It is now much easier to experiment with different styles because HTML stylesheets can be used to specify the graphical attributes (e.g. font, size, color) of classes of individual elements.
- Many image formats can be displayed without having to add specific code as before.

New interface elements can be quickly implemented with Javascript, such as scrolling content overview displays, animating highlights to indicate that the current poster is about to change.

The layered approach required that we implement a special communication channel between the two web browser controls and the Visual Basic code that supports interface repurposing. Each component of the interface communicates what is happening to it, and allows its actions to be overridden. This is achieved through extensive use of dynamic manipulation of the content displayed in the browser controls and monitoring of their input and output event streams. An application level communication protocol has been defined to convey relevant interface state information of a component and allow others to control its actions. In effect we restructure the existing interface into bi-directionally communicating processes that would not normally be able to communicate.

Example Repurposing Using the Layered Approach

We illustrate the repurposing process with a specific example from the Plasma Poster operation: redefining content scrolling (Figure 2). We have replaced Web page scroll bars by direct manipulation of the content with the reader's finger: put finger on content and drag to scroll. Using an alternative scrolling mechanism allows a bigger target for the finger to hit (e.g., the entire page), makes the content look less like a desktop computer display, and provides a compelling visual attraction to the Poster when seen for the first time.

In Step 1 of Figure 2, a timer has expired in the Branding control, indicating that a new posting is to be shown. This event is communicated from the Branding Web Browser component to the Visual Basic Controller application by having the timer (encoded in Javascript) issue a Web page navigation with a special URL ("next://"). This event is trapped by the controller (Step 2), which access the Plasma Poster database server for the new content and its associated metadata.

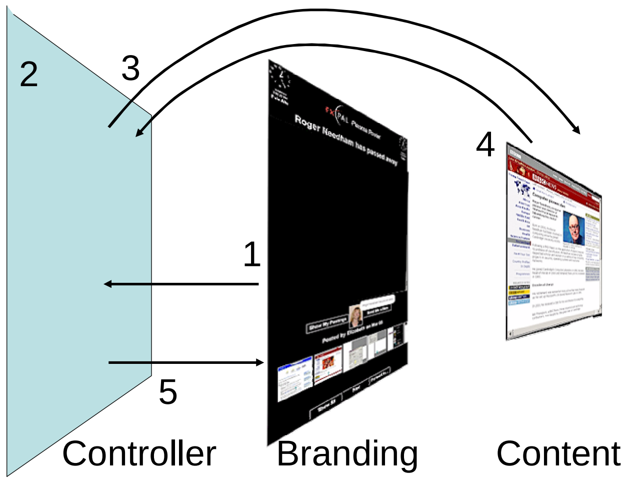

Figure 2. Layered Appraoch for Repurpurpspsoing in the Plasma Poster Interface.

The Metadata is communicated to the Branding component by calling routines in the Branding Web page that represents this part of the Poster interface. This is accomplished through accessing the Document Object Model (DOM) of the Branding page from VB. Instead of modifying the HTML elements directly through the DOM interface, we instead call Javascript functions embedded in the HTML page. For example, to set the title, the Visual Basic code calls:

```
webbrowserinterface.document.parentWindow.
execScript = "setTitle('new title')"
```

This is similar to an object-oriented approach in which the HTML interface is modified through setter and getter methods, allowing us to maintain a separation between the code that supports the look and feel and the code that supports the logic for controlling the sequencing and display of content.

The content (e.g., a posted URL) is accessed from the internet or local network by causing the Content Web browser control to navigate (i.e., executing the VB Navigate2 command). When the Content loading completes, the Controller accesses the DOM of the Content browser to include Javascript code that modifies how users can scroll the document. We overwrite the default OnMouseDown, OnMouseMove, and OnMouseUp events to let users scroll the document by touching and dragging anywhere on the web browser control that displays the content – essentially converting the entire visual representation of the document into a large invisible scrollbar.

