NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A STUDY ON CURRENT PRACTICES OF REQUIREMENTS TRACEABILITY IN SYSTEMS DEVELOPMENT

by

Timothy P. Powers, LCDR, USCG
and
Curtis D. Stubbs, LT, USN

September, 1993

Thesis Advisor: B. Ramesh

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB Np.

1.AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 10 September 1993	3.REPORT TYPE AND DATES COVERED Master's Thesis

4.TITLE AND SUBTITLE : A STUDY ON CURRENT PRACTICES OF REQUIREMENTS TRACEABILITY IN SYSTEMS DEVELOPMENT	5.FUNDING NUMBERS
6.AUTHOR(S) Timothy P. Powers and Curtis D. Stubbs	
7.PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8.PERFORMING ORGANIZATION REPORT NUMBER
9.SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	10.SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (*maximum 200 words*)

The Department of Defense (DoD) currently spends approximately four percent of the total life cycle costs on requirements traceability efforts in large scale systems development. As current DoD standards that require traceability do not clearly specify what information should be captured and used, the practices and usefulness of traceability vary considerably across systems development efforts.

The goal of this research is to conduct a comprehensive study of current practices to provide the various views and uses of traceability by the different stakeholders in the System Development Life Cycle (SDLC).

Using a field study of 35 systems development organizations, this research profiles the "low end" users who use traceability only within their own domain of the SDLC and the "high end" users who view traceability as a means to force higher quality into systems design implementing a traceability methodology across all areas of systems development. Models describing low end and high end uses of traceability practice are also developed. Finally, a detailed case study of a DoD systems development organization was conducted providing a comprehensive view of use and perceived benefits of traceability.

			15.NUMBER OF PAGES 80
			16.PRICE CODE
17.SECURITY CLASSIFICATION OF REPORT Unclassified	18.SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19.SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20.LIMITATION OF ABSTRACT Unlimited

NSN 7540-01-280-5500Standard Form 298 (Rev. 2-89)i
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

A Study on Current Practices of Requirements Traceability in Systems Development

by

Timothy Patrick Powers
Lieutenant Commander, United States Coast Guard
B.S., United States Coast Guard Academy, 1980

and

Curtis David Stubbs
Lieutenant, United States Navy
B.S., University of the State of New York, 1985

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September, 1993

Author: Timothy P. Powers
Timothy P. Powers

Author: Curtis D. Stubbs
Curtis D. Stubbs

Approved by: B. Ramesh
Balasubramaniam Ramesh, Thesis Advisor

M. Bui for
Tung Bui, Associate Advisor

D. R. Whipple
David R. Whipple, Chairman
Department of Administrative Sciences

ABSTRACT

The Department of Defense (DoD) currently spends approximately four percent of the total life cycle costs on requirements traceability efforts in large scale systems development. As current DoD standards that require traceability do not clearly specify what information should be captured and used, the practices and usefulness of traceability vary considerably across systems development efforts.

The goal of this research is to conduct a comprehensive study of current practices to provide the various views and uses of traceability by the different stakeholders in the System Development Life Cycle (SDLC).

Using a field study of 35 systems development organizations, this research profiles the "low end" users who use traceability only within their own domain of the SDLC and the "high end" users who view traceability as a means to force higher quality into systems design implementing a traceability methodology across all areas of systems development. Models describing low end and high end uses of traceability practice are also developed. Finally, a detailed case study of a DoD systems development organization was conducted providing a comprehensive view of use and perceived benefits of traceability.

TABLE OF CONTENTS

I. INTRODUCTION

A. OBJECTIVES OF THE RESEARCH

The goal of this thesis is to conduct a comprehensive study of the current practices of requirements traceability in the systems development industry today. Such a study should provide the various views and uses of traceability by the different stakeholders in the system development life cycle. Additionally, the current Computer Aided Software Engineering (CASE) tools that support requirements traceability will be introduced. The primary objective of this research is to derive lessons learned from these current practices.

