

Implementing Requirements Traceability: A Case Study

B. Ramesh, T. Powers and C. Stubbs
Code SM/RA
Naval Postgraduate School
555 Dyer Road
Monterey, CA 93943
E-mail: ramesh@nps.navy.mil

M. Edwards
Code B40
NSWCDD
10901 New Hampshire Avenue
Silver Spring, MD 20903
E-mail: medward@relay.nswc.navy.mil

Abstract

Many standards that mandate requirements traceability as well as current literature do not provide a comprehensive model of what information should be captured and used as a part of a traceability scheme. Therefore, the practices and usefulness of traceability vary considerably across systems development efforts, ranging from very simplistic practices just aimed at satisfying the mandates to very comprehensive traceability schemes used as an important tool for managing the systems development process. In this paper we present a case study of a systems development organization employing a comprehensive view of traceability. A model describing the traceability practice in the organization, perceived benefits of such a scheme and lessons learnt from implementing it are presented.

Keywords: requirements traceability, requirements traceability tools, design rationale, requirements engineering

1 INTRODUCTION

Requirements traceability refers to the “the ability to follow the life of a requirement, in both forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases” [1]. Requirements traceability is used to capture the relationships between requirements, design, and implementation of a system. All system components (hardware, software, humanware, manuals, policies, and procedures) created at various stages of the development process are linked to requirements. Traceability provides stakeholders with a means of showing compliance with requirements, maintaining system design

rationale, showing when the system is complete, and establishing change control and maintenance mechanisms.

Many standards for systems development such as the U.S. Department of Defense (DoD) standard 2167A mandate that requirements traceability be practiced. U.S. DoD standard 2167A insists “that the functional requirements which are identified as a part of the functional baseline be traceable directly to specific capabilities within the allocated baseline, which must then be directly traceable to specific capabilities within the product baseline” [2], [3]. The standard requires that system requirements are traceable throughout the development process without much elaboration on the type of information to be maintained to achieve this [4].

Although requirements traceability has been in practice for over two decades, there has yet to be a consensus on what information should be captured and used as a part of a traceability scheme. There are many different definitions of traceability, each changing with a stakeholder’s view of the system. Stakeholders could be the program sponsor (customer), the project manager, the system analyst/designer, the test engineer, system maintenance personnel, or the end user of the system. Through the System Development Life Cycle, the stakeholders’ definition and view of traceability changes. For example, to the customer, traceability could mean being able to ascertain that the system requirements are satisfied. The maintenance engineer’s primary concern with traceability may be how a change in a requirement will affect a system, what modules are directly affected and what other modules will experience residual effects. Some of these stakeholders’ views or definitions of traceability overlap. The practice and benefits of traceability vary considerably among large scale system development efforts. Whereas many organizations look at traceability as a mandate to be satisfied, some organiza-

tions view traceability as an important component of implementing a quality system engineering program. In this paper, we present a case study of an organization in the later category discussing its practice of traceability and lessons learnt from implementing it.

2 CASE STUDY

2.1 The Organization

The case study focused on the use of requirements traceability by the Weapon System Technology Support Branch at the McClellan Air Force Base, a U.S. DoD organization (hereafter referred to as WST) in their day to day operations as well as on a specific project (referred to as ADIP). The main focus of the study was to determine the impact of traceability on the ADIP and the actual and perceived benefits of traceability as viewed by the WST. Several on-site interviews and observation sessions at the WST were the primary source of data for this research. One-on-one interviews were conducted with the organization's upper management, the project manager, system designers, and test/audit personnel to determine how the various stakeholders view traceability, to what extent each uses traceability in their daily job tasking, and the perceived benefits of using traceability.

The WST was chosen as the subject of this case study after an extensive search for an organization that uses traceability, not only to satisfy the standards or project sponsor's requirements, but as an important component of its systems development and management activities.

The mission of the WST is to provide embedded computer systems and software support for projects generated by system program directors, other DoD Agencies, and private industry. The organization has achieved a level 3 rating under the Software Engineering Institute's (SEI) Capability Maturity Model (CMM). The WST consists of a Branch Chief, two Section Chiefs, System Engineers and contract employees who provide support for the traceability CASE tool used by the organization.

WST acts as an independent contractor when taking on projects for other divisions and branches as well as external organizations. As its operations are similar to those of commercial organizations it competes with, the lessons learnt from implementing traceability at WST can be expected to be applicable to commercial settings as well.

