Theoretical Bases for Selecting Traceability Approaches to Support Software Development

Kannan Mohan
Baruch College

Nanda Kumar
Baruch College

Follow this and additional works at: http://aisel.aisnet.org/amcis2004

Recommended Citation
http://aisel.aisnet.org/amcis2004/195

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2004 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Theoretical Bases for Selecting Traceability Approaches to Support Software Development

Kannan Mohan  
Department of Computer Information Systems  
Baruch College  
kannan_mohan@baruch.cuny.edu

Nanda Kumar  
Department of Computer Information Systems  
Baruch College  
nanda_kumar@baruch.cuny.edu

ABSTRACT

Traceability is the ability of stakeholders to describe and follow the life of an artifact throughout the software development life cycle. Traceability is one of the most important quality attributes that are indicative of the maturity of software development organizations. While there has been extensive research focusing on the development of traceability models and support systems for different kinds of software development environments, there is a paucity of research that characterizes the tasks that need to be performed in implementing and using traceability. This research addresses a gap in this area by providing theoretical justifications for, and identifying key factors that dictate the choice of media based on the task characteristics. These prescriptions will help traceability tool developers in delivering traceability support that is tailored to specific tasks involved in different types of projects.

Keywords  
System development, traceability, media, media choice

INTRODUCTION

Requirements traceability is defined as the ability to describe and follow the life of a requirement through the software development life cycle (Gotel & Finkelstein, 1994). Traceability is considered to be critical for effective development and maintenance of software systems (Conklin, 1989; Domges & Pohl, 1998). Poor traceability practices can lead to low system quality, and will impact the maintainability of the system, thereby increasing the cost of changes. Despite the numerous research efforts focusing on developing effective traceability approaches, many challenges are yet to be addressed. Achieving traceability is often seen as an overhead by certain project stakeholders. Though prior research has attempted to emphasize the importance of tailoring traceability practices to project-specific environments (Domges & Pohl, 1998), project stakeholders are still affected by the lack of specific and concrete guidelines on how to make certain choices in establishing an appropriate traceability practice. Past research has highlighted that software developers do not currently have adequate guidelines on what knowledge to capture and how to capture and use the same in various scenarios (Domges & Pohl, 1998). For instance, often, developers tend to consult colleagues to identify impact of changes rather than relying on documentation. They do not have specific guidelines that suggest the use of particular types of communication media for different traceability tasks. Deciding on the appropriate level of traceability structure and selecting an appropriate medium for establishing communication among stakeholders who manage traceability are some of the challenges that have gained little attention.

This research addresses these issues by drawing from the literature on media selection. Prior research on media selection has focused on investigating factors that should be considered in the selection of appropriate media for communication. Trait theories and social interaction theories have been used in the past to explain the selection of appropriate media. Apart from characteristics of task, media, fit between task and media, social environment, and recipient availability as determinants of media selection (Straub & Karahanna, 1998), past research has considered numerous variables that determine media selection. Given the critical nature of traceability for system development success, and the importance of the criteria used to select appropriate media for communication tasks, how can project managers, analysts, designers, and developers select the appropriate media and structure for traceability? This research focuses on addressing this question by describing the various tasks involved in traceability and identifying the determinants for media selection for those tasks. We argue that these determinants will be useful in guiding project stakeholders in following a suitable traceability practice with reduced or justified levels of overheads.
BACKGROUND ON TRACEABILITY

Definitions and the Need for Traceability

Gotel and Finkelstein (1994) define requirements traceability as the ability to describe and follow the life of a requirement, in both a forward and backward direction, i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of the software development phases. Traceability has been considered as a quality attribute and many standards emphasize the establishment of traceability documents (Ramesh & Jarke, 2001). It is intended to ensure alignment between stakeholder requirements and the various outputs of the system development process (Ramesh & Jarke, 2001). Traceability is the characteristic of the system in which requirements are clearly linked to their sources and the artifacts created during the system development life cycle based on those requirements (Ramesh & Jarke, 2001).

The importance of and need for maintaining traces among artifacts are well documented in the literature (Gotel & Finkelstein, 1994; Pohl, 1994). Prior literature describes the impact of poor traceability practices on project costs and time (Domges & Pohl, 1998). Decrease in system quality, increase in the number of changes, loss of knowledge due to turnover, erroneous decisions, misunderstanding and miscommunication are some of the common problems that result due to lack of or insufficient capture of traceability information (Domges & Pohl, 1998).

