DesignMinders: A Design Knowledge Collaboration Approach

Gerald Bortis and André van der Hoek

University of California, Irvine
Department of Informatics
Irvine, CA 92697-3440
{gbortis, andre}@ics.uci.edu

Abstract. Software developers face numerous challenges in capturing and reusing knowledge during informal design sessions. While the knowledge that is brought to bear and generated during these sessions provides valuable insights that can ideally be reused, informal software design is a time when developers are gathered at the whiteboard working to solve problem and during which the developer must remain "in the moment" while engaged in discussion and sketching. In this paper we describe DesignMinders, a tool currently being developed to address these issues by augmenting an electronic whiteboard with the ability to capture, refine and explore design knowledge in the form of notecards. This paper documents our progress and describes several key challenges that we face.

Keywords: Software design, knowledge reuse, whiteboards, notecards

1. Introduction

Consider a team of software developers starting a new project. They hold a meeting to discuss the design and to look for some inspiration from solutions applied in other projects. They are not looking for a specific approach or technique; they just want to get a general idea of what others have done or problems they have encountered when working on similar projects. What if they had a way to easily explore this information?

Consider another group of developers gathered at a whiteboard in a meeting room working on a design problem. As they work through a potential solution, one of the developers recalls a certain constraint that had to be worked through the last time a similar problem was encountered. None of the developers can recall it, and the meeting stops. What if they were using a system that could automatically provide them with this information?

Finally, consider yet another group of developers, in the midst of session, when they realize that they are making some crucial decisions about the design that they would like to keep for later. What if they could easily capture these decisions, and have them presented when needed during subsequent sessions?

This is the space on which our research is focused: supporting software designers in capturing and using design knowledge in a lightweight way when they are designing at the whiteboard. This paper presents a vision of how this might be done, documents our progress towards it, and discusses several key challenges that we face.

2. Background

The above scenarios highlight obstacles that are commonly encountered by developers during informal software design. This is a time when developers are gathered at the whiteboard to better understand and work through a design problem. It is an activity that spans the development lifecycle and during which developers bring to bear knowledge from past projects or personal experiences that influence the decisions that are made [1-3]. The knowledge involved is at times explicit, easily conveyed, and about general software development practices, solution patterns, or a particular application domain. It can also be tacit knowledge that is specific to the organization or project [4]. As this knowledge is applied to the design, new knowledge is generated in the form of discussions and artifacts on the whiteboard that can provide valuable insights into how the system was designed; insights that in the ideal world are easily recalled during subsequent design sessions.

Unfortunately, as in the above scenarios, most of this knowledge "vaporizes" and fails to be reused after these design sessions, since it is difficult to capture and represent using existing approaches [5]. Informal design is a time when developers are engaged in ad hoc sketching and discussion, and are unlikely to follow a formal or prescribed process. It is crucial that developers be able to capture knowledge in a lightweight and informal way using a representation that allows for the knowledge to be refined at a later time. The developers also must be able to explore the collected knowledge and have it presented in a relevant way.

Existing techniques for capturing and reusing this design knowledge either fail to address these challenges altogether, or are focused on capturing knowledge during the more formal specification or implementation phases of the development process, when the informal design knowledge has already been lost. For example, knowledge management systems that allow developers to populate and search a knowledge base containing formal design documents fail to address the need for concise and relevant representations of knowledge during informal software design sessions when the design problem is still being understood. On the other hand, design rationale techniques attempt to capture the decisions that were made during a design session using an argumentation schema that is incompatible with the activities and processes that developers engage in while at the whiteboard. Such techniques are difficult to apply during informal software design when developers are engaged both with the whiteboard and with each other in discussion, and are unlikely to interrupt an opportunistic exploration of the design problem to populate the knowledge base with recently acquired knowledge about the design problem, or to formalize their partial solutions into an argumentation schema [6].

3. DesignMinders

To address these challenges, we present DesignMinders, a software design knowledge reuse tool to support developers in: capturing design knowledge in an easy way, organizing the collected knowledge as a set of notecards, and making this collection available for exploration and search during design sessions. We began by extending the electronic whiteboard and sketching tool Calico [7] that provides developers with distinct advantages over traditional whiteboards, the most important of which is the ability to directly manipulate sketches that are made on the whiteboard by selecting and moving them around. This allows for anything drawn or written on the whiteboard to become elements of design knowledge that can be built upon. As a result, we envision DesignMinders being used alongside Calico by developers during design sessions. To better understand how DesignMinders accomplishes this, imagine the following scenario.

A team of developers gathers to discuss the design of a new clinical application for viewing and managing patient account information. Before they begin sketching concepts for the user interface, they turn to the DesignMinders noteboard running alongside the electronic whiteboard and begin exploring the available notecards. They search for all cards tagged with the "ui" keyword, and are presented with a list of notecards. They quickly browse through the names of the notecards, reading the descriptions and glancing at some of the associated diagrams, and quickly assess if a notecard is relevant to their application. They drag-and-drop several relevant notecards from the search list to the noteboard and create a stack.