Given the above objectives, the following questions are addressed:

- What are the current practices of traceability throughout the system development industry?
- How do the various stakeholders use traceability in their daily work environment and in systems development?
- What are the capabilities of current CASE tools available to support traceability?

B. METHODOLOGIES

The approach taken in this research is to conduct a field study of systems development organizations to determine the current state of practice and assessing the benefits of requirements traceability. Four tools were employed in this research: a literature review, a written questionnaire and one-on-one interviews with various system development organizations, a review of traceability CASE tools used across the industry,

and a comprehensive case study of a Department of Defense (DoD) systems development site.

The literature review provided a thorough background on different aspects of traceability and the documented perceived benefits of using traceability throughout systems development. The interviews with the various system development companies provided a view of the current practices of traceability within the industry as well as how CASE tools were being used in their traceability efforts. The traceability CASE tool vendors provided detailed information and user training on their products, highlighting the capabilities of their tools in support of requirements traceability. User training was provided by the developers of, Requirements & Traceability Management (RTM), Teamwork/RQT, Requirements Traceability System (RTS), and Requirements Driven Development (RDD-100). Finally, a detailed case study of a DoD systems development site was conducted, providing the actual use and perceived benefits of traceability within the systems development life cycle.

C. SCOPE AND LIMITATIONS OF THE STUDY

A variety of stakeholders are involved in the systems development process, including project sponsors, project managers, system designers/analysts, system testers/auditors, system maintenance personnel, and the end users. The approach used in this research to identify the various stakeholders' needs for traceability has been empirical, using interviews of 35 various stakeholders from systems development organizations. Further, our study explores the current practices across the industry to meet those needs.

The study included over 15 organizations that represent the "low end user" of traceability and over 10 others that employ "advanced" or "innovative" uses of traceability, referred to as "high end users." The low end user practices requirements traceability in their systems development efforts, but usually only within their own domain of the systems development life cycle (i.e., the systems designer uses traceability only when designing the system). In contrast, the high end user views requirements traceability as a means to force higher quality into system design and the end product while reducing time expended and life cycle costs, implementing traceability methodology across all areas of the systems development life cycle.

All major CASE tools that support traceability were included in our sample. However, due to the elaborate time requirements for data collection, the population of the study was limited to 35 participants. Though we believe the study provides a representative account of traceability practices, no statistical sampling procedures were used in its design. Due to a lack of evaluation framework regarding requirements traceability, this study was conducted to develop a framework of analysis (i.e., different stakeholder's views, profile low and high end users) allowing a thorough case study to be directed at one systems development organization.

D. ORGANIZATION OF THE DOCUMENT

Chapter II provides background information on the topic of traceability and the benefits of using traceability in the systems development process as discussed in literature.

Chapter III describes the "Low End User" and "High End User" of traceability based on results of the questionnaires completed by industry personnel and interviews. This chapter also provides models describing the low end user's and high end user's practice of traceability.

Chapter IV provides the analysis of the case study conducted. It discusses the major findings and lessons learned and relates them to current literature.

The final chapter presents the summary of the authors' findings and provides recommendations resulting from the research effort.

Appendix A provides background information detailing how different stakeholders view requirements traceability and how they perceive traceability as benefiting systems development.

Appendix B presents the various CASE tools that support requirements traceability. This Appendix is intended to increase contact, awareness and understanding of the CASE tools currently available, not to compare the tools to one another.

II. BACKGROUND

This chapter will present the benefits that the various stakeholders can experience from using requirements traceability and introduce to the reader some of the current requirements traceability CASE tools in use by systems development companies. This introductory material lays the foundation for the analysis of the data gathered in the author's interviews and the case study. The definition of requirements traceability that follows was minimized so as to not duplicate the efforts of other studies being conducted at the Naval Poastggraduate School.

A. A BRIEF DEFINITION OF REQUIREMENTS TRACEABILITY

Although requirements traceability has been in practice for over two decades, there has yet to be a consensus on what traceability really means. 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 effect a system, what modules are directly effected and what other modules will experience residual effects. Some of these stakeholders' views or definitions of traceability overlap.

