Automated capture and retrieval of architectural rationale

H. Richter, P. Schuchhard, and G.D. Abowd
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
{pascal,hrichter,abowd}@cc.gatech.edu

Abstract

The Software Architecture Analysis Method (SAAM) was developed at the Software Engineering Institute in the mid-90's and has been effective in helping industrial projects to uncover a shared understanding of the high-level organization of large software systems as well as to reveal how that structure is impacted by suggested changes to system requirements. In the MORALE project at Georgia Tech, we are further investigating the potential for a process such as SAAM to capture the architectural rationale of an evolving software project. One of the features of SAAM is that it is a people-oriented process. Yet this creates difficulties in keeping track of the very rich discussions on system organization and change. In this paper, we present the use of ubiquitous computing technology to augment the conventional SAAM process through automated capture, integration and access to the architectural discussions and artifacts produced by a SAAM session.

Keywords

architectural analysis, SAAM, ubiquitous computing, automated capture

1 INTRODUCTION

Software evolution is a costly and time consuming software development activity. Effective system evolution requires understanding both the way that an existing system accomplishes its tasks and the mission-oriented rationale for any changes that feed its evolution. The MORALE project at Georgia Tech addresses the problem of designing and evolving complex software systems. MORALE addresses these problems by integrating several different technologies: reverse engineering to extract and visualize architectural information (Jerding 1995); requirements analysis to structure and understand an organization's changing mission-oriented goals (Potts 1994); and finally software architecture impact analysis to determine how requirements changes affect the high-level organization of a software system.

This paper focuses on a particular process in support of the latter technique. Specifically, we make use of the Software Architecture Analysis Method (SAAM) developed at the Software Engineering Institute in the mid-90's (Bass 1998, Kazman 1996 and Kazman 1994). SAAM is a scenario-based method that can help extract how changing requirements will impact an already existing software system. In this context, we can understand the entire SAAM process as a design rationale technique that is centered on software architecture.

The SAAM process, summarized in Section 3 below, is a very people-oriented technique, and as such, has advantages and disadvantages. We are concerned with the disadvantage that SAAM requires additional time and effort to produce a coherent written summarization of the results of the analysis. In this paper we investigate how ubiquitous computing technology might address this disadvantage.

Overview of Paper

In Section 2 we provide background on design rationale and automated capture environments. In section 3, we present an initial prototype built to investigate how automated capture tools can support an architectural rationale technique such as SAAM. The prototype, called SAAMPad, is a meeting room environment centered around a large electronic whiteboard. We will demonstrate how this prototype environment has been shaped to support naturally defining a software architecture and capturing scenario-based impact analyses during a SAAM session. Finally, we conclude with our future plans for rationale capture of SAAM.

2 BACKGROUND AND RELATED WORK

We have introduced the SAAM process as a design rationale technique centered on architecture. In the following section we discuss design rationale in general as well as specific techniques that help to capture or structure the rationale behind some artifact. Next, we look at the more general problem of capture of live experiences as a paradigm for multimedia authoring and retrieval.

2.1 Design Rationale

Design rationale is the explanation behind the design - why the design is the way it is. Rationale can include assumptions made about the system, the alternatives considered, and the reasoning behind decisions. Often this rationale is contained only in the minds of the developers of the system. Yet this undocumented information could help system evolution by exposing previous decision points, alternatives, and assumptions that are vital to efficiently changing the software.

Rationale capture, however, can require additional effort to document and organize information alongside the design.

Traditional approaches to rationale capture have involved two methods for documenting and structuring design rationale: process-oriented and structure-oriented (Rittel 1973, Shum 1994). A process-oriented approach focuses on documenting rationale as it occurs during design meetings, yet can disrupt, and therefore alter, the actual design activity that it is trying to support. A structure-oriented approach focuses instead on a post hoc structuring of the rationale to show the complete design argument but risks losing critical information by waiting until after the fact to record rationale. A nice balance could be achieved if there was a way to capture the rationale as it occurs, but without introducing interruptions to the design process.

