

Repurposing: Techniques for reuse and integration of interactive systems

Les Nelson and Elizabeth F. Churchill

Abstract—This paper explores aspects of reuse and intergration of existing applications. We introduce an approach for redefining applications for use in new user settings not accounted for in the application’s original design process. Based on our previous research into novel interfaces for supporting communication and collaboration, we identify six reuse and integration techniques we call *repurposing* and show how they may be used in a design extension of a standard email client that preserves its familiar operations.

I. INTRODUCTION

With the acceleration of technological development we are reaching the point where our systems and their user interfaces become outdated to some degree as soon as they are released.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? Further, end-users are increasingly coming to expect that they can adapt, tailor and extend applications to suit their needs.

However, designs are explicitly or tacitly formed around *particular* notions of use that the developers then cast into a system. For example, agile development methods explicitly involve ‘customers’ of a technical solution in defining user scenarios that directly influence each build of a solution. Unanticipated behaviors or expectations are handled in redefinition of scenarios and from these a refactoring of solution. But at some point iterative development ceases, the tool “freezes” and does not change in response to user needs. Thereafter if users desire new capabilities, they have to be creative about using the tool. The most recent example of this can be seen in mashups (e.g., www.mashupcamp.org), where users who can envision novel functionality, extend available tools in the permitted ways to satisfy their needs. But mashup examples are limited; most software environments and applications support complex task models, and are not so readily manipulated.

When applications are creatively combined and/or extended (e.g. adding a visualization capability that draws on a non-native data source), the resultant system can be brittle. Adapting or manipulating existing elements of an application can have knock-on effects given the (often unseen) dependencies between system components. For example,

removal of a PC application or running of a daily system update can render other programs or the system temporarily inoperable. It has long been held (e.g., [1]) and recently been expanded upon in the Open Source community [5] that stability may be improved in complex system development by designing modular solutions consisting of loosely coupled, highly cohesive sub-parts, where “designers try to minimize and standardize the interdependencies among modules”. However, such solutions tend to be in homogeneous settings where we can rebuild the systems to new requirements (e.g., SmallTalk Model-View-Controller implementations, well-maintained Open Source development environments). In many cases we, are not free to abandon old systems. It takes maney and resources to change technology as well as practices (e.g., even though we might prefer a more capable email client or word processor, it generates work to be out of step with the supported or prevailing application).

In this paper we explore ‘repurposing’, our name for reuse and reintegration. Here, interactive content is adapted or re-appropriated for use into social and/or technical situations that differ from the original design context, while preserving the operation of the existing resources and practices. We consider content ranging from specifically authored materials such as media-rich Web pages to dynamic content such as information presented in interactive applications like communication and collaboration support or enterprise management systems.

We describe specific instances of how the need for repurposing arose in our research into novel forms of human-computer interaction and computer supported communication and collaboration. From the various approaches for repurposing used in our work and elsewhere, we suggest a socio-technical design approach that honors existing systems and their user and software interfaces and implementations while allowing the re-representation of the input, processing, or results to accommodate new uses. Our method involves 6 operations that lead to the restructuring of applications without needing to reimplement their internals. Typical solutions rely on source code, or are constrained by what the API provides so customization is limited. Our methodology allows developers to work through and around these constraints; we illustrate our approach with examples.

II. EXAMPLES OF REPURPOSING

The area of support for communication and collaboration (e.g., Computer Supported Cooperative Work) provides some good illustrations for the needs for repurposing. Technical

Manuscript received May 1, 2006.

Les Nelson is with the Palo Alto Research Center, Inc., Palo Alto, CA 94304 USA (phone: 650-812-4716; fax: 650-812- 4258; e-mail: Les.Nelson@parc.com).