Appendix A details how various stakeholders may use traceability throughout the Systems Development Life Cycle.

The remainder of this chapter will present the benefits that the various stakeholders can experience from using requirements traceability and introduce to the reader some of the current requirements traceability CASE tools in use by systems development companies. This introductory material lays the foundation for the analysis of the data gathered in the authors' interviews and the case study.

B. EXPECTED BENEFITS OF USING REQUIREMENTS TRACEABILITY

Traceability is used to provide 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.

To be effective, traceability must be carried across the entire system and throughout the entire system development life cycle. Some of the significant benefits of using traceability in system development will not be realized until this view is accepted and practiced.

1. Traceability Helps Compliance with Requirements

Requirements traceability enables all parties to prove the product does what is wanted and does not do anything except what is wanted. During requirement design/code

reviews, traceability provides a means for system developers to prove to their client that the requirements set forth in requirements documentation are mutually understood and that the product will fully comply with those requirements. Also, traceability shows that the product does not contain any additional functionality that has not been identified as a requirement. Oftentimes this additional "goldplating" results in significant additional costs and time delays to the project.

"Quality, as viewed by the customer, is the degree of compliance to their needs the product exhibits." (Wright, 1991, p. 1) Requirements traceability is used to show the system is of high quality by providing a measurable way to ensure that the product will do everything that it is required to do and nothing that it is not required to do. Compliance with the validated requirements is accomplished through reviews and testing. Requirements traceability aids this process by providing the means to identify the relevant components to be reviewed and tested.

2. Traceability Helps Justify Design Rationale

Understanding the why of, or the reason for design decisions is extremely important in various stages of life cycle development. "To understand why a system design is the way it is, we also need to understand how it could be different and why the choices which were made are appropriate." (MacLean, et al, 1989, p. 247) Traceability linkages to design rationale provide "corporate memory" that could easily be overlooked or lost in large projects with changing requirements and personnel.

Capturing decisions, or design rationale, made throughout the system development on how to fulfill requirements will provide an invaluable tool for future designers and maintainers to use when designing a new system or modifying an existing system. Capturing this "corporate memory" should occur throughout the system development.

Traceability of design rationale helps others to understand why a designer did something the way he/she did and what other alternatives were considered in making the decision. Capturing design rationale information eliminates rework and facilitates the understanding of design decisions saving time and money when considering changes to the system.

"Traceability is the ability to discover the history of every feature of a system. It is also being able to find out what resulted from a change request." (Hamilton and Beeby, 1991, p. 1) The history of projects is protected through the use of design rationale documentation. Oftentimes when a system is being evaluated for upgrade or maintenance, there is no one available that was involved in the initial system development. This loss of "corporate memory" leaves the maintainers at a loss. Design rationale is their only valid link to the true workings of the system.

3. Traceability Helps Monitor Completeness of the Project

Traceability provides stakeholders with a measurable means by which to project completion of a system. As modules are tested and requirements are validated as being met, stakeholders are provided a simple and effective metric for tracking project status.

As the requirements are validated as being successfully met, project completion status can be easily ascertained.

Traceability provides not only a method to assess project completion, but also assess completion of the various stages along the way. The system designer may use traceability to show completion of development of a module by ensuring that the requirements mapped to that module are successfully addressed by it. System Designers and Engineers may use this mechanism of ensuring completeness down to the Computer Software/Hardware Component level.

4. Traceability Helps Control and Maintain Change

When considering making a change to the requirements of a system, traceability allows the designer and customer to see the effects of the proposed change. Traceability shows how the proposed change effects other elements of the requirements and the design addressing them. In large scale systems altering requirements or introducing new requirements to an already existing system will cause residual or trickle-down changes to other requirements. Without using requirements traceability, these hidden changes are oftentimes discovered after the proposed change has been implemented, resulting in the astronomical cost of updating systems. Traceability, if properly executed, can readily show the extent of a proposed change, revealing the "hidden" changes.

