In Proceedings of the First Int'l Workshop on Multimedia Requirements Engineering - Beyond Mere Descriptions, (MeRE'06),
14th International Requirements Engineering Conference (RE'06),
September 11-15, 2006 Minneapolis/St. Paul, Minnesota, USA.

Incorporating Multimedia Source Materials into a Traceability Framework

Heather Richter, Robin Gandhi, Lei Liu and Seok-Won Lee
Department of Software and Information Systems
University of North Carolina at Charlotte
{richter, rgandhi, lliu10, seoklee}@uncc.edu

Abstract

Requirements engineers generate domain models and requirements specifications from a variety of rich, informal sources. Yet much of this informal information is not preserved to maintain the traceability of requirements back to their origins. In this paper we describe TRECRE, a traceability framework for preserving and providing access to a variety of multimedia source materials.

1. Introduction

Software development is a complicated, information-rich process: requirements engineers seek domain and system information and transform that knowledge into requirements models and specifications to use throughout development. The process of creating these artifacts is naturally a lossy one as rich and informal information is abstracted and formalized. This is in most cases beneficial, as only a relevant and structured understanding remains to be moved further through development. However, as requirements evolve and change, or as problems are discovered, the formal specifications may no longer be sufficient to understand the impact and necessary changes. Additional details about the underlying knowledge and decisions that led to the specification may be useful to facilitate requirements evolution and further development.

Traceability is the ability to follow the evolution of an object – a piece of knowledge, a requirement, a design idea – from its inception throughout software development [3]. We are particularly concerned with pre-traceability of requirements: tracing a requirements object back to the original knowledge used in its creation. Requirements knowledge may have many sources such as customer discussions, expert interviews, sketches and diagrams, and documents and other specifications. All of this original, often informal, requirements knowledge is difficult to formally document and maintain due to its volume and complexity.

Our aim is to support requirements knowledge traceability by automatically capturing and maintaining source materials in as close to original form as possible. This means recording the discussions, and saving the sketches, images and other artifacts. We will then provide means for efficiently searching and browsing all of these materials, and as automatically as possible, link media to related domain models and requirements specifications. Thus, we are proposing a multimedia traceability framework to address the capture and review of requirements source materials, aiming to complement, not modify, existing requirements methods and artifacts.

In this position paper, we present our conceptual traceability framework — TRaceability-preserving Evidence Capture for REquirements domain modeling (TRECRE) — for maintaining the rich, informal information contained within a variety of multimedia source materials. We first present our motivating experiences in recording informal discussions and the potential use of those discussions in software development. These experiences motivated the formation of a more general framework to address the issues of maintaining and navigating a number of informal information sources. We present our initial framework and our plans for implementing this framework as part of an existing requirements domain modeling toolset. Finally, we conclude with the challenges and issues we expect to address in implementing and evaluating this framework.

2. Meeting Capture Background

A common theme in ubiquitous computing and multimedia research is the automated capture of everyday activities for later access and use. A number of prototypes have been built and deployed in both the classroom and meeting room. Many systems attempt to augment slides, notes, or whiteboard activity with

audio and/or video. For example, the TeamSpace system records interactions with presentations, agendas and action items and integrates them with the meeting audio and video [11]. Meeting participants can then use the slides, meeting notes, or agenda items to replay portions of the meeting. A number of similar systems have investigated various methods for recording and indexing such meeting content.

While recording generic meetings may be beneficial, we have also been investigating more specific discussions in software development, where recording has long term implications for addressing important problems such as rationale and traceability. In previous work, we have implemented two interface prototypes to evaluate the use of captured discussions in software development tasks.

We first investigated the capture of Software Architectural Analysis Method (SAAM) sessions [10], a method for analyzing the effects of proposed changes on the software architecture. As part of that effort, we video taped a real set of SAAM discussions and manually created an interface to review that video. Users navigated the video by the locations of keywords — the architectural elements — that were contained and linked in a text document summarizing the session. In an evaluation, we observed users answering questions about the software architecture and the SAAM analysis, using both the document and video. Users stated that the video was beneficial not only in providing additional details that were left out of the document, but also in providing information at different levels of abstraction, and in adding additional authority to the statements in the document. In other words, the video recording of a discussion contained useful information even when a formal document summarizing that discussion existed.

We then investigated the recording of knowledge acquisition sessions in the TAGGER project [9]. We video taped a knowledge acquisition session, produced an automated transcript and annotated that transcript with keywords. We observed how people used the video and transcript to create a requirements document based on the knowledge acquisition session that they watched previously. Users looked up specific details in the video they may otherwise have left out of their documents. They clarified important points and had fewer inaccuracies as a result. And, they used the video to look for issues they had forgotten about or not taken notes on. An important result of our evaluation was that the video was only utilized when users had helpful navigation mechanisms, such as the annotated transcript, to make finding specific conversation points relatively easy. Thus, merely recording such conversations is not sufficient. Users need indices, summaries, or other navigation mechanisms in order to find and utilize information within the recordings.