Elizabeth F. Churchill is with Palo Alto Research Center, Inc., Palo Alto, CA 94304 USA (e-mail: Elizabeth.Churchill@parc.com).

solutions in one communication setting may be inappropriate in the face of seemingly simple changes in physical and social settings (e.g., changing the orientation and intended audience of a display). We consider recent designs in three areas to illustrate this point and require adaptation and re-integration of existing technical solutions. These activities cover a range of environments: public display of interactive media, re-working the animated behavior of desktop displays, and creating a new form of slideshow presentation tool by using a tangible user interface. Note that the first and last of these efforts lead to products, and the other prototype was used to test the limits of the prevailing desktop environment.

A. Working with Interactive Multimedia

The Plasma Posters (Figure 1) are large screen, digital, interactive poster-boards designed for informal content sharing within teams, groups, organizations and communities [2]. People interact with digital content through physical contact with the board (i.e., touch overlay).



Figure 1. Interactive public displays show content designed for Web browsing on a PC, but placed into an entirely different use setting.

Content shown on the displays (e.g., text, Web pages, free form scribbles, images, movies) is posted via email to the Posters by other community members, and can choose to make reply to the content via email from the public display. The Plasma Posters have been deployed in various forms in public and office settings since 2002.

The design of these systems involved moving the viewing of content originally authored for browsing on a personal computer (e.g., with monitor, mouse, keyboard, sitting down and ‘leaning forward’ user experience) to a functionally similar but experientially dissimilar context (i.e., large, portrait oriented, wall mounted, touch sensitive screen, and, standing or ‘passerby’ user experience). In this situation, there were a number of things ‘wrong’ with our browser of choice, in this case Microsoft’s Internet Explorer (IE). However, we did not want to switch to a different browser implementation because users expect content that is shown on IE to behave in a certain way (that is in part defined by the Web standards and by the Web technology implementer, Microsoft, who also adapt their application look and feel). Not only is it impractical to keep up with all changes to all content types supported in IE, this would also be a bad case of ‘re-inventing the wheel’, wasting development resources to

problems that are in some sense solved (even if somewhat problematic in its consequences).

In the case of the Plasma Poster displays, we observed that users found some browser responses annoying, greatly taking away from a user experience that was meant to be attractive and attracting.

Web pages appear to ‘flicker’ upon page refresh. This was perhaps more relevant in the time before new capabilities such as AJAX became available; however, not all content has been expressed and re-implemented in these new forms.

The ‘pop up’ feature of a browser is not conducive to large displays that are supposed to be ‘posters’, not computer desktop windows that just happen to be mounted on the wall. Pop-ups are annoying, but simply blocking them may prevent access to relevant content. Asking users if they want to unblock content might be reasonable in browsing sessions on their own PC or laptop; however this is added work for a passerby to a poster. Users who walk up and leave a poster are not likely to want to digitally clean up these extra windows after browsing. Some navigation behavior (i.e., navigate in new window) provide similar issues to pop-ups, and also get triggered under content author control rather than the user’s.

Scrolling content using navigation bars can be awkward on a large screen display due to the difference in accuracy between a mouse and a finger on touchscreen. While scroll bars can be made bigger, this is not a very pretty option and reduces the available content area a bit. Posters in public spaces are not simply functional units for delivering content, but are also meant to be attractive.

The Plasma Posters also have added capabilities not commonly associated with PC viewing practice. As suggested in this last example on scrolling, redefined renditions of old capabilities need to be linked with new ones, together providing a different reading experience.

The display cycles through the list of posted content one posting at a time. A person reading content will want this cycling paused while reading. Thus, we chose a simple action that if for any reason one touches a poster it should pause from automatically advancing to new content. This requires re-interpreting the usual click actions (e.g., link following, or a click having no effect),

The poster layout associates and rearranges metadata with the currently displayed content (titles, thought bubbles indicating attached comments to the postings, thumbnails of the postings to aid in browsing and navigation of what has been posted to the poster). This association requires the linking of the IE instance that displays the posted content, with the presentation layer for the metadata. We wanted a consistent, adaptable, and expressive medium for the metadata and, hence used a different instance of IE for this data, thereby producing the need for two web pages to synchronize and communicate. An advantage to such use of web pages is that greater customization is allowed through using different stylesheets or simple edits to the Web syntax to re-express the appearance of both content and meta-data.