Once the full effect of a proposed change has been determined, including identifying the hidden changes, the designer will be able to estimate the cost of

implementing the change. This provides the customer the data necessary to evaluate the merit of the change through cost-benefit analysis.

C. Automated Support for Traceability

Due to the enormous volume of data that is captured, requirements traceability can only really be achieved in an automated environment. "There have been many cases where it appeared, at the outset, that it would be an easy task to keep track of it [manually], but when the system design is complete, and the customer is trying to understand whether all the test data really satisfies the original requirements they wrote, the automated traceability would be 'worth its weight in gold'." (Thayer and Dorfman10, 1990, p. 66)

In response to DoD standards of traceability (DoD-STD-2167A), tools which automate the systems development process are becoming increasing popular. As traceability grows in practice, the number and sophistication of available CASE tools also will grow. The degree of complexity of the available CASE tools can vary drastically, from a simple word processor and spread sheet, to a complex integrated system that helps automate a comprehensive system development methodology.

During interviews of systems development companies, the authors gathered data on CASE tools that were actively being used to support requirements traceability. Appendix B provides a brief description of some of these tools.

III. CURRENT PRACTICES IN TRACEABILITY

A. INTRODUCTION

This chapter provides an overview of the current practices in requirements traceability throughout industry, while also exploring any unique or unconventional applications identified during our study of twenty six systems development organizations. Our study intends to profile the "Low End User" of requirements traceability in systems development to establish a baseline of the current practices of traceability. Further, we would also identify and explore those applications of traceability practiced by the more advanced users in industry today, who we refer to as the "High End User." This chapter will profile the low end user and high end user of traceability emphasizing:

- size and complexity of the systems developed.
- experience level of the user with traceability.
- the user's definition of traceability.
- basic applications of traceability.
- the user's requirement for a support tool.

B. METHODOLOGY

In order to understand the practices and become familiar with the capabilities of current tools, the research team attended training courses in some of the leading traceability CASE tools and participated in detailed demonstrations by vendors. List of

clients that use various traceability CASE tools obtained from vendors was screened to identify participants in the study. The participants were limited to those organizations that provide systems development support to the U. S. Government and subject to follow Standards such as DOD-STD-2167A that specify traceability requirements throughout systems development. Initial contact was made with these clients via telephone, defining the scope of our research, including an outline detailing the types of information we were attempting to collect. The organizations developed aerospace, communications, weapons, aircraft, and accounting systems, as well as performed systems integration. This initial interview was followed by one-on-one interview sessions with the participants. All but a few of the thirty seven interviews were conducted at the organization's development site. Due to time, travel, and schedule constraints, the remaining few were conducted over the telephone. The results from these interviews were used to profile the low end user and high end user of traceability.

The size and complexity of the systems being developed that were studied varied from less than 1,000 initial requirements to over 10,0000. The "typical" system contained between 1,000 and 2,000 initial requirements and used an off the shelf CASE tool to track these requirements. Organizations that were developing systems with far fewer requirements used simple spread sheets to manage the limited number of requirements. Most of these smaller systems involved basic requirements, not complex enough to require a more sophisticated methodology for tracking. On the other end of the spectrum were

several organizations which were developing systems composed of over 10,000 requirements requiring a very sophisticated CASE tool to assist in the traceability effort.

C. THE LOW END USER OF TRACEABILITY

We use the term "low end user" of traceability to refer to a member of the systems development life cycle who applies traceability in their systems development efforts, but only to a limited degree. The low end user uses traceability within their own domain in the systems development life cycle (i.e., the systems designer uses traceability only when designing the system), but does not take advantage of what traceability has to offer across the development process. This was the case in over half of the participating organizations. Within the typical organization, only three members of the project development team practiced requirements traceability, as discussed below, in the organization's systems development efforts: the project manager, the design engineer, and the test engineer.