Our experiences in meeting capture and access show that recording informal information such as conversations can be beneficial both in creating new artifacts and in reviewing existing ones. Our prototypes demonstrated that recording and indexing such conversations is feasible, at least on a smaller scale. Automated technologies provide many ways to record conversations and artifacts with little or even any additional burden on software developers. However, any single discussion is only going to have limited use. Thus, to further study the impact of recording meetings and other informal information we need to more fully incorporate those materials into the overall software design process.

3. Traceability background

Based on interviews with requirements engineers, Gotel and Finkelstein concluded that many problems in requirements traceability were with inadequate pre-traceability – an inability to record and trace information related to requirements production and revision [3]. Furthermore, Gotel and Finkelstein discovered that for many, the crux of the problem was the inability to locate and access the source of a requirement.

A number of tools and techniques address issues of requirements and design post-traceability – tracing requirements elements as they are evolved and used (e.g. DOORS [14], RDD-100 [4].) One common solution is to allow any object within a modeling environment to be related to another, such as relating a design object to the requirements it addresses. Other tools allow explanations or descriptions of rationale to be attached to objects to help explain their creation [8]. These techniques certainly provide valuable traceability, and allow software engineers to follow the relationships between elements of supported models. Each of these techniques, however, requires that the desired piece of information already exists within the modeling or specification environment.

We are concerned with more informal information, such as a conversation or sketch, that is not currently represented within existing toolkits or environments. Our experiences in meeting capture suggest that this information could be beneficial throughout requirements and development. However, representing such information in yet another formal model or specification is an added burden and likely insufficient. Instead, our approach relies on recording and maintaining the original materials in as close to original form as possible, and only requiring sufficient

additional structures to help index the content and to link portions of that content to related formal elements within existing environments. These structures will then aid software engineers in reasonably navigating and finding information within those source materials. The following scenario further illustrates our goals:

Phil is working on the requirements specification of a system to electronically display checklists for pilots in the cockpit. The domain object model contains two subtypes of checklist and Phil is unsure if those types need to be displayed differently. He uses the model to query the collection of source materials for additional information on the two types of checklists. Phil reads a document from an airline and replays portions of an interview with a pilot to better understand the subtle differences between these two checklists. As he creates a new requirement, he links it to the domain object model elements. Based on his recent activity, the interview and document he just viewed are also automatically linked to that requirement.

4. The TRECRE Framework

We are developing a framework, called TRECRE, for supporting traceability by preserving informal knowledge within software requirements. The framework is outlined in Figure 1. Our framework is based upon two intermingled activities of software requirements. Requirements engineering is an ongoing, informal process where people talk to customers, users and other experts, read documentation on the domain and on development, sketch and brainstorm ideas, and work and make decisions together to make progress. This process is represented on the bottom of our framework in Figure 1, where we plan to record and preserve much of this source material.

Requirements engineers also create formal models and specifications of the domain and requirements within a modeling or development environment. As work progresses, those models evolve and new ones are created and linked to existing elements. This is the top layer of our framework in Figure 1. Traceability has mostly been supported within this layer, the more formal aspects of this process.

Informal activities can already be easily preserved if desired. Discussions can be video or audio taped. Informal documents can be saved. Yet, this is not currently common practice because it does not actually maintain traceability. We need structures that bridge the gap between the informal "universe of discourse" and the modeling and specification domain. The middle layers of Figure 1 represent this bridge, where formal elements are automatically linked in a variety of

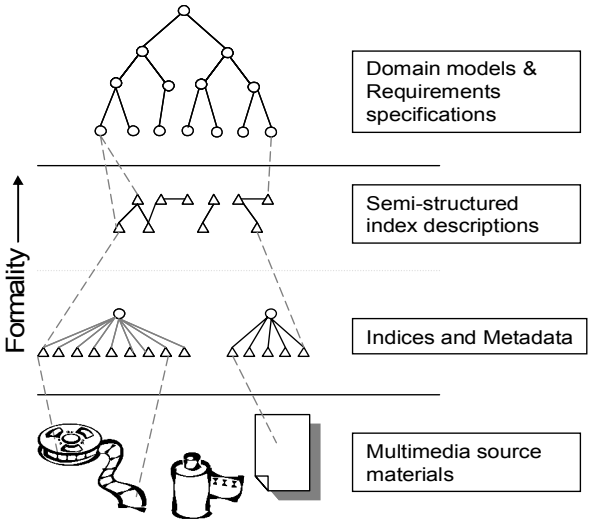

Figure 1. The conceptual TRECRE framework.

ways to related source materials, and vice versa, to maintain traceability. Software engineers thus navigate up and down the layers to find and review the information they desire. Our goal is to explore a variety of techniques that can create these links automatically, based on both processing the media and on tracking the use of the media. Thus, we are not changing the current methods or process of modeling and specifying requirements, but instead attempting to preserve that process and provide access to that record when needed.