When a reader scrolls the poster (Step 4), the new scrolling code executes the new behavior. This action also indicates interest in the currently showing posting. We thus need to reset the timers that automatically advance the display to its next posting in the sequence. To do this the Content HTML document has also been modified to trigger an operation in the Visual Basic Controller. We use a special communication protocol designed for this by having the Content component issue a navigation to a special URL:

'touch://x=100 &y=200 &callback=Pause'

This URL is not a valid Web address, but because the Visual Basic application is notified before any navigation occurs, it has an opportunity to cancel the navigation and analyze the URL to determine what action needs to be taken. The callback method, Pause, indicated in the Branding control is thus invoked from the Controller. Note that in this way, two Web pages that previously could not communicate, may now know and trigger state changes in the other, thereby collaborating in the overall repurposing of each.

While simple, this mechanism has enabled us to implement all communication from HTML to the Visual Basic code. The general structure of the repurposing messages is,

action://command?parameter=value,

which we have found sufficiently expressive for all the interface adaptations we have so far needed.

USE AND EVALUTION

We have used this technique to build, deploy, and further evolve a network of the Plasma Posters in an office space.

We have found that a new poster instantiation can be created in about 2 weeks of effort involving a graphic designer. Testing and evolution of the design in place with the intended users of the system usually lasts about another two week. Changes are primarily made to the base Web pages invoked by our controlling VB application. These changes are adding buttons and graphical elements and arranging elements. Changes to the VB code itself are required when we define a new concept, such as a new kind of thumbnail pane that requires new data storage and retrieval methods. For example, we added the capability for people to leave comments in the form of free form graffiti. These are then saved for others to browse and display on the main content display. This change involved changing the VB code to invoke a new instantiation of our free form scribble application and store the resulting bitmaps on disk. Such changes are primarily copying and slightly modifying code that already exists.

CONCLUSION

The presentation technology described here creates malleable, easily manipulable interfaces, in effect "Gooey" GUIs. These use a general strategy of encapsulating previously developed user interface elements in the following manner:

- We add new controlling processes that access and modify existing interface components. This is accomplished by 'wrapping' the other components as in ActiveX controls, as illustrated above to Internet Explorer. This may be generalized for use with other forms of communication (e.g., remote procedure call, Web services, or some other form of application programming interfaces).
- We augment existing display interfaces by instrumenting content viewed in an interface by accessing the underlying data model of the content being displayed . We have shown here how this may be done with the Document Object Model of Web pages. Augmentations may also take the form of new scripts, macros, method overriding or other forms of state change in the interface. The technique of instrumentation is well known. What we introduce here is the addition of application level communication protocols into the content for the purposes of redefining the presentation appearance and behavior.
- We use application level repurposing communication protocols to create bi-directionally communicating processes that redefine the original interface. In effect, we create a Model-View-Controller structure that allows components to be adapted and allows delegation of selected interface actions to other components with different capabilities (even when such communication was not originally designed into the original implementation).

We are now working on three new Plasma Poster systems: an office space in Japan; a network between offices in different time zones; and in a café/art gallery. We are in the process of adapting our work to create a "Plasma OS", a development environment that others may use to create repurposed content suitable for attractive and attracting public displays. Although our motivation for developing this repurposing technique was from public displays, our method generalizes to other devices and display types (e.g., PDAs). We see the strength in our method as deriving from a concern to give equal attention to interface aesthetics, usability, and technical elegance.

REFERENCES

1. Bickmore, T., Girgensohn, A., and Sullivan S., Web Page Filtering and Re-Authoring for Mobile Users, The Computer Journal, 42 (6), pp. 534-546, 1999.
2. Churchill, E.F., Nelson, L. Denoue, L., Helfman, J., and Murphy, P., Sharing Multimedia Content with Interactive Public Displays: A Case Study, Proceedings of DIS2004 Designing Interactive Systems, Cambridge, MA, USA, 1-4 August 2004.
3. Knutsson, B., Lu, H., Mogul, J., Hopkins, B., Architecture and performance of server-directed transcoding ACM Transactions on Internet Technology (TOIT), Volume 3 Issue 4 , 2003