We allowed highlighting (through yellow highlighter markings) of specific items of interest on a posted page. The posting email designates the text of interest. Highlighting is added dynamically by the poster client by manipulating the underlying document model of the posted content.

Finally, we needed to integrate in capabilities that were not related to the user experience. We log all clicks on content, thereby requiring we override the usual click actions of the underlying browser. All clicks with metadata and other poster interface elements (buttons, logos), are also logged and associated with the posted content running in a separate browser instance.

B. Animating Graphic Displays

In the AttrActive Windows prototype ([3]), the goal was to provide a different, compelling, but functional look and feel to a display that would invite people to grab the display objects and manipulate them in simulated 3D. We chose to model a digital desktop much like a physical one, while maintaining all the application interactions. Figure 2 shows the mapping of an unmodified desktop window containing any running PC application. Its visual appearance is mapped onto a three dimensional representation with simulated physical dynamics (e.g., folding by a user, flapping in a simulated breeze). User interactions on the mapped representation are quickly translated back to the original window coordinates by using values stored in the texture mapping data structure. In this way, all the application level processing (e.g., button pushes, form fills, link navigation) are preserved, but a whole new set of dynamics is applied to the object as shown on the screen.

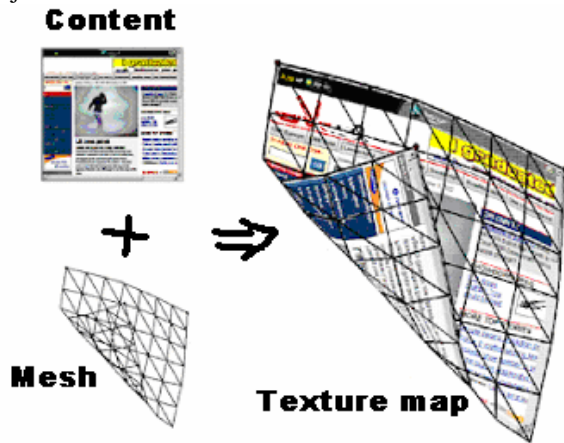


Figure 2. An entirely new set of physical dynamics is applied to 2D PC window applications to create 3D representations that perform exactly the same application processing in the same way.

C. Creating Tangible User Interfaces

Lastly, we considered an example arising from a very different domain than public and desktop displays: tangible user interfaces (TUIs). A tangible user interface is one in which a person interacts with digital content through a physical representation that in some way is related or corresponds back to the digital representations.

The Palette ([6]) is a TUI that assists a presenter in the task

of showing multimedia content to an audience (e.g., Microsoft Powerpoint presentations). Figure 3 shows a presenter manipulating physical cards to control the projection of the corresponding digital slides (e.g., swiping the barcode on a card shows the slide). The Palette presentation interface replaces the Powerpoint Slide Show view by using an alternative input stream taken from a physical environment sensor (i.e., laser barcode scan data arriving at the presentation PC's serial port). The features of Powerpoint interface are re-mapped to correspond to card scanning activities (e.g., show next slide), but are also modified for new modes of expression that the physical objects provide (e.g., show any slide from any presentation that the presenter has represented as a card). In fact, the purpose of the TUI approach for redefining the slide show interface was to make these new modes of expression available with very simple physical actions that make little explicit attention demand on the presenter.