1. Experience Level of the Low End User

The majority of the low end users had limited experience with traceability and many were developing a system using traceability methodology for the first time. The amount of experience the low end user had in the development of complex systems was ten years, while the amount of experience in using and practicing traceability was considerably less, usually between one and two years. This mismatch is due in part to the recent (1987) requirement to provide traceability in critical systems development for the Government. Adding to this limited exposure is the lack of guidelines detailing what is expected from practitioners of requirements traceability.

2. The Low End User's View of Traceability

The low end user views traceability as a transformation of system-level requirements to design requirements, producing a type A specification which becomes the technical requirements portion of the response to a Request For Proposal (RFP). Traceability should show that each requirement at least partially satisfies a system-level requirement. Also, traceability should highlight the verification process from the bottom of the specification tree (hardware/software component testing) up to the top (system verification). The less refined user defined traceability "as yet another requirement that must be fulfilled," basically feeling that traceability was simply tracing requirements to components in the design phase.

3. The Low End User's Basic Applications of Traceability

A model of the typical low end user's traceability efforts is provided in Figure 1. This model does not represent any one organization's traceability practice but is an abstraction of the practice observed among low end users. Also, names of objects and links presented here are abstractions of what is observed in practice. Many organizations simply use "trace to" links between various components which do not convey the semantics of the relationships. This section will discuss the information captured in the various links of the model and its uses. The basic applications that were evident among the low end users were requirement decomposition, allocation of requirements to system components, compliance verification, and change control.