2.2 The ADIP Project

The ADIP project was contracted out to the WST. The initial Statement of Work (SOW) required that a flight control software program (referred to as OFP) for a jet aircraft be redesigned from Jovial programming language to Ada, incorporating enhancements. The ADIP contains approximately 75,000 lines of code and over 3,000 requirements. The new system was required to be functionally equivalent to the existing one so that the pilots will not require total retraining.

2.3 Requirement for Use of Traceability

The SOW mandated requirements traceability across the entire project. Traceability was required from the Product Specification to the Interface Requirement Specification and Software Requirement Specification. These specifications were further traced to system components.

3 REQUIREMENTS TRACEABILITY PRACTICE

The WST has embraced requirements traceability methodology as vital in their day to day operations as well on specific projects. It views traceability as a mechanism that will ease the task of life-cycle maintenance.

3.1 Traceability Model Employed by WST

Though WST has not developed a formal methodology for implementing traceability, an information model can be used to convey the semantics of the various types of traceability information being captured and used by the organization. Figure 1 identifies various types of traceability information used in areas such as requirements rationale capture, design rationale capture, allocation of requirements to system components, and use of resources by various system components¹. As the ADIP has not reached the testing stage, the model represents the traceability mechanisms WST plans on using during that stage of development.

The project involves re-engineering, with very little information available on original requirements, the organizational level objectives the system is trying to

¹For brevity, only salient aspects of the model are discussed in this paper

address, and so on. In this effort, therefore, it is nearly impossible to "trace back from requirements" [1] to their sources. WST, however, maintains rationale behind requirements, "reengineering" them whenever feasible. Throughout WST, stakeholders create source documents capturing various types of traceability information identified in the figure. These documents include design rationale and requirements rationale through the engineer's notebook and requirements tracing through traceability matrices. Though WST requires the capture of design and requirements rationale, this information is not captured in any pre-defined format. WST requires documentation on how requirements are defined by stakeholders or modified by change proposal requests. Further, higher level requirements are iteratively refined to derive lower level requirements. Requirements that identify system constraints and dictate the system design activity are explicitly identified. Also, traceability information on how the requirements are allocated to the system components is captured.

3.2 Upper Management

The upper management of the WST views the use of requirements traceability as a must for survival. According to the management "Requirements traceability is not an option, it is essential to keep the customer happy. Traceability ensures customer satisfaction by providing us a documented means by which to prove to the customer that all of the stated requirements are met and that the job is completed".

Equally important, from their standpoint, was the need to minimize the possibility of missing a stated or derived requirement in the process of developing large, complex systems. In the case of the ADIP, missing even a single critical requirement could be catastrophic to the pilots flying the aircraft. Management uses traceability in evaluating and accepting potential projects. The WST first identifies the requirements for a proposed system from the customer's project management plan. These requirements are then referred back to the customer to ensure that the interpretation by WST is accurate. With a complete list of validated requirements, management estimates the size and scope of the project. Using heuristics, management determines the projected staffing level required for the project and ultimately, developing a bid for the project. Traceability information relating requirements and these projections are maintained to assess impacts of changes. Management also uses traceability as a work management tool in the daily operation of the office. The traceability matrix provides

the manager a means of tracking staff progress on the project. By tracing the requirements down to the Computer Software Unit (CSU) level, management can readily assign tasks and track completion status. For instance, a traceability matrix provides management with status of a module of the system being developed, with information such as completion status and projected completion date for the various tasks. This information is used for tracking and projecting budget information such as manpower expended for a specific work period, on various stages of the system's development. This information assists in change management, as well as in projecting future workload. The hours expended would be used to project the estimated time it would take to effect a change on a specific section of code and the cost of implementing a change.

3.3 Project Manager

The project manager believes that proper use of traceability provides a means of showing s/he is in full control of the project. The project manager tracks the original as well as derived requirements to the CSU level. This information is used to assure the customer that all requirements are understood and validated, that derived requirements are documented and validated, and that the resulting system design will meet all of the stated requirements.

Through the design phase, the project manager uses traceability to track project status similar to the way upper management does, only in more detail. The project manager is more concerned with the daily progress of the production staff, whereas upper management's concerns are more in meeting deadlines. A GANTT chart generated from the traceability information is used by the project manager in developing weekly project status reports. A GANTT chart is completed by each engineer with the rows representing tasks assigned and the columns representing weekly work periods. Also, traceability helps the project manager in detecting project delays. By tracing requirements to the CSU level, derived requirements become readily apparent and are entered into the system.