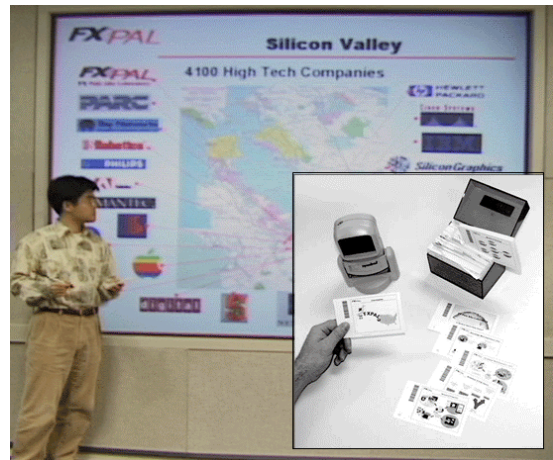


Figure 3. Physical cards that correspond to slides in a multimedia presentation are used to aid a presenter in controlling the display of information to an audience.

D. Issues in Reuse and Integration

The above examples represent specific instances of repurposing for specific design situations, but illustrate a range of issues that come up in system design, including:

- Modification of the appearance of displayed objects;
- Overriding the behavior of an object in response to input or changes of state (e.g., different visual dynamics or overriding existing processing).
- Changing the communication interfaces between objects, either by creating a communication path that was previously not there, removing or filtering an information flow, or redirecting communications elsewhere (e.g., allowing separate Web pages to interoperate).
- Changing persistence of data, capturing formerly ephemeral information for later use (e.g., logging).
- Changing the modality of input or output (e.g., using gestures for input, paper for output representations)

III. REPURPOSING HEURISTICS

These examples of reuse and re-integration arising from research also provide some good illustrations for the solutions applied to repurposing.

The Palette interface uses Microsoft ActiveX controls to encapsulate or *wrap* the Microsoft Powerpoint application so we may communicate with it through a standard interface (i.e., data to be passed, functional interfaces to communicate that data). This enables an *input replacement* where new controlling software is introduced into the system to accept the alternative input stream and interpret those inputs as control specifications for the wrapped application.

In AttrActive Windows, an existing application is hidden and runs in parallel with a new application. The new interface implements *transformations* of both static appearances and new dynamic effects on those appearances. The new application is itself wrapped so that the input stream from the user may undergo an *inverse transformation* and stream back to the original target application to provide the selected processing behavior.

The Plasma Poster interface is perhaps the most technically elaborate example given here. In this case IE is wrapped, a controlling application is created to create new supporting instances of IE for alternative processing of the content. A *repurposing protocol* is used to communicate events, actions, and callbacks between the re-distributed interface processes (original and controlling components of the system). To support communication between the parts of the system that involves the direct interaction with repurposed content, the *underlying content model* is accessed and manipulated (e.g., links are changed, supporting code is added to alternatively process mouse clicks)

The implementations described above provide a set of six heuristic approaches for creating new interactions by overriding and extending existing interfaces. Repurposing may involve some combination of the following capabilities:

- Wrap the software to trap its events and communicate with its functions. (e.g., Application Programming Interfaces, APIs, and use of components such as .NET and ActiveX).

- Manipulate application hosted content (e.g., Document Object Model modification, content transformations such as XSLT, server side processing such as PHP, JSP, ASP, and use of application macros, and scripting).

- Introduce control restructuring and repurposing protocols (e.g., Plasma Poster and AttrActive Windows) to allow an implementation to be re-distributed and have interaction state passed between the newly reorganized elements so that new processing rules may be applied.

- Provide input replacement where the input stream is taken from an alternative source (Palette).

- Transform (and provide inverse transformation methods as in AttrActive Windows). This approach enables an output replacement that preserves correct responses in the interaction.

- Finally, customization through profiles, stylesheets, and

parameterization is a mainstay of creating adaptable and reconfigurable applications. These constrained and largely pre-planned adaptations combine with the repurposing methods above to further enhance their capabilities for adaptation (e.g., Plasma Poster).

IV. ASSESSING THE HEURISTICS

This prior work in prototypes, deployed systems and products suggests the technical soundness of the various repurposing approaches described here. We further explore this assertion by considering how we might reuse and re-integrate a fundamental work tool such as an email client.