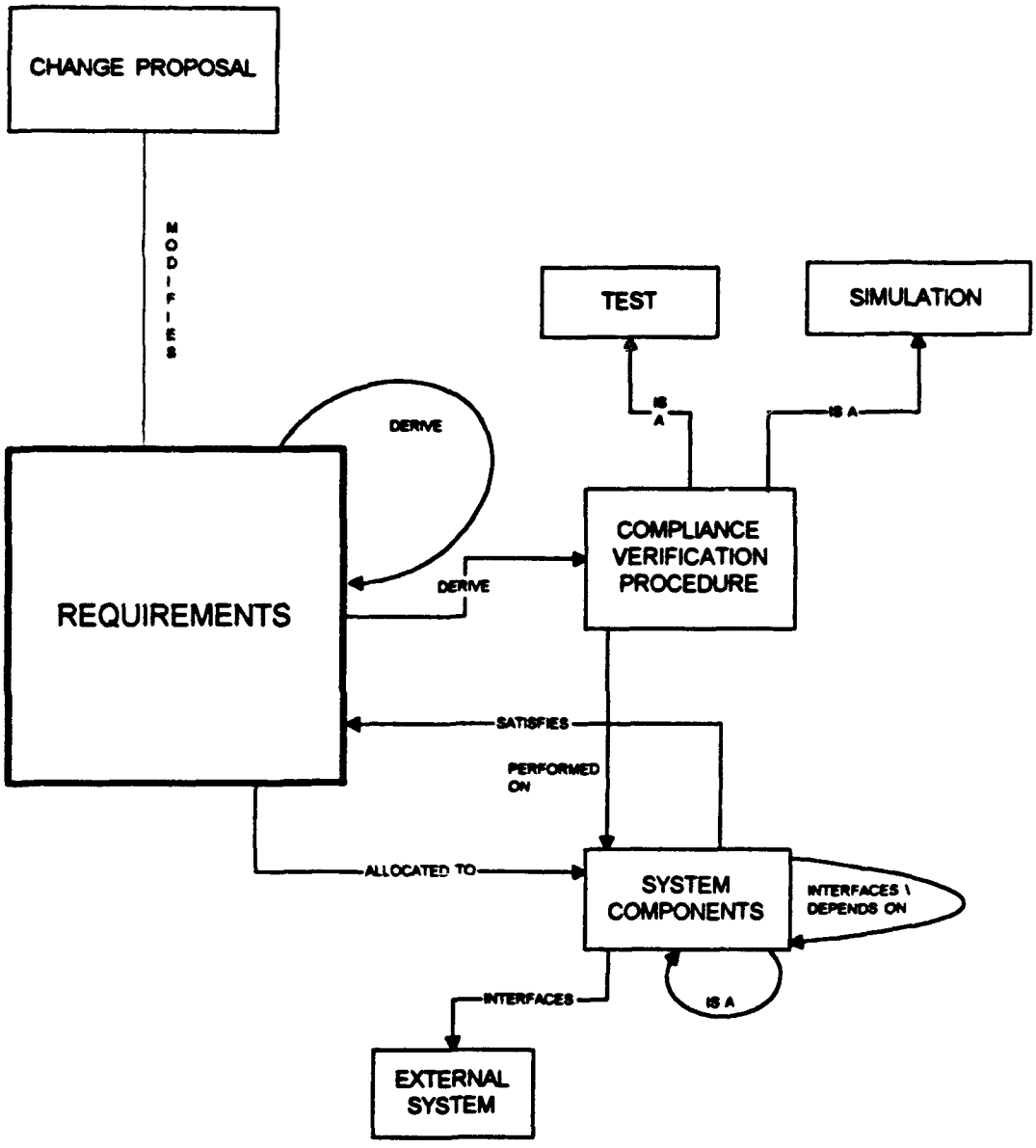

Figure 1. Low End Traceability Model

Typically, the design engineer viewed requirements traceability as providing a link from initial requirements to the actual components that satisfy those requirements. However, first, the system level requirements must be decomposed to a more refined level. During this recursive loop, seen in Figure 1 as the "derive" link on requirements, derived and lower level requirements are traced to system level requirements. Every lower level requirement has a parent, higher level requirement, from which it was derived. Typically, this information is captured in a relational data base, and used in the form of a traceability matrix.

After the requirements have been decomposed to the lowest level possible, components of the system design are traced to these requirements. This trace is usually accomplished by a simple two way mapping between requirements and system components. By capturing which components satisfy which requirements and which requirements are mapped to which components, the designer is able to verify that all requirements are addressed by the system.

In the testing phase of systems development, low end users use the requirements database, which contains the most current version of the system's validated requirements, to develop the system test plans, tying them to specific requirements and system components. Compliance verification procedures derived from the requirements, in the form of tests or simulations, are performed on the system components verifying that the component satisfies the requirements. Results of the tests are captured and used to verify that the system works and that it meets all of the requirements. If a change should occur

in the requirements then the traceability links could identify the system design components that would have to be modified and retested.

The low end user lacked source documentation in numerous aspects of traceability. In regards to requirements, information concerning requirement issues, how they are resolved, and the rationale for the decisions is not captured. Likewise, information is not captured concerning the system design, constraints, design issues, design decisions, and design rationale. The extent of the missing traceability efforts can be seen by comparing Figure 1, the model for the low end user, and Figure 2, an aggregate model for the high end user of requirements traceability.

D. THE HIGH END USER OF TRACEABILITY

The distinction between the approach of the high end user and the low end user in practicing traceability is drawn by the detail to which traceability is practiced. The typical high end user views requirements traceability as a means to force higher quality into system design and the end product while reducing time expended and life cycle costs.

The high end user of traceability has implemented traceability methodology across all areas of the systems development life cycle. Figure 2 provides a model of requirements traceability for the high end user. This model does not reflect any one organization's traceability practice, but is an aggregation of the advanced practices seen across the industry. A typical organization captured and used only a subset of the information presented here. As stated earlier, the names of objects and links presented here are abstractions of what is observed in practice. This section provides several examples of the