Once the project reaches the testing phase, the project manager plans to use traceability to verify and to prove to the customer that the system meets the stated requirements and the job is complete. By using traceability to acceptance test plans for every validated requirement, including derived requirements, the project manager can prove to the customer that the system "completely" meets their needs.

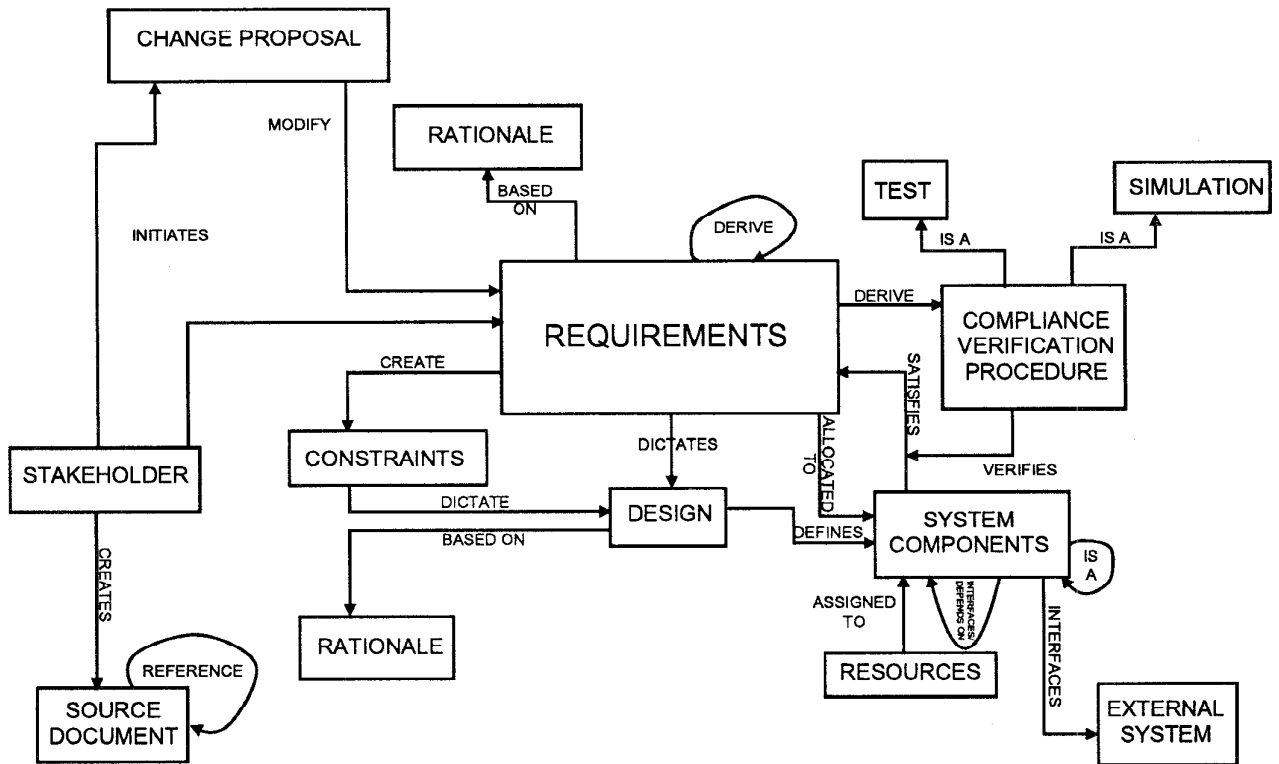


Figure 1. Requirements Traceability Model

Upon completion of the ADIP, the project manager intends to use the traceability extensively throughout the system maintenance effort. It is planned that upon receiving a request for a change to the system, the project manager will track the requirement being changed through the use of Ada Structure Graphs (ASGs) maintained in the CASE tool to determine the extent of the proposed change. This will provide the project manager with a means of estimating costs for changes, which can be relayed to the customer so a cost-benefit analysis can be completed.