4.1. Indexing Source Materials

In our framework, source materials can be a variety of formats, from multimedia recordings of meetings, to text documents and web pages. Reviewing a meeting will require very different capabilities than reading a document. However, the same structural notions can apply to all of these media. First, any particular file may have metadata associated with it. This data describes properties of the media item, such as time, date, description and others. For example, a meeting will likely have a date, start time and end time, list of participants, and a description. This kind of metadata is what many multimedia information retrieval systems use to search for files or objects in a multimedia database. This metadata is important information, yet accessing an entire discussion or document is often not the appropriate scope. A particular topic may only be discussed for 10 minutes during a meeting, or relate to one page of a document. Thus, we need ways to represent, structure, and link to smaller and more specific segments of the source materials.

The most basic building block of such a structure is an *Index*. An index is simply a meaningful location in a

media source. In audio or video recordings, the location is represented as a point in time in the recording. In documents, an index could be described in different granularities – perhaps a page, or section, or line of text. In both cases, the index also contains a meaningful descriptor that indicates what is located there. In our previous work with meeting recording prototypes, we explored a variety of indices such as agenda items, slides, architectural elements, and domain keywords. Thus, our framework will generically support the notion of sets of indices, and we will explore a variety of specific indices for various media. Additionally, indices may be related to each other and we will provide mechanisms to group and model sets of related indices to facilitate linking these structures to formal requirements and domain elements.

4.2. Semi-Structured Indices

The bridging middle layers of Figure 1 are critical to the usefulness of this framework. Given the range and complexity of the source materials in the informal world, and the diverse RE modeling methods in the more formal world, there will not be one deep structure providing this bridge. Instead, we will need to support a variety of broad, flexible, and shallow structures that can describe the source materials in ways that relate to the generic notions of goals [15], viewpoints [12], and scenarios [13] most often used for requirements and domain modeling. Such semi-structured sets of indices will help users navigate between the formal and informal worlds to gather, browse, and search for knowledge artifacts related to requirements.

4.3. Requirements Domain Models

To realize TRECRE, we are building upon the Onto-ActRE [6] requirements domain modeling framework. The Ontology-based Active Requirements Engineering (Onto-ActRE) [6] framework, through its theoretical foundations as a mixed-initiative approach, offers flexibility to gather and represent knowledge artifacts based on multiple RE methods and the synergy among them. The framework combines the strengths of multiple complementary RE modeling techniques in a unifying ontological knowledge engineering process. A uniform ontological frame representation promotes traceability among knowledge artifacts from multiple modeling philosophies with well-defined semantics for their structure and interoperability. More specifically, the Onto-ActRE framework includes models and methods for 1) Goal-driven scenario composition; 2) Requirements domain model; 3) Viewpoints hierarchy, and 4) Other domain specific taxonomies to hierarchically organize the application domain concepts, properties and their relationships.

To support the representation of rich knowledge structures, various ontological engineering processes are provided by the GENeric Object Model (GenOM) [7] toolkit. GenOM is an integrated development environment for ontological engineering processes with functionalities to create, browse, access, query, and visualize associated knowledge-bases. The conceptual architecture of GenOM is shown in Figure 2.

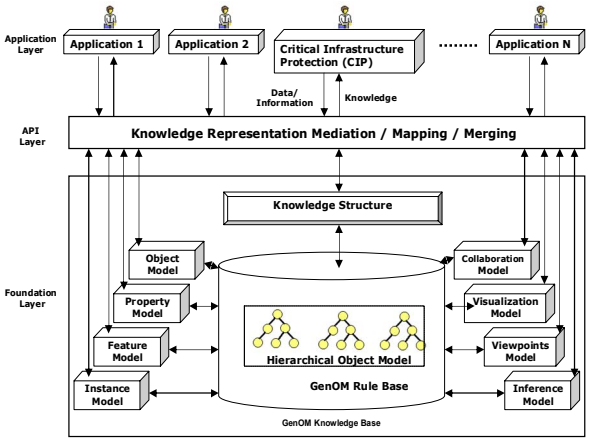

Figure 2: GenOM Conceptual Architecture

The GenOM meta-language consists of *Objects*, *Properties*, and *Features* with semantics that effectively support knowledge acquisition and representation. GenOM *Objects* with support for single or multiple inheritances are used to model hierarchical structures that describe the concepts in a domain. GenOM *Properties* are used to describe the characteristics or attributes of *Objects* and *Features*. Finally, GenOM *Features* are used to describe the relationship or dependencies that exist between *Objects*. Once the *Objects*, *Properties*, and *Features* are defined, they are instantiated to represent specific *Instances* that exist in a problem domain. GenOM is associated with an inference engine [1], which supports reasoning based on the *Objects*, *Properties*, and *Features* and *Instances* defined in its knowledge-bases.