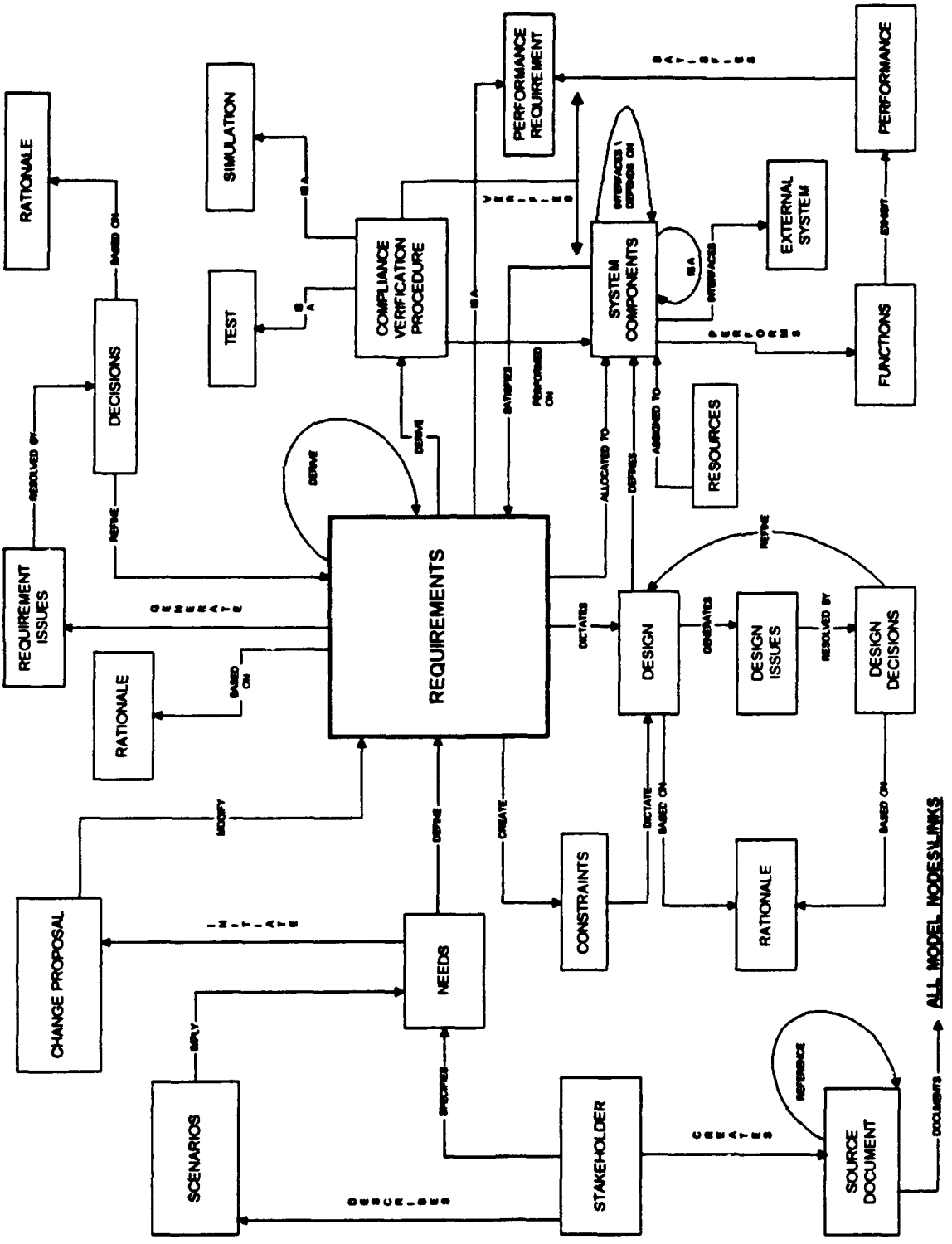

Figure 2. High End Traceability Model

innovative traceability practices of the high end user and discusses the information captured during their efforts.

1. Experience Level of the High End User

The amount of experience the high end user has in complex systems development was ten to 30 years, while the experience in using traceability was usually between five and 15 years. The high end user employs a traceability methodology with systems engineering practices that are very advanced. Also, the high end user utilizes a CASE tool that supports requirements traceability in their systems development practice.