Conklin et al. (1991) attempted to allow less disruption to the design process with a graphical tool, gIBIS, to record the rationale. The WinWin project (Boehm 1994 and Bose 1995) is looking specifically at recording architectural rationale. While both gIBIS and WinWin attempt to reduce the overhead in capturing rationale, they focus on particular elements that must still be formally documented during the discussions. In this next subsection, we discuss the general problem of capture that aims to provide a ubiquitous yet unobtrusive service.

2.2 Capture

A general challenge in ubiquitous computing is to provide automated tools to support the capture, integration and access of this multimedia record (Abowd 1996b). The purpose of this automated support is to have computers do what they do best, record an event, in order to free humans to do what they do best, attend to, synthesize, and understand what is happening around them, all with full confidence that the specific details will be available for later perusal.

At Georgia Tech, we have a lot of experience in building environments that capture live experiences in order to make them available for later review and summarization. Most of our experience in this area has been applied to the education domain, in a project entitled Classroom 2000 (Abowd 1998, Abowd 1998b, and Abowd 1996). In Classroom 2000 the many streams of activity in a typical lecture ---what is spoken, what is seen, what is written down on a whiteboard and what is shown on public displays--- are combined to provide a rich interactive experience that is becoming increasingly more difficult to capture using traditional pen and paper notes.

The challenge in applying ubiquitous technology to the SAAM process is to provide an organization of multimedia streams that facilitates access to the architectural rationale. This requires going beyond simple record and playback schemes to providing meaning to the various activities and records of a SAAM session. Since SAAM is a structured process, as explained in the next section, our task is made simpler.

3 ENHANCING SAAM WITH SAAMPAD

The Software Architectural Analysis Method, or SAAM, is a structured method for understanding the high-level organization of a software system and determining the impact of requirements changes on that structure. For a detailed description of SAAM see Bass (1998), Kazman (1996) and Kazman (1994). During a SAAM session a great deal of architectural rationale can be discussed. Constraints and assumptions may be raised while understanding the original architecture. Evaluators may consider a number of solutions and make tradeoffs. Yet users trying to take detailed notes of all of this information are not likely to fully participate in the discussions. Additionally, it is hard to know exactly how important certain items are during discussion so that they may be documented more fully. Instead, they only become important at the time they are actually needed. As such, we propose a new way of automatically capturing the entire experience of a SAAM session and allowing easy access to the information later.

A typical SAAM session is a live event involving discussions by 3-10 designers, managers and facilitators that is centered around drawings of the architecture on a public display. By converting the public display to an electronic whiteboard surface and recording the discussion with digital streaming media technology, we can automatically capture the SAAM sessions and then provide the ability to salvage summary information afterwards. By allowing the ubiquitous computing infrastructure to do what it does best, record and capture activity, we can free the humans to do what they do best, synthesize and understand activity in a technical discussion.

Our biggest challenge in providing automated capture support is to enhance the SAAM process *without* changing the way the method currently works. The capture tool we propose must (1) allow the users to concentrate on the SAAM process and not on the capture, (2) provide summary information of the SAAM activities, and (3) provide users with meaningful ways to access all of the captured information.

The approach we used to achieve these goals was to start with a general capture tool, the central tool in Classroom 2000, and extend it to support SAAM specific activities. To demonstrate SAAMPad, we performed a SAAM evaluation of the Key Word In Context System (KWIC) described by Parnas (1972) for the proposed changes listed in chapter 9.3 of Bass et al. (1998). Examples from this case study are used throughout the rest of the paper. Next we describe how users will interact with SAAMPad during the different stages of the SAAM process.