3.4 System Designer/Engineer

The most significant use of traceability throughout this project was observed through the system engineer's "Engineer's Notebook." The data captured is in free text form, which requires a disciplined engineer to ensure effectiveness. The notebook captures the engineer's design rationale explaining why the system was designed the way it was. This information could prove invaluable throughout life cycle maintenance and on

the development of similar systems. Although the project is not at the stage of maintenance, the system designer foresees using requirements traceability extensively in tracing changes to code modules and documentation. The system designer plans on using the traceability information captured in the CASE tool to trace proposed requirement changes to the CSU level, thus identifying which modules a change will effect. This will greatly enhance the system maintenance effort by providing an automated means for capturing the system's design rationale and residual changes caused by the change of a requirement. The system designer also plans on using traceability to determine which test plans and documentation are effected by a change so that they could be updated and rewritten. The software designer used traceability as a "fit and function" verification tool. The system requirements, as stated when the system was originally written in Jovial, were verified as still being correct. The system architecture was then developed using Ada Structure Graphs (ASGs). Using the traceability tool, the original system functional requirements were mapped to

the ASGs, providing a level of confidence that the Ada system design was complete.

3.5 Tester/Auditor

The system testers plan on using traceability in writing the acceptance test plan. Making use of the traceability tool, the testers will verify that the Acceptance Test Plan tests all of the system requirements, thus ensuring completeness and that it operates as it is designed to, while meeting all of the customer's requirements.

3.6 The use of CASE tool

The WST had recently purchased a CASE tool to assist with their traceability efforts. At the time of the case study, WST personnel were still learning the benefits available from the tool. As they become more familiar with the CASE tool and its applications, their traceability efforts will become more thorough. A major concern of upper management was the initial outlay of funds for the CASE tool and the amount of time required to train WST personnel on its use. These were viewed as sunk costs that would prove beneficial "down the road" but not on the initial project.

4 LESSONS LEARNED

In this section, we present several observations from the case study that could potentially benefit other organizations interested in implementing a comprehensive traceability scheme.

4.1 Definition of Traceability

The first step in implementing a traceability scheme is the identification of traceability information that the organization intends to capture and use. WST has not articulated a formal methodology for implementing traceability. It should be noted that the absence of a common and well defined perspective on traceability is a major factor that contributes to the wide variation in the current practices. For instance, in ADIP, the engineer's notebooks that include design rationale information are maintained in a free form text, and the level of detail and the type of information captured vary very widely among engineers. Standardization of the form and content of such information will ensure that the data captured is useful and consistent. ADIP has addressed the issue by defining templates for several reports that are produced as a part of the traceability scheme.

4.2 Loss Leader Strategy

An organizational decision was made to venture into the use of a traceability CASE tool as a long term commitment, realizing that the cost of learning and using the CASE tool could never be recouped in the initial project. It was accurately predicted that the steep learning curve associated with the CASE tool would result in drastic schedule delays on the initial project. To introduce the use of a CASE tool for traceability, the ADIP was identified as a loss leader project. In doing this, management selected an initial project that could afford the time delays that were expected. Management recognized the long term benefits of using traceability in project development and was willing to suffer the initial "growing pains" associated with incorporating this discipline into their corporate views. In selecting a relatively small scale project that did not have stringent deliverable requirements, WST has identified a successful implementation strategy. WST identified the tangible benefits of this project as embracing requirements traceability across the systems development life cycle and learning how to use the traceability CASE tool as opposed to a deliverable of a new system.

4.3 Re-engineering Efforts

The re-engineered ADIP system was required to work with existing hardware with changes transparent to the end user. The original system contained very little documentation and no detailed traceability. The first step in the re-engineering process was to identify the original requirements of the system. This involved identifying low level requirements resulting from various levels of decisions, such as hardware and interface design decisions. The development team conducted and reviewed trade studies, studied operators manuals, and examined code line by line in an effort to understand the original requirements and design. Ultimately, the staff had to back-hire engineers from the initial project, at a significant cost, to assist in determining the detailed requirements. Had the initial system been developed with traceability, the majority of this time and effort could have been saved. It was estimated that ten employees lost over six months of productive work time, resulting in over 60 lost work-months. With shrinking budgets, legacy systems are becoming more and more common throughout (specifically, in the U. S. Government). With reengineering efforts, such as those mandating the use of Ada in any significant system rewrite, the problems experienced by the WST (having to "discover" lower level

requirements and missing requirements and design rationale) will become common. WST experience suggests that developing systems that provide comprehensive requirements traceability will greatly enhance the efforts of re-engineering those systems at a later date.