Types of Traceability

Prior literature recognizes the distinction between pre and post-traceability. Traceability of the refinement, deployment and use of a requirement is termed as post-traceability, and the traceability of a requirement back to its origin is termed as pre-traceability (Pohl, 1996a). These two types of traceability, also referred to as forward and backward traceability are considered to be equally important (Gotel & Finkelstein, 1994; Pohl, 1996b).

Prior research also makes the distinction between horizontal and vertical traceability (Gotel & Finkelstein, 1994). Horizontal traceability is defined as the ability to trace dependent items developed during any one phase of the software development life cycle. Vertical traceability is defined as the ability to trace dependent items across artifacts generated during different phases in the development life cycle.

Establishing traceability

The process of establishing and using traceability pervades throughout the entire software development life cycle. As requirements are elicited and recorded in the requirements specification document, these requirements should be linked to the stakeholders from whom these requirements were elicited. At a later point, it should be possible to trace any requirement to any discussion with the stakeholders who initiated the requirement. As these requirements in the specification document are analyzed and specified using CASE tools and languages like Unified Modeling Language, traces should be established between the requirements and the elements in the system specifications. When design models are created, traces among requirements and specific design elements that accommodate these requirements, should be established and maintained. Such traces ensure the completeness of the system. As the design is transformed into code, tracing should continue to link specific design elements to code segments or components. Links across test cases and requirements should also be established. At each stage, apart from just establishing links across related artifacts, justifications for various decisions taken should also be recorded. Recording such a history of changes will be useful in supporting different stakeholders for varying purposes (Ramesh & Dhar, 1992). During maintenance stage, traceability knowledge that has been acquired during earlier stages is used to locate the design elements and code segments that need to be changed to handle changes in requirements or to correct errors, and study the impact of changes. As the system is changed during the maintenance phase, traceability knowledge should be constantly updated to ensure consistency across the documentation and code.

The following questions are considered to be key in establishing traceability (Pohl, 1996a):

- What kind of information is to be recorded?
- How to structure the information?
- How to capture the information?

In an attempt to answer the above questions, researchers have focused extensively on developing traceability models that guide the acquisition and use of traceability knowledge. Issue-based models like IBIS (Conklin & Begeman, 1988) and ReMap (Ramesh & Dhar, 1994) are some examples of such efforts. Recent research has also focused on empirically grounding the traceability models (Ramesh & Jarke, 2001).
Prior literature on traceability also discusses several ways of establishing traceability. Traceability matrices (Davis, 1990), hyperext linking requirements to descriptions of domain objects (Kaindl, 1993), and cross-referencing by tagging Evans, 1989 #22; Lindvall, 1996 #24] are some of the common traceability techniques that are currently used in practice (Gotel & Finkelstein, 1994; Kean, 1997). Past research has also discussed about different kinds of traceability tools that are used in practice. General purpose tools, special purpose tools, and workbenches are some of the classes of traceability tools that are currently in use. General-purpose tools include word processors, spreadsheets, database management systems, and hypertext editors that can be configured for traceability purposes (Gotel & Finkelstein, 1994). Special purpose tools include those that focus on well-defined activities in requirements engineering. Some examples of such tools include KJ Editor (Takeda, Shioml, Kawai, & Ohiwa, 1993), which traces ideas to requirements, PORC (Langford, 1991), which provides traceability between interview transcripts and requirements, and T tool (Sodhi, 1991) that traces requirements to test cases. When a collection of such tools is used in concert to provide less restricted traceability, Gotel and Finkelstein term those as workbenches. While emphasizing the importance of project-specific trace data type support, Domges and Pohl (1998) review the capabilities of currently used tools and argue that these tools do not focus on providing support for project specific adaptation of traceability. They note that most of the current traceability tools provide a predefined set of data types from which the project managers can select a subset appropriate for a project.

In summary, it is observed that while there are a variety of tools that strive to provide traceability support, they do not easily facilitate project-specific adaptation by supporting different models and semantics. More importantly, there are very few concrete guidelines for project managers in selecting the appropriate strategy for traceability practice. The problem is even more aggravated when we consider the need to tailor traceability practice depending on the phase of the development process and the tasks involved in establishing and using traceability.