4.1 Using SAAMPad

Participants in the SAAM session gather around an electronic whiteboard in a room capable of recording audio or video. Figure 1 is a picture taken of someone using SAAMPad in such an environment. The participants then proceed to follow the steps outlined in the SAAM method. SAAMPad adjusts its behavior to support

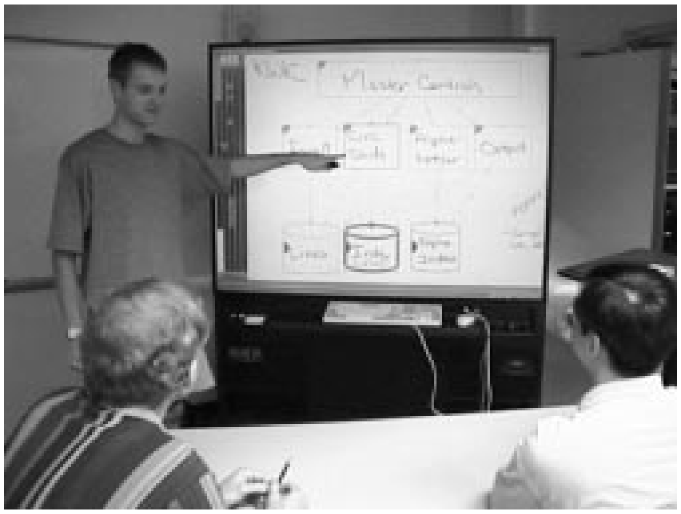

Figure 1. The SAAMPad environment.

each particular SAAM step. SAAMPad's functionality is described for each of the SAAM stages below.

1. Describe the existing architecture. The team of evaluators draws the architecture onto the electronic whiteboard, much as they would draw upon a traditional whiteboard. We created a simple gesture-based interface that would allow for a clean box-and-line drawing. An architecture for the KWIC system is shown in Figure 2. What is important to note here is that all of the activity in generation is captured. Discussion is recorded and the times associated with various actions on the whiteboard surface (all activity with the pen) is timestamped to allow us to later index into the discussion.

2. Develop scenarios. The team must develop task scenarios that illustrate the kinds of activities the evolved system must support and the kinds of anticipated change to the system. SAAMPad provides a simple interface to record brief names associated with scenarios as well as explanatory text. Though this part of the process is not explicitly captured, it would be a simple and potentially useful task to record the scenario brainstorming session.

3. Perform scenario evaluations. For each scenario, the analysis team describes how the architecture would execute the scenario, or what modifications would be necessary to meet it. Additional gestures are added to the SAAMPad interface from phase 1 to allow adding, deleting and marking components and connectors. Color is used to denote the changed status of the components. Designers can also use the pen to highlight the component that is the focus of attention.

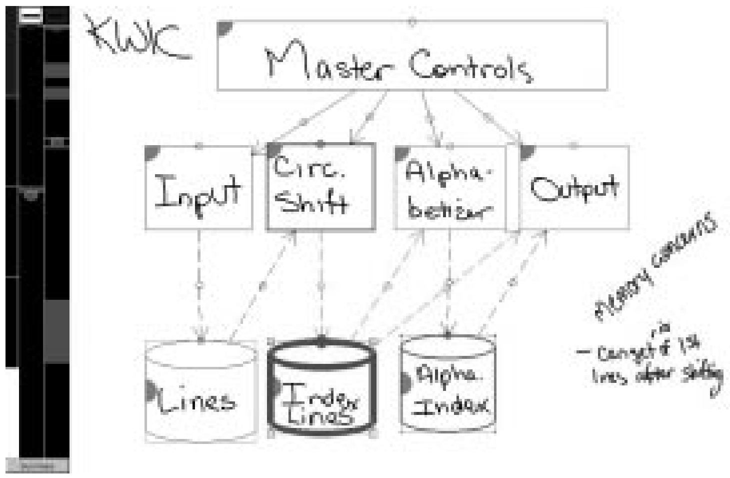

Figure 2. Visualization of a scenario evaluation of the KWIC system. The timeline on the far left represents the entire SAAM session, while the other timelines show the activities for two specific components.

4. Summarize the information. The purpose of the previous steps, from SAAMPad's perspective, is to capture relevant timestamps associated to important SAAM activities. The timestamped activity is associated to parts of the architecture, meaning that this information can be used to provide a number of automatically generated summaries and visualizations of the architecture that will help designers understand the overall impact of the SAAM session. Designers can view these visualizations for a quick overview of the impact of the SAAM session, and also use them to index into the other multimedia streams (e.g., the recorded audio and video of the session) to find more details. Figure 2 shows an example visualization of a KWIC scenario evaluation. This visualization is further explained in the following section.