4.4 Traceability for Hardware Upgrades

The ADIP system is already under consideration for hardware upgrade. The traceability effort has captured detailed hardware-software interface requirements of the system. This effort is expected to provide immediate payoff during the hardware upgrade. With rapid advances in the hardware technologies, frequent hardware upgrades are becoming common in large scale systems. The ADIP experience suggests that with the development of any software upgrade under a requirements traceability methodology, the resulting documentation will make it much easier to implement a hardware upgrade, especially in complex embedded systems. Further, it may be advantageous to even re-engineer some of the requirements traceability information during a system upgrade.

4.5 CASE Tool Compatibility

WST believes that requirements traceability information should be maintained and updated throughout the Systems Development Life Cycle. Much of the information capture occurs early in the life cycle and some of the major uses occur in later phases. Also, with very large scale complex systems, different components may be developed and maintained by different organizations using different tools. As current CASE tools do not share traceability information well, in the ADIP situation, a follow-on project that does not use a compatible CASE tool as is presently being used could render much of the information captured of little benefit. Therefore, ability to share traceability information across different platforms and tools should be an important consideration in the choice of a tool for large scale systems development.

4.6 Choice of CASE Tools

In the ADIP several reports and documentation associated with traceability can not be produced with the CASE tool package being used. The tool does not provide a way to incorporate project management related information (schedule, budget, etc.) that are used as a part of the traceability scheme. Mechanisms to aggregate this information from lower lev-

els to higher levels and vice versa are required. For example, schedule reports produced by the engineers were passed to the project manager, who then had to manually consolidate them into one report. Secondly, budget information by the manager had to be manually reentered into the system by lower level workers. WST has created templates of various types of information required to be captured and entered them into the CASE tool. Facilities for automatically capturing standard information such as name, project/module, or other common characteristics greatly enhances the usefulness of templates, freeing the user from mundane data entry. The automatic capture of all possible relevant information is viewed by the users as a necessary requirement in a tool to support traceability.

4.7 Political Issues

With the detailed amount of documentation associated with requirements traceability, concerns arise over a *political* issue: the information provided by the system development personnel may be used by the management in performance evaluations. Management at WST handles this by instituting a "Team Responsibility" philosophy throughout the shop. As an engineer completes a portion of the project, the other engineers would review the work and documentation as a team. All products are considered team products and no individual has to worry about being punished.

4.8 High Costs of Traceability

Planning for the increased workload and documentation required for implementing requirements traceability, the initial budget for the ADIP planned for twice the normal documentation costs associated with developing a system of that size and complexity. This estimate still fell far short of the actual costs associated with traceability. However, management calmly accepted these costs viewing them as reducing total life-cycle costs due to development of a higher quality product and reduced maintenance costs. Training the various members of the organization in use of the CASE tool was both time consuming and expensive. However, WST management believes that is most likely a one-time cost that will be more than recovered as the organization continues to practice requirements traceability in their systems development efforts.

4.9 Traceability in Process Improvement

WST management viewed implementing traceability into the organization's systems development methodology as "an important concept in improving the process of systems engineering activity and overall project quality." Additionally, management viewed requirements traceability as an important component in increasing their SEI CMM rating. Traceability is an area where many organizations throughout the systems development industry fall short. Though adopting traceability itself that would not automatically lead to an increase in an organization's process maturity, implementing a comprehensive traceability scheme has provided WST a critical review of the Systems Engineering process and an opportunity to modify those processes that resulted in improvements.

5 CONCLUSIONS

The WST uses requirements traceability as an important quality management tool. From the upper level management to the system maintenance personnel, every person believed that traceability is needed for the successful completion of a project and that without it, their organization's success would be in jeopardy. WST experience suggests that the costs associated with implementing a comprehensive traceability scheme can be justified in terms of better quality of the product and the systems development and maintenance process with potentially lower life cycle costs.

Acknowledgments

This research was supported by the Office of Naval Research Engineering Complex Systems project of the Naval Surface Warfare Center Dahlgren Division.

References

- [1] O. Gotel and A. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings of the First international conference on requirements engineering*, (Colorado springs, CO), pp. 94-101, April 1994.
- [2] U.S. Department of Defense, "Military standard: Defense systems software development, DOD-STD-2167A," February 1988.

- [3] W. Roetzheim, *Developing software to government standards*. Englewood cliffs, N.J.: Prentice Hall, 1991.
- [4] N. Walters, "Requirements specification for ada software under DOD-STD-2167A," *Journal of Systems Software*, vol. 15, pp. 173-183, 1991.