One of the selected notecards is named "Multiple Monitors" and describes how a user interface can be designed to take advantage of multiple displays by allowing certain elements to be expanded outside of the primary window. Similar to a design pattern, the notecard's name provides a brief description of the situation in which the knowledge can be applied. Brief, descriptive names allow for notecards to be easily identified when browsing a large collection. The body of the notecard contains the details of the design knowledge that is being captured, like a concise description of the commonly encountered situation and a suggestion for a possible solution. Realizing that the approach suggested by the notecard is relevant to their application since the primary window could potentially be cluttered with other information, they incorporate it into their discussion.

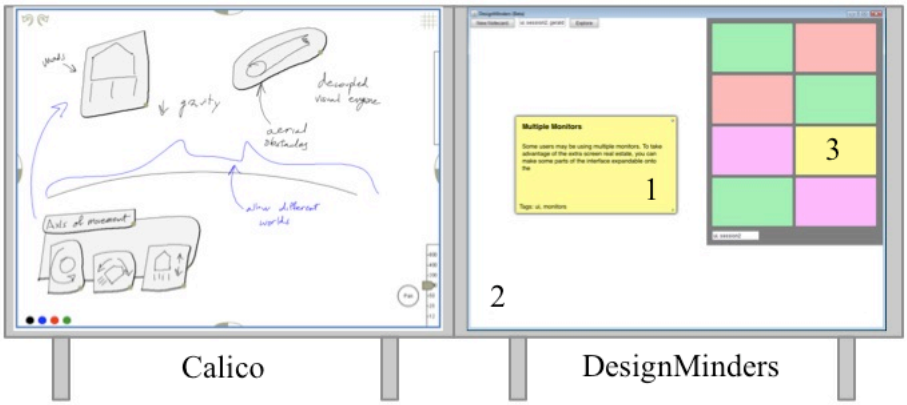

Fig. 1 DesignMinders running alongside Calico on a pair of electronic whiteboards showing (1) a notecard, (2) the noteboard, and (3) the search list.

The developers then proceed to use Calico to draw sketches of what the user interface will look like, numerous class diagrams describing the model that will be used to represent the account information, and a list of tasks that will need to be done to complete this part of the project. They decide on a user interface that provides search functionality that populates a list of accounts. As they work through the design, DesignMinders is running on a secondary display alongside Calico and is monitoring their canvas actions for keywords that can be used to bring up notecards that are relevant to the design problem. It notices that the word "search" was written and automatically brings up any notecards that are tagged with the keyword. As the notecards are displayed on the noteboard, the title "Large Search Results" catches the attention of one of the developers. He selects the notecard to view more details and sees a description of a problem commonly encountered when displaying search results. The details of the notecard describes an approach to breaking the results into smaller sets that can be retrieved one at a time, a technique which the creator of the notecard had employed in a previous project. The reverse side of the notecard contains a class diagram created in a previous design session that roughly describes the implementation. Realizing that this problem most certainly does apply to their application, the developer brings up the notecard and the team re-evaluates their existing design to incorporate the pattern and address the issue.

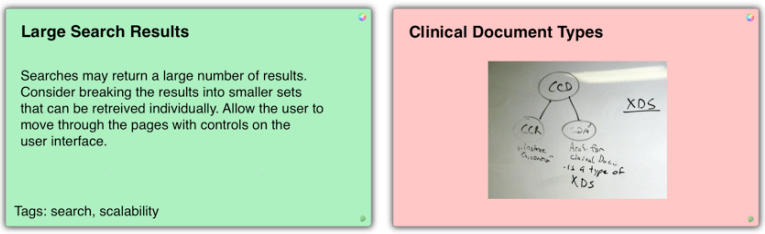

Fig. 2 Notecards showing name, details, tags, and reverse side with image captured from whiteboard.

As the session progresses, the developers come up with numerous conceptual sketches of the user interface, brief notes on how patient account information will be retrieved, and some network diagrams describing the eventual deployment of the application. Realizing that these sketches and notes could be useful for future design sessions, they lasso several areas of the Calico canvas and select the "Create Notecard" function, which result in several new notecards in the DesignMinders interface. The developers decide that the user interface notecards should be colored green, the patient account notes yellow, and the deployment diagrams red. The notecards are automatically tagged with the day's date and the internal name of the project. With the cards now on the noteboard, they quickly go through each one and give it a name and provide additional details or tags. The following week, the developers return to the conference room to further work out details of the design. They use the previously created notecards as a starting point to continue their discussion.