4.2 Visualization

In order to facilitate the retrieval of architectural rationale, we need to organize the captured information around meaningful structures of the SAAM process. Architectural diagrams act as the center of discussion, provide a visual representation of the system, and indicate the effects that scenarios have. As such, these diagrams serve as the central focus of the capture and access of the rationale. We chose to use gesture recognition to detect architectural elements, enabling SAAMPad to recognize events surrounding the architecture without disrupting the users' discussion. We then organized the visualization around the architecture and these events. Additionally, the SAAM process is composed of different phases, namely architecture generation, scenario generation, and scenario evaluation. SAAMPad supports these different phases with tailored capture and visualizations.

When trying to visualize an entire recorded SAAM evaluation, we want to show all of these events in a way that would support effective browsing by the designers. The solution we chose was to represent the entire SAAM evaluation along a timeline that explicitly delimited the generation phase and the individual scenario evaluation sessions. In Figure 2, the timeline for the whole KWIC SAAM evaluation is shown on the extreme left of the screen as a simple image broken into a number of sections. The top section indicates the generation phase of the SAAM evaluation. Tapping on that portion of the screen reveals a timeline area for the generation phase in the adjacent portion of the screen as well as the architectural diagram in the main portion. This phase-specific timeline can show information for any of the architectural elements shown. For example, tapping on the component labeled "Input" in Figure 2 would reveal a timeline of events in the generation phase that are related to that component, indicating when in the phase that component was created and when it was the focus of attention of the discussion. The phase-specific timeline is interactive. Tapping it at various places will launch an audio or video player at that point in the recorded session

Besides this timeline visualization, we are also exploring ways of visualizing an entire SAAM session, instead of one phase or evaluation at a time. We have implemented a version of a 'fisheye' diagram, showing the components and connectors in the initial diagram with the line width representing the frequency that they change over all of the scenario evaluations. This gives a quick indication of which are most affected by the scenarios. Another view we have implemented is a simple textual table of the architectural components and the scenarios in which they are created or require change. These visualizations provide compact summaries of the overall results of the SAAM evaluation.

5 CONCLUSIONS AND FUTURE WORK

SAAM is a people-oriented architectural analysis method for extracting how changing requirements impact an existing software system. SAAM discussions can be rich in reasoning about the architecture and the impact of changes. Yet trying to capture this information places a burden on those performing a SAAM evaluation. We introduced SAAMPad, a tool for the automated capture and retrieval of architectural rationale surrounding SAAM. SAAMPad utilizes ubiquitous computing technology extended with architectural semantics to achieve this goal while preserving the SAAM process. One potential drawback of this tool is that it concentrates only on supporting the SAAM process and might be less suited when developers want to try a new way of evaluating systems. However, by concentrating on a single method, we were able to provide more meaningful ways of capturing and accessing the rationale.

SAAMPad has so far only been evaluated on a simple case study of the KWIC system. We plan to perform additional case studies on larger, real-world systems. Additionally, we plan to investigate more visualization techniques to improve the access to the architectural rationale and the automated summaries that are created. We hope that the additional case studies will help us identify specific areas of improvement. We would also like to take advantage of recognition technologies so that the system can better understand the content of information being captured and facilitate more useful retrieval.

6 ACKNOWLEDGEMENTS

This effort is sponsored by the Defense Advanced Research Projects Agency, and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-96-2-0229. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, Air Force Research Laboratory, or the U.S. Government.

7 REFERENCES

Abowd, G.D., Brotherton, J., and Bhalodia, J. (1998) Automated Capture, Integration, and Visualization of Multiple Media Streams, in *Proceedings of the 1998 conference on IEEE Multimedia and Computing Systems*.

Abowd, G.D., Atkeson, C.G., Brotherton, J., Enqvist, T., Gulley, P., and LeMon, J. (1998b) Investigating the capture, integration, and access problem of ubiquitous computing in an educational setting, in *Proceedings of the 1998 conference on Human Factors in Computing Systems (CHI'98)*, 440-7.