Much has been reported about how the user practices around the 'inbox' interface and email messaging protocol have been appropriated beyond simple asynchronous exchange of text and images (e.g., email as a "habitat" []). Recent work in visualization (e.g., Themail,[7]) considers alternative representations for reading email content. Themail presents email content through a series of columns of keywords arranged along a timeline. Keywords are shown in different colors and sizes depending on their frequency and distinctiveness as determined by text analysis of the message content. This abstracted representation of the messages gives a quick overview of the key topics being discussed over time, thereby aiding in reading and browsing .

The Themail visualization is basically a generated report of the email content that may be browsed, but is not integrated with the application view of an email client. This presents an opportunity for exploring a repurposing; namely, how can one construct a "mashup"-like view of a standard tool that integrates with its functionality seamlessly? A mashup approach has potential for avoiding the difficulty in influencing an existing application development process and product team to change an established application such as an email client ([8]).

In the remainder of this section, we consider how a popular email client, namely Microsoft Outlook, can be repurposed so that new interfaces for alternative uses are made available (i.e., supporting an interactive view of email content provided by a text analysis, as in the Themail example).

A. A Brief Summary of an Email Client

An email client such as Microsoft Outlook (Figure 4) provides a set of application views (e.g., mail, calendar) that consist of collections (e.g., mail folders) of objects of interest (e.g., mail messages) that may be selected and manipulated. The tasks supported for email include reading, searching, filing, forwarding, replying, deleting, and processing messages (such as printing attached content or making calendar entries from messages).

B. A Repurposed Email Interface based on Text Analysis

Consider an alternate view of email that focuses on implicit relationships in the email content, such as through text analysis. Such a view permits a new set of possible uses. For example, the Themail interface shows keywords of interest

called out by word frequency in messages and arranges them over time. Browsing of such representations suits different styles and purposes of reading, including more reflective questioning about what is the tone and theme of correspondence one has with another individual and how that has changed. More generally, with text analysis we may change the selection criteria for text (e.g., ‘pick lists’ of words to include in the analysis, ‘stop lists’ of words to exclude) and the type of text processing (e.g., frequencies, nearness of terms, similarity of messages) to produce many possible visualizations in this mode of inquiry.

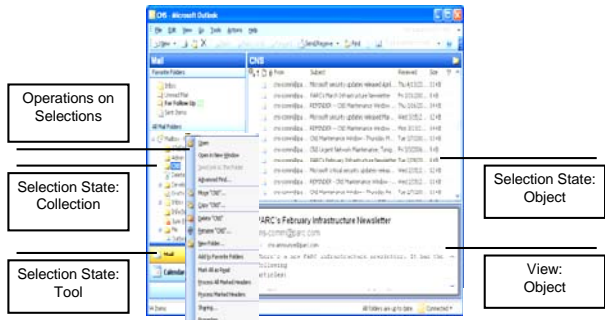


Figure 4. An email client allows tool, collection, and object selections, permitting operations to be performed on each.

Suppose we wished to create a new email processing that performed the following operations: count the occurrence of a set of pre-selected relevant keywords in messages sent to a particular email distribution list and show these over time in descending order of use. This arrangement would at a glance give members of the distribution list an idea of the importance of the selected topics over time in the list. This arrangement invites specific searches into selected time periods where activities around a topic are most intense.

Figure 5 shows such an example arrangement drawn from mailings to a project distribution list. Greater occurrences are marked by larger fonts. Interpretation of what was actually occurring is not obvious, but this representation would likely invite one to want to know why a moderate interest in working groups and workshops at the end of the year became dominated by meetings and planning in February.