As an example, let us consider the Rational Unified Process (RUP). RUP is a software engineering process and a disciplined approach to assigning tasks and responsibilities within a development organization1. RUP comprises of various phases and disciplines. Business modeling, requirements, analysis and design, implementation, testing, deployment, change and configuration management, project management, and environment management are the various disciplines that are part of RUP. Focus on these disciplines differs depending on the phase of the project. Each discipline involves different kinds of tasks that are to be performed by various stakeholders. For example, during business modeling, business analysts study the current business processes that are followed by communicating with the customers. They try to understand the various business rules involved in developing the new system. During analysis and design, the software architect has to identify the appropriate classes that handle various requirements. The architect has to make sure that all the requisite classes have been identified so that all the requirements that are to be implemented in the current iteration are handled. These two tasks, one in the business modeling discipline, and the other in the analysis and design stage, have different characteristics in terms of uncertainty involved, complexity in terms of number of information cues required to perform the tasks, and thereby the need for a specific type of communication among the stakeholders involved. Such differences in these tasks should be recognized while developing the guidelines for traceability practice. We argue that tailoring traceability practice for different classes of tasks that pervade the software development process will enhance the effectiveness in acquiring and using traceability knowledge.

In the following sections, we draw from the media selection literature to identify the appropriate traceability practice for different types of tasks.

BACKGROUND ON MEDIA SELECTION

Past research in media selection has explored the extent to which media characteristics need to match task characteristics (Daft & Lengel, 1986; Daft, Lengel & Trevino, 1987). Carlson and Davis (1998) characterize richness of a medium as the capacity for a medium to convey rich information, that is the ability to give immediate feedback, variety of communication cues available, language variety attainable and personalization of the medium. Social presence refers to the degree to which a medium allows a user to establish personal connection with the other users (Short, Williams, & Christie, 1976). A high presence medium is rated toward the sociable, warm and personal end of the continuum. The theory further postulates that the level of social presence needed by a particular communication task determines the use of a medium (task-medium fit hypothesis).

Researchers have used Media Richness and Social Presence theories to rank the media in order of increasing media richness and social presence respectively. Media Richness theory postulates that media selection depends on the equivocality and

---

1 RUP is explained in detail elsewhere (http://www-306.ibm.com/software/awdtools/rup/)
uncertainty of the task at hand. Social Presence theory postulates that selection of media is based on the degree to which social presence is necessary for a particular communication task. These two theories together have been grouped under “Trait Theories of Media Selection” because of the similarity of their approaches to media selection (Carlson & Davis, 1998).

However, empirical evidence so far has been mixed for both of these theories especially in the case of media selection (Dennis & Kinney, 1998; Webster & Trevino, 1995). Two reasons account for this state of affairs: evolution of capabilities of newer media and the inadequate attention paid to the role of social influences. For example, e-mail was traditionally thought to be a lean medium; however, this has changed with the evolution of email from primarily a text-based medium to a richer multi-media enabled medium. In addition, Markus (1994) showed that e-mail, even if it is text based (lean medium), can be used for richer communication when the social processes surrounding media use define it as a rich medium as users over time ascribe certain characteristics to the media that increased the richness of the medium. Recent research has argued that individual media characteristics need to be examined separately to understand media choice and use (Dennis & Kinney, 1998; Te'eni, 2001). Hence, in this research we use individual media capabilities to match tasks with media. We use task characteristics like task complexity and the level of granularity of information needed to perform the task effectively, as the basis for selecting media. We characterize task complexity in terms of objective task qualities (Campbell, 1988). Number of acts to be performed to fulfill the task and the number of information cues needed to perform these acts dictate the complexity of a task (Campbell, 1988; Wood, 1986; Wood, Mento, & Locke, 1987). Level of granularity refers to the level of specificity of information needed to perform the task. Based on these task characteristics, we characterize the media requirements through information richness and social presence. We use the term “information richness” to refer to the ability of the media to represent multiple cues and language variety. We also argue that one needs to look at “social presence” separately to understand the extent to which the presence of the other partner(s) is needed in a particular situation.

FACTORS AFFECTING MEDIA SELECTION FOR TRACEABILITY

To illustrate how several factors have to be considered in establishing an appropriate traceability practice, we consider some illustrative traceability tasks that need to be performed when a software development team follows the Rational Unified Process (RUP). Table 1 shows a list of tasks for each discipline in RUP. This list of tasks is intended to be illustrative rather than exhaustive. The table uses granularity and complexity of tasks as the basis for determining the choice of the medium. The table also shows that based on certain task characteristics variables like social presence and information richness vary. These characteristics have been specified in the table for a particular type of system. Based on the system under consideration, these characteristics may vary.

