Traceability Task Modeling

Paula Laurent
DePaul University
School of Computer Science
Chicago, IL
plaurent@cti.depaul.edu

Abstract

Requirements traceability is defined as “the ability to describe and follow the life of a requirement, in both a forwards and backwards direction”. [1] Once a requirement is documented, traces for every component of the software system that are impacted by the requirement, are constructed and maintained. During later discussions regarding bug fixes and proposed system enhancements, these traces are used for analyzing the impact the proposed modifications might have on the system. Tool support for manual link construction and maintenance is available as part of existing requirements management packages and for automated methods of research prototypes.

Prior research has identified and discussed traceability problems such as problems related to user interaction and the significant effort required to create and maintain traceability links. There have been many case studies, but no formal task analyses or specific research to identify and model tasks that real users must perform as they participate in trace related activities. Without such a model, the development of traceability tools becomes an unpredictable venture without a way to measure their quality or effectiveness. This paper therefore surveys preliminary work on traceability task analysis and proposes a technique for analyzing and modeling traceability tasks.

1 Introduction

Requirements traceability is defined as “the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)” [1]. Traces can provide details about the requirement’s origin, design, development and testing. They help to ensure that an application satisfies the agreed upon requirements [2]. Later when application and system modifications are proposed, the traces provide information for impact analysis. Traceability is needed to ensure that adding enhancements or fixing reported bugs won’t adversely affect the current system.

Traceable requirements helps the development team to monitor and control the software applications they create and maintain by supporting tasks such as requirements validation, impact analysis, and compliance verification. Tool support for manual link construction and maintenance is provided as part of existing requirements management toolkits such as Telelogic’s DOORS and IBM’s Rational Requisite Pro, while support for automated methods is provided in research prototypes such as Poirot [3] and RETRO [2].
Previous studies have extensively identified and discussed traceability problems [1] such as problems related to communication between stakeholders and the excessive effort required to construct and maintain traceability links. Ramesh et al have studied business motivations for traceability [4] and have constructed detailed metamodels for requirements management, contribution structures, design allocation, and compliance allocation. These models ask questions such as ‘who will perform the trace?’ ‘which artifacts will be traced?’ and ‘what is the purpose of the artifact being traced?’ and the business motivations for traceability.

However no one has attempted to identify and model tasks that real users must perform as they participate in trace related activities. Without such a model, the development of traceability tools becomes a hit-or-miss venture with no yardstick by which to evaluate their quality or effectiveness. In fact, the strawman version of the grand challenges of traceability document [5] identifies the following challenges related to Human Factors in software traceability.

F-GC1 The creation of traceability tools that meet the stakeholders’ needs based on studies of their current usage.
F-CG2 Recognizing that users will make mistakes during the traceability process, functionality to help prevent mistakes and minimize their impact.
F-CG3 The creation of usability tools that inspire user trust.

There are four general usability concepts/objectives, usefulness, ease of use, learnability and likability. This paper proposes using task analysis and modeling to contribute to the design and development of a useful automated traceability tool - a tool that 1) performs tasks that users need to have done and 2) generates valid results. The paper first surveys existing analysis and modeling techniques, it then identifies a few critical traceability tasks, and finally illustrates the task modeling approach through analyzing and modeling one specific traceability task.

2 Task Analysis and Modeling

Task analysis is the process of detailing what it is that users need to accomplish, more commonly referred to as goals; and all of the steps that the user performs to complete these goals. Some of these steps will be tool specific, while others represent more generic activities that must be supported by any traceability tool. During task analysis, the developer assembles information about the user’s current goals and activities; what’s working and what isn’t. The developer notes missing functionality and what can be improved on, as well as any system work-arounds the user has devised and current functionality that’s not being used.

Task modeling and analysis is a methodology borrowed from the psychology domain, where it was used for examining the structure of work. The first task analysis method, Hierarchical Task Analysis, (HTA), represents task structure hierarchically, in either of two forms. Text-based descriptions are noted using indentation and numbering to signify task and their subtasks. Hierarchical trees are used as graphical descriptions [6]. The two formats are shown in figure 1.
There are several tools that usefully support task analysis. Model-Based Interface Designer, (MOBI-D), is an interface modeling environment. U-TEL is a tool within MOBI-D that lets users create an outline of specific tasks by typing in task descriptions and then categorize them by key terms. These outlines can then be used by the developer to build user task models [7]. HTA is the approach incorporated in the engineering model of human performance known as GOMS (Goals, Operators, Methods and Selection Rules) [8]. GOMS is another methodology that can be used for task analysis.

Task analysis can be performed in a variety of ways. There’s the “pure” observation technique, where researchers observe the users and take notes. This is different from the “day-in-the-life-of” observation technique where the researcher and user discuss what’s happening during the observation period. The day-in-the-life-of technique allows the researcher to probe for explanation and clarification while watching and noting how the user accomplishes their tasks. Interviewing users is another option as is using tools that allow the users to narrate or document their tasks.

Models allow people to deal with real-world complexities by helping to identify crucial aspects and relationships. System designers use models as structural descriptions of pertinent information. Unified Modeling Language, (UML) is the most popular software engineering modeling technique. It focuses on creating objects and then object activities [6].

Task models can be used for designing either an entirely new application or part of an existing one. Using these models is especially necessary when designing systems because they characterize the logical activities that support the users in accomplishing their goals [6]. Once the models are analyzed and understood, then system design and development can begin. “Task models help to ensure that the resulting interface directly and transparently supports the user in performing their task. The appropriate level of abstraction for task models is determined by a tradeoff between the desire for broad coverage and the need to provide as much specific information as possible about the functional behavior of the corresponding interface entities.” [9]

Since there are many project stakeholders, (client, user, developer, QC, etc), associated with any software system, several task models will be needed to represent the high-level goals of the variety of potential users. After thorough analysis of the traceability tasks, traceability task models can be constructed.
2.1 Sample Traceability Task Model

Figure 2 depicts a high-level traceability task model for accomplishing a user’s goal of finding the links for requirement 7. Three tracing tasks must be performed: find the links from the requirements to 1) design documentation, 2) system source code and 3) documented test cases. GOMS was used for the task analysis to create this model.

![Sample Traceability Model Diagram]

**Figure 2. Sample Traceability Model.**

3 Next Steps

Future work will include a more thorough analysis of traceability tasks. Task analysis is needed for reviewing the current requirement traceability tools, their usage and the gap analysis of current functionality. The resulting task models will then be used to refine the design and development of new traceability tools and functions.
References