Abowd, G.D., Atkeson, C.G., Feinstein, A., Hmelo, C., Kooper, R. Long, S., Sawhney, N. and Tan, M. (1996) Teaching and Learning as Multimedia Authoring: The Classroom 2000 project, in *Proceedings of the ACM Conference on Multimedia (Multimedia'96)*, 187-98.

Abowd, G.D. (1996b) Ubiquitous Computing: Research Themes and Open Issues from an Applications Perspective. Technical Report GVU96-24, Georgia Institute of Technology.

Bass, L., Clements, P., and Kazman, R. (1998) Software Architecture in Practice, Addison-Wesley.

Boehm, P., Bose, P., Horowitz, E. and Lee, M.J. (1994) Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach, in *Proceedings of the International Conference on Software Engineering (ICSE 17)*.

Bose, P. (1995) A Model for Decision Maintenance in the WinWin Collaboration Framework. *Knowledge Based Software Engineering (KBSE'95)*.

Conklin, J.E. and Yakemovic, K.C.B. (1991) A Process-Oriented Approach to Design Rationale. *Human-Computer Interaction*, **6**(3&4), 357-91.

Degen, L., Mander, R. and Salomon, G. (1992) Working with Audio: Integrating Personal Tape Recorders and Desktop Computers, in *Proceedings of 1992 conference on Human Factors in Computing Systems (CHI'92)*, 413-18.

Garlan, D., Monroe, R.T. and Wile, D. (1997) ACME: An Architecture Description Interchange Language, in *Proceedings of CASCON'97*, 169-83.

Jerding, D. and Rugaber, S. (1997) Using Visualization for Architectural Localization and Extraction, in *Proceedings of the Fourth Working Conference on Reverse Engineering*, October.

Kazman, R., Abowd, G., Bass, L., and Clements, P. (1996) Scenario-Based Analysis of Software Architecture. *IEEE Software*, **13**(6), 47-56.

Kazman, P., Bass, L., Abowd, G. and Webb, S.M. (1994) SAAM: A Method for Analyzing the Properties of Software Architectures, in *Proceedings of the International Conference on Software Engineering (ICSE 16)*, 81-90.

Minneman, S. Harrison, S. Janseen, B., Kurtenbach, G., Moran, T., Smith, I. And van Melle, B. (1995) A Confederation of Tools for Capturing and Accessing Collaborative Activity, in *Proceedings of the ACM Conference on Multimedia (Multimedia'95)*.

Moran, T.P., Palen, L., Harrison, S., Chiu, P., Kimber, D., Minneman, S., van Melle, W., Zelweger, P. (1997) Salvaging Multimedia Meeting Records, in *Proceedings of the 1997 conference on Human Factors in Computing Systems (CHI'97)*, 202-9.

Parnas, D.L. (1972) On the Criteria to be Used in Decomposing Systems into Modules. *Communications of the ACM*, **15**(12), 1053-58,

Potts, C., Takahashi, K., and Anton, A.I. (1994) Inquiry-Based Requirements Analysis. *IEEE Software*, **11**(2), 21-32.

Rittel, H. and Webber, M. (1973) Dilemmas in a general theory of planning. *Policy Science*, **4**, 155-69.

Shum, B.S. and Hammond, N. (1994) Argumentation-Based Design Rationale: What Use at What Cost? *International Journal of Human-Computer Studies 40*, **4**, 603-52.

Stifelman, L.J. (1996) Augmenting real-world objects: A paper-based audio notebook, in *Proceedings of 1992 conference on Human Factors in Computing Systems (CHI'92)*, 199-200.

Weber, K. and Poon, A. (1994) Marquee: A tool for real-time video logging, in *Proceedings of 1994 conference on Human Factors in Computing Systems (CHI'94)*, 58-64.

Whittaker, W., Hyland, P. and Wiley, M. (1994) Filochat: Handwritten notes provide access to recorded conversations, in *Proceedings of 1994 conference on Human Factors in Computing Systems (CHI'94)*, 271-77.