IMPLICATIONS FOR RESEARCH AND PRACTICE

The table helps provide an initial understanding of media choice for traceability based on prior literature on medium-task fit. Various stakeholders involved in the software development process can use these as a set of guidelines to select the appropriate media for specific tasks based on the tasks characteristics. Such a selection of appropriate media will tend to reduce the overheads involved in establishing traceability. Developers of process support tools for software development may use these guidelines to semi-automatically direct the stakeholders to appropriate media based on the characteristics of the tasks that are executed on their process platforms. These prescriptions will help traceability tool developers in delivering traceability support that is tailored to specific tasks involved in different types of projects.

This research takes the first step in establishing the importance of selection and use of appropriate media in traceability practice. However, impact of the changes in the characteristics of system under consideration on the task and media characteristics should be examined and a task taxonomy that incorporates these considerations should be developed. Also, the efficacy of the task taxonomy, the impact of task characteristics on characteristics of media to be selected, and thereby on performance in traceability tasks needs to be tested in an empirical context; then, the recommended medium can be integrated as a part of the traceability support system in such a way that the optimal medium will be made available for communication while keeping the flexibility of choice. Such (semi)-automation of media selection based on task characteristics becomes challenging due to the need to specifically identify, understand, and explicate the various task characteristics. Also, there may be exceptional scenarios where the impact of task characteristics on characteristics of media to be selected is counterintuitive. We are currently investigating the feasibility of such media selection in traceability support environments. We are also examining the applicability of these guidelines for other complex system development tasks, apart from establishing traceability, as media selection for traceability may impact the media selection for other related activities due to inherent interdependencies.
<table>
<thead>
<tr>
<th>RUP Discipline</th>
<th>Stakeholders involved</th>
<th>Example tasks</th>
<th>Task characteristics</th>
<th>Media and Info. Characteristics/Needs</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| Business modeling | Business process analyst, Business reviewer, Business designer | The business analyst is trying to understand a particular part of the business process. S/he is concerned if this part of the process will remain the same when the software system is implemented. S/he needs to discuss this with the client stakeholder that is responsible for this part of the process. | Low in complexity  
Coarse level of granularity | High social presence  
Low in information richness | Fine-grained information is not needed. A discussion with the client will be better, as the amount of information needed is not high. |
| Requirements | Systems analyst, software architect, requirements specifier, User interface designer, requirements reviewer | The business analyst identifies a conflict between two business rules. S/he needs to verify which rule is accurate or how to resolve the conflict. | Medium in complexity  
Medium in level of granularity | High social presence  
Medium information richness | Specific information is needed. Also, dependencies across rules are important. But confirming this with the client in a discussion is essential. |
| Requirements | Systems analyst needs to manage dependencies across requirements. Attributes are assigned to requirements. For example: what are the benefits of a requirement? What is the effort needed for a requirement? | Systems analyst needs to manage dependencies across requirements. Attributes are assigned to requirements. For example: what are the benefits of a requirement? What is the effort needed for a requirement? | Medium in complexity  
Coarse level of granularity | Medium in social presence  
Medium in information richness | Systems analyst will have to communicate with the customer, software architect, etc., to elicit the various attributes of requirements. |
<table>
<thead>
<tr>
<th>RUP Discipline</th>
<th>Stakeholders involved</th>
<th>Example tasks</th>
<th>Task characteristics</th>
<th>Media and Info. Characteristics/Needs</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis and design</td>
<td>Software architect, architecture reviewer, designer, design reviewer, implementor, database designer, integrator</td>
<td>Software architect identifies analysis and design classes. S/he has to make sure that all the requirements are covered by establishing links between design elements and requirements</td>
<td>High in complexity Fine level of granularity</td>
<td>Low in social presence High in information richness</td>
<td>Any stakeholder who needs to check for completeness may prefer to check the traceability documentation, which documents the links at a fine level of detail rather than communicating with an architect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Designer uses several design patterns while designing the classes. Use of such design patterns have to be justified.</td>
<td>High in complexity Fine level of granularity</td>
<td>Low in social presence High in information richness</td>
<td>A maintainer may find the justification useful while handling changes. Also, since use of design patterns inappropriately may impact performance, explicitly justifying the use of patterns may surface assumptions that might question the use.</td>
</tr>
<tr>
<td>Implementation</td>
<td>Software architect, designer, code reviewer, implementor, integrator, tester</td>
<td>Implementor, while producing source code in accordance with the design model, should establish links between source code and design elements.</td>
<td>High in complexity Fine level of granularity</td>