	Dec-05		Jan-06		Feb-06	
Working Group	7	Project	10	Meeting	30	
Meeting	6	Meeting	8	Project	23	
Presentation	6	Plan	8	Plan	19	
Solution	5	Workshop	7	Summary	5	
Workshop	4	Discussion	4	Training	4	
Brainstorming	3	Analysis	3	Discussion	3	
Discussion	3	Discussion	3	Approval	2	
Proposal	3	Solution	2	Draft	2	

Figure 5. Arrangement of emails by keyword occurrence permits a different view into the underlying activity than an InBox or thread inspection.

For purposes of grounding the discussion on repurposing issues, we will assume the following use situation. Assume that a person wants to make use of only the mail messages in the InBox (or appropriate subfolder) relating to project related emails sent to a single distribution list. Suppose a

number of project staff (say approximately 20 heavy posters to the DL and 60 people overall included in the list) are communicating. We further assume the InBox will remain open, along side of a keyword-based, text analytical interface such as shown in Figure 5. The purpose of the keyword interface is to give overview and allow for targeted exploration of all the project emails in a more informed manner than simple keyword search or sequential, sorted listings.

This interface arrangement suggests possible user actions on the keyword view, including the following:

- Selecting one or more keywords;
- Choosing menu actions based on current selections;
- Changing time frames;
- Querying available keywords; and
- Frequent switching between these alternate views of the messages.

Here the keyword-based interface provides a synchronized support view for the normal InBox, thus maintaining consistency but not requiring replication of InBox functions.

C. Re-mapping the Interfaces

The existing interaction of Outlook defines a set of user expectations in terms of selection and operation that a new view should either preserve, or in some manner indicate its difference in a non-contradictory way.

Selection consistency is maintained between the existing views (e.g., InBox and keyword interfaces) such that any selection in one view may be reflected back to the other. A view such as Figure 5 provides an alternative listing of selectable objects that are related to the mail system objects (e.g., folders, messages, contacts). Possible access methods and operations on the underlying email messages, including the following:

- Selecting groups of messages for further browsing and queries by selecting keywords in the keyword view.
- Highlighting keywords when one or more folders or messages are selected in the InBox view.

Operating consistency is maintained by keeping consistent state between views, including:

- Updating the keyword view when new messages arrive or existing messages are read, filed, archived, or deleted from the InBox;
- Propagating operations selected from the keyword view back to the email client application;
- Indicate unread messages, importance, subject, and how messages are directed (i.e., from/to).

D. Applying the Re-purposing Heuristics

The demands of selection and operational consistency suggest a synchronizing component is needed between views. In the case of the Outlook email client and the keyword view of Figure 5, such a component would be defined as a control restructuring with the following operations:

- Monitor state of the InBox for new, read, deleted, and filed messages (e.g., by polling or subscription) and update

the text processing and attributes (e.g., colors, bolding) accordingly in the keyword view. Periodically interrogating the InBox state would be a loosely coupled way of keeping the keyword view up to date, with the addition of the polling latency. Alternatively, having the Outlook client broadcast its changes as they happen would allow close synchronization between views, at the cost of having to override processing on each of the Outlook events of interest (e.g., NewMail, Folders, and Items events).

--Monitor message and folder selection in the InBox view and update the text processing, selection, and attributes (e.g., colors, bolding) accordingly in the keyword view.

--Propagate operations in the keyword view and invoke a corresponding processing on the corresponding Outlook objects. For each selected object, locate (e.g., Items.Find, or traverse the item objects) the corresponding messages and invoke the mail operation.

--Propagate selections in the keyword view to the corresponding objects in Outlook. Here the design becomes more complicated, due to inherent limitations of the existing applications. In this case, Outlook 2002 does not allow programmatic manipulation of the selection data structure. For example, selecting the keyword 'Meeting' in the 'Feb-06' column of Figure 5 would correspond to the selection of the 30 Outlook message items. If these messages are in a single folder as described in the use situation above, this corresponds well to the InBox folder structure. However, we would need to manipulate the mouse events to invoke the selections (as in the inverse transformation of the AttrActive Windows example above).. Otherwise, a more complicated correspondence could be considered such as creating an Outlook Form to gather messages in one window.