The Onto-ActRE framework and GenOM together have been applied to build domain models from regulatory requirements documents for the Department of Defense Information Technology Security Certification and Accreditation (C&A) Process (DITSCAP) [2] automation with promising results in the initial stages [5].

In TRECRE, the sets of indices will also be represented within GenOM to facilitate linking

between the source material descriptions and the various domain models of Onto-ActRE, providing uniform representation of knowledge throughout the framework. Additionally, the use of GenOM provides a powerful mechanism to explore modeling and reasoning about the structure and relationships of indices themselves, and their relationship to domain model elements. This will enable us to create scalable navigation paths between a large number of potentially related elements and media.

5. Conclusion

In this position paper, we outlined our conceptual framework for preserving and providing meaningful access to informal requirements source materials. We are beginning to implement and evaluate aspects of this framework. Implementing the TRECRE framework involves a wide variety of technologies for recording, preserving, linking, and reviewing the multimedia source materials. For each of these stages we will need to investigate which technologies or methods best achieve our goals. We will investigate various technologies for recording and identifying sets of indices, such as using natural language analysis to determine conversation topics. We will examine the use of ontologies for discovering and analyzing the relationships between the media and model elements. We will expand upon our experience in building applications to review recorded meetings to create interfaces for navigating a large set of materials.

Finally, our goal in all of this work is to explore on a larger scale the usefulness of preserving informal source materials in software development. What materials are most important to maintain? How can users navigate a large corpus of source materials and easily find relevant information? And most important, what is the benefit of preserving all of these media on requirements creation and use?

6. References

[1] Carroll, J. J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K., "Jena: implementing the semantic web recommendations," In Proc. of the 13th Int'l World Wide Web Conf., USA, pp: 74-83, 2004.

[2] DoD Instruction 5200.40, "DITSCAP," 1997.

[3] Gotel, O. and A.C. Finkelstein, "An Analysis of the Requirements Traceability Problem," in the Proceedings of the Conference on Requirements Engineering, April 1994, pp 94-101.

[4] Holagant Corporation, The RDD-100 Product Family. http://www.holagent.com/products/product1.html.

[5] Lee, S. W., Gandhi, R. A., and Ahn, G., "Certification Process Artifacts Defined as Measurable Units for Software Assurance," To Appear in the International Journal on Software Process: Improvement and Practice, Wiley, July, 2006.

[6] Lee, S.W. and Gandhi, R. A., "Ontology-based Active Requirements Engineering Framework", Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC '05), Taipei, Taiwan, Dec. 15 - 17, 2005, pp. 481 – 490.

[7] Lee, S.W. and Yavagal, D., "GenOM User's Guide," Technical Report TR-SIS-NISE-04-01, Knowledge Intensive Software engineering Research Group, Dept. of Software and Information Systems, UNC Charlotte, 2004.

[8] Ramesh, B. "Supporting Systems Development by Capturing Deliberations During Requirements Engineering," IEEE Transactions on Software Engineering, 18(6), June 1992, 498-510.

[9] Richter, H., C. Miller, G.D. Abowd, and I. Hsi. "An Empirical Investigation of Capture and Access for Software Requirements Activities," in the Proceedings of Graphics Interface, 2005, pp 121-128.

[10] Richter, H., P.l Schuchhard, and G.D. Abowd. "Automated capture and retrieval of architectural rationale," Position paper, First IFIP Working Conference on Software Architecture, San Antonio, TX, February 1999.

[11] Richter, H., W. Geyer, L. Fuchs, S. Daijavad, and S. Poltrock. "Integrating Meeting Capture Within a Collaborative Team Environment," In the Proceedings of the International Conference on Ubiquitous Computing, Atlanta, GA, September 2001, pp123-138.

[12] Sommerville, I. and Sawyer, P., "Viewpoints: Principles, Problems and a Practical Approach to Requirements Engineering," Annals of Software Engineering, Vol. 3, 1997, pp. 101-130.

[13] Sutcliffe, A., "Scenario-based requirements engineering," In Proceedings of the 11th Requirements Engineering Conference, IEEE International, 8-12 Sept. 2003, pp. 320- 329.

[14] Telelogic DOORS Requirements Managements Traceability solutions. http://www.telelogic.com/corp/products/doors/index.cfm.

[15] van Lamsweerde, A., "Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice", In Proceedings of 12th IEEE Joint International Requirements Engineering Conference, Kyoto, 2004, pp. 4-8.