<td>Low in social presence High in information richness</td>
<td>A stakeholder fixing a defect will find it useful to refer to the traceability document rather than communicate with the implementor to understand the impact of a change.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Integrator needs to identify specific scenarios and the classes related to these scenarios while selecting the incremental subsystem that is to be integrated in a particular iteration.</td>
<td>High in complexity Fine level of granularity</td>
<td>Low in social presence High in information richness</td>
<td>Integrator may use traceability knowledge to locate the classes and subsystems related to specific scenarios.</td>
</tr>
<tr>
<td>Testing</td>
<td>Test manager, test analyst, test designer, tester</td>
<td>A test analyst has to link the test ideas generated to appropriate units that are to be tested.</td>
<td>High in complexity Fine level of granularity</td>
<td>Low in social presence High in information richness</td>
<td>Identifying the units that need to be linked to the test ideas and the results or defects identified to the units will require a fine level of granularity</td>
</tr>
<tr>
<td>RUP Discipline</td>
<td>Stakeholders involved</td>
<td>Example tasks</td>
<td>Task characteristics</td>
<td>Media and Info. Characteristics/Needs</td>
<td>Explanation</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------</td>
<td>-----------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Deployment</td>
<td>Project manager, deployment manager, course developer, technical writer, Change control manager, system administrator, tester, test analyst, customer, implementor, configuration manager</td>
<td>Implementor needs to develop all software needed install and uninstall the product quickly, without interfering with other applications. Any dependencies on other applications should be documented.</td>
<td>Medium in complexity</td>
<td>Medium in social presence</td>
<td>Dependencies and interferences across applications may be explained to the user along with considerable documentation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The technical writer develops support material to be used by the customer. When interfaces in the system are changed, the technical writer should be able to locate the parts in the support material that should be changed.</td>
<td>Low in complexity</td>
<td>Low in social presence</td>
<td>This involves changing documenting in help files and in customer support material.</td>
</tr>
<tr>
<td>Configuration and Change management</td>
<td>Change control manager, configuration manager, project manager, integrator, software architect</td>
<td>Configuration manager plans how CM activities are to be carried out throughout the project development life cycle. While doing so, s/he needs to understand the traceability needs in conjunction with configuration management.</td>
<td>Low in complexity</td>
<td>Medium in social presence</td>
<td>This would constitute Creating CM policies and plans and explaining them to other stakeholders involved in the project.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When a change control manager receives a change request and after changes are done to any base-lined configuration item, these changes should be linked to the change request.</td>
<td>High in complexity</td>
<td>Low in social presence</td>
<td>Identification of specific design elements/source code elements that are changed, and linking them to the change request is important.</td>
</tr>
<tr>
<td>RUP Discipline</td>
<td>Stakeholders involved</td>
<td>Example tasks</td>
<td>Task characteristics</td>
<td>Media and Info. Characteristics/Needs</td>
<td>Explanation</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------</td>
<td>---------------</td>
<td>---------------------</td>
<td>--------------------------------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Project management | Project manager | A project manager has to develop several plans that would constitute the software development plan. These plans are to be linked to appropriate organizational policies, quality requirements in the organization, etc. | High in complexity  
Coarse level of granularity | High in social presence  
Low in information richness | These are high-level links and discussion among stakeholders across different departments may be required. |
| | Project manager | Project manager is responsible for monitoring the status of the project. Any deviation in the project status from the plan is to be linked to the cause for the deviation. | Medium in complexity  
Coarse level of granularity | Medium in social presence  
Low in information richness | High level explanation of status deviations might involve discussion among stakeholders involved in the project. |
| Environment | Process engineer, tool specialist, and other stakeholders involved in the development who will use these process support tools | Process engineer develops the development case defining the software development process. While doing so, organizational policies and objectives should be linked to the decisions made in selecting process elements. | Medium in complexity  
Medium in level of granularity | Medium in social presence  
Medium in information richness | This involves tailoring a process framework (like RUP) to make it suitable to the project at hand. Such tailoring will involve identification of project characteristics and organizational policies, objectives, quality needs, and linking them to the change suggested in the process. |
| | Tool specialist | Tool specialist installs and sets up tools required for supporting the software development project. Customization of tools might be necessary to suit the current environment. Such customizations have to be justified. | Low in complexity  
Coarse level of granularity | High in social presence  
Low in information richness | Choices and customizations in process support tools are crucial, but at a considerably high level of granularity. This might need discussions among the various stakeholders involved in the development. |

Table 1. Traceability tasks and Factors that affect traceability practice
REFERENCES