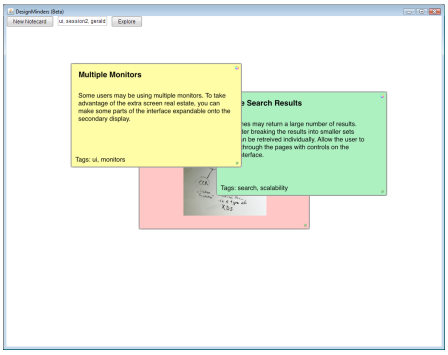

Fig. 3 Noteboard with several notecards.

4. Discussion

DesignMinders faces the knowledge reuse challenges that exist during informal software design by presenting developers with a lightweight means of capturing,

representing, and applying design knowledge. This is partially accomplished by representing bits of knowledge using notecards that mimic physical index cards. Physical cards lend themselves to easy browsing, manipulating, annotation, and ad hoc organization through stacks. Our goal was to reproduce this interaction in an electronic form to take advantage of the indexing and searching that can be performed. The ease with which a notecard can be created "in the moment" and the fact that none of the elements of a notecard (name, details, and tags) are required significantly lowers the barrier to capturing important knowledge during design sessions. The conciseness of the notecards also lends themselves to quick browsing to determine relevancy to a particular design. Our goal of a lightweight knowledge retrieval tool is also accomplished through the use of a search interface that allows for quick browsing of collected notecards and a reduction of the search space through the use of tags and colors to indicate significance or categories. The noteboard also allows for the ad hoc creation of localized groups by stacking relevant cards, much like one would create a stack of index cards.

5. Challenges

We see much promise in DesignMinders as a design knowledge reuse tool that can aid in collaboration during software design sessions and answer the "what if?" questions. Our lightweight and informal approach to knowledge reuse presents several challenges for future work.

The first challenge is to continue to lower the barrier to quickly and easily capturing knowledge by improving the integration between the ongoing design and the knowledge representation. This includes the current ability to manually select elements of the Calico canvas and create new notecards on the noteboard, as well as to have notecards be created automatically based on groupings or certain criteria such as the amount of time spent on a specific canvas. Also, we would like to increase the amount of information that is automatically collected from Calico by taking advantage of contextual elements, like UML diagrams and lists, to automatically populate detail and tag fields on new notecards.

The second challenge is to allow developers to easily annotate the design with reusable knowledge. Notecards are currently retrieved by browsing a list that allows for filtering by any of the fields on the notecard as well as the color. Relevant cards can be dragged onto the noteboard to create stacks of cards. We are exploring the ability to drag-and-drop cards from the DesignMinders interface directly into the Calico canvas, like attaching a sticky-note to a whiteboard. This would allow for the notecards to essentially become part of the design, and even become rationale for certain design decisions that are made.

The third challenge is to improve the relevance of the knowledge that is presented to the developer during a design session. This includes adding the ability to recognize words and phrases written on the canvas and present the designer with relevant cards based on keywords. This further reduces the amount of work that the developer must do to locate prior knowledge. There is also the potential to find relevant cards based on similar diagrams. For example, if a UML diagram is drawn on the Calico canvas

while in UML mode, notecards with similar diagrams could be retrieved and displayed alongside, or even directly incorporated into the canvas as reusable palette elements.

6. Conclusion

A prototype version of DesignMinders is currently being developed. While the basic functionality that we have described provides a novel approach for capturing and presenting knowledge during informal software design sessions, DesignMinders can be improved by addressing the challenges presented. We began by posing several scenarios in which developers were hindered in their ability to capture and recall previously gained insights during a software design session. We see DesignMinders as a way to remove these barriers to knowledge reuse while addressing the needs of the developers in capturing and presenting knowledge during this distinct activity.

Acknowledgments. Effort partially funded by the National Science Foundation under grant number 0920777.

References

[1] G. Fischer, "Seeding, Evolutionary Growth and Reseeding: Constructing, Capturing and Evolving Knowledge in Domain-Oriented Design Environments," *Automated Software Engineering*, vol. 5, pp. 447-464, 1998.

[2] M. Cherubini, G. Venolia, R. DeLine, and A. J. Ko, "Let's go to the whiteboard: how and why software developers use drawings," *Proceedings of the SIGCHI conference on Human factors in computing systems,* 2007.

[3] A. Murray and T. C. Lethbridge, "On generating cognitive patterns of software comprehension," *Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative research,* 2005.

[4] I. Frank M. Shipman and C. C. Marshall, "Formality Considered Harmful: Experiences, Emerging Themes, and Directions on the Use of Formal Representations in Interactive Systems," *Computer Supported Cooperative Work,* vol. 8, pp. 333-352, 1999.

[5] R. Farenhorst, "Tailoring knowledge sharing to the architecting process," *SIGSOFT Software Engineering Notes,* vol. 31, p. 3, 2006.

[6] A. Aurum and M. Handzic, *Managing Software Engineering Knowledge*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

[7] N. Mangano, A. Baker, and A. v. d. Hoek, "Calico: a prototype sketching tool for modeling in early design," *Proceedings of the 2008 international workshop on Models in software engineering,* 2008.