This last example points out a design issue in repurposing. Adding new views can add new affordances that are not supported directly in the original design. Careful design and trade-offs become necessary when the existing underlying model is both hidden and its operating assumptions changed.

We have implemented a control restructuring that uses the email stream as the communication mechanism between the InBox and keyword views. The selection and operation commands are passed back as an email message, that is intercepted (Application_NewMail) and processed in the scripting environment of Outlook.

Further refinements to improve latency and handle issues such as the selection issue described above are ongoing. With respect to our repurposing approach, this first cut solution does not use all of the possible repurposing approaches described above. We could for example, extend the solution using these approaches in the following ways:

--Provide transformations to the keyword view based on text summaries of the message subjects or content and overlay these on the keyword table (e.g., 'pop ups' or 'tool tips').

--Implement clicking on the various keywords as an invocation of operations to file or tag these messages. This

involves content manipulation at the levels: of content of folders and meta-data on message forms.

--Input replacement in this situation suggests the use of alternative devices (e.g., mobile phones, PDAs) to mark or otherwise organize mail for later processing.

V. CONCLUSION

The desire to substantially redefine existing applications may be seen in recent trends such as software mashups. From the various approaches for reuse and reintegration in a range of work considered here, we suggest six technical design techniques that honor existing systems, while allowing the re-representation of the input, processing, or results to accommodate new uses. When coupled with multi-disciplinary, iterative, and user-centered design, we believe the resulting socio-technical design approach will better allow systems evolution within the ever changing use situations these systems operate within.

We see repurposing as a step towards designing modular solutions consisting of loosely coupled, highly cohesive sub-parts, and doing so while preserving the features in applications that users come to depend on. Our future work in the area around repurposing protocols involves enhancing the ability for software components to communicate their state and accept guidance in their operations. We are pursuing this currently by reshaping existing APIs such as in the email example considered above to support such protocols.

ACKNOWLEDGMENT

We thank the FX Palo Alto Laboratory for support of a succession of separate research activities revisited here.

REFERENCES

- [1] Booch, G., Object Oriented Design with Applications, Benjamin/Cummings Pub. Co., 1991.
- [2] Churchill, EF, Nelson, LD., Denoue, L, Helfman, J, Murphy, P., Sharing Multimedia Content with Interactive Displays: A Case Study, ACM Proceedings of the 2004 Conference on Designing Interactive Systems, ACM Press, 2004.
- [3] Denoue, L, Nelson, LD., Churchill, EF, A fast, interactive 3D paper-flier metaphor for digital bulletin boards, Laurent Denoue, Les Nelson, and Elizabeth Churchill, Proceedings of the 16th annual ACM Symposium on User Interface Software And Technology, ACM Press 2003.
- [4] Ducheneaut, N., & Bellotti, V. (2001). Email as habitat: an exploration of embedded personal information management. *Interactions*, 8(5), pp. 30-38.
- [5] Narduzzo, A. & Rossi, A., Modularity in Action: GNU/Linux and Free/Open Source Software Development Model Unleashed, <http://opensource.mit.edu/papers/narduzzorossi.pdf>, May 2003.
- [6] Nelson, LD., Churchill, EF, Denoue, L, Helfman, J, Murphy, P., Goocy Interfaces: An Approach for Rapidly Repurposing Digital Content, CHI '04 Extended Abstracts On Human Factors In Computing Systems, ACM Press, 2004.
- [7] Viégas, F.B., Golder, S., Donath, J., Visualizing Email Content: Portraying Relationships from Conversational Histories, ACM Proceedings of the 2006 Conference on Human Factors in Computing, CHI2006, ACM Press, 2006.
- [8] Wattenberg, M., Rohall, S. L., Gruen, D., & Kerr, B. (2005). E-mail research: Targeting the enterprise. *Human-Computer Interaction*, 20, 139-162.