2. The High End User's View of Traceability

The more sophisticated user viewed traceability as a means to increase the probability of producing a system that meets all of the customer's requirements and will be easy to maintain. Traceability is not simply the linking of requirements to components, but is also the linking of information that is vital for the purpose of understanding requirements, design, defining accountability, and supporting periodic customer and management reviews. This broader view of requirements traceability provides the high end user a methodology, not only spanning across one system's design life cycle, but also is carried to the design of other systems and throughout the organization's daily work routine.

3. The High End User's Applications of Traceability

The stakeholder depicted in the high end user model represents not only the various development personnel throughout the systems development life cycle (i.e.,

program manager, design engineer, etc.), it also includes the customer and the end user. A system's design begins with the customer specifying needs. These needs identify the customer's shortfalls in existing systems (i.e., an interactive intelligence database has outgrown the capabilities of the current system) usually in the form of a Mission Element Needs Statement (MENS) or Operational Requirements Document (ORD). In some instances, the customer builds operational scenarios, describing the desired system in a simulation model, that enables the user to validate the needs stated in the MENS and ORD. Other times, the customer's needs would be in the form of a change to an existing system and would be detailed in an Engineering Change Proposal.

The high end user begins using traceability during the Pre-Concept Analysis of a project (sometimes referred to as Pre-Proposal Stage), which normally begins with receiving a MENS. Rather than wait for a Request for Proposal (RFP) to be released, aggressive systems development organizations would seize this opportunity to begin their system development efforts. During this phase, the organization expends the majority of its time developing documentation detailing the proposed system, discussing with the customer their needs, and formulating a plan for project development.

The Pre-Concept and Development Department extracts the needs of the customer from the MENS, ORD, and operational scenarios (if available) and, using a CASE tool, perform their own behavior simulation to determine if developing the proposed system is feasible with the organization's technological expertise. System level requirements are then identified from the MENS and ORD. In this effort, the high end

user decomposes the system level requirements to a level at which a preliminary system design can be developed. Throughout this stage, the high end user captures a link between the customers needs and system level requirements that satisfy those needs and a link validating the preliminary system design with the customer's needs.

The next source document provided by the customer would be the Request for Proposal (RFP), developed by the end user's contracting department, which formally details the system requirements. The high end user compares these formal requirements with those requirements extracted from the MENS and ORD in their Pre-Concept Analysis. Any differences will result in requirements issues which are addressed with the customer for clarification. This process results in a design which not only satisfies the requirements as stated in the RFP, but also mirror the user's needs as originally stated in the MENS and ORD. The aggressive engineer regards the MENS as the original source document identifying customer needs, but legally, are only responsible for requirements stated in the RFP. By designing a system that addresses both aspects, the high end user eliminates confusion commonly found between end users and their contracting department.

Implementing traceability so thoroughly at such an early stage may seem costly, especially if the organization does not win the contract award, but there are several benefits that make these costs acceptable. The biggest benefit of doing this is that a requirements database is already in place if the contract is awarded to the organization. The user already has a baseline on which to proceed and if changes are made to

requirements, the transition is easy. Several organizations stated that they were not the low bidder on a contract, yet they were awarded the contract because of the "appearance" of expertise that were able to show due to the documentation resulting from their pre-concept traceability efforts.

Also, the documentation resulting from this early traceability effort is reused by organizations that produce a specific system, such as a satellite repeater, in future bid proposals. In many cases, numerous RFPs are evaluated by an organization over a short time frame with only minor changes to the system being developed. By implementing traceability in the Pre-Concept Analysis, these organizations already have a large part of their bid proposal completed.

As the requirements are decomposed, issues arise regarding how to break the requirement down or what the requirement really is. Capturing the decisions associated with these issues, as well as the rationale upon which the decision was made, is an important aspect of requirements traceability. This information, usually captured in form of the engineer's notebook, is linked to the requirements. If, later in the project development or in maintenance, a question arises concerning the requirement, the rationale and decision information can be easily traced providing the background for resolving the initial requirement issue. Of equal importance is capturing information regarding the derived requirements. Once a system level requirement is decomposed, information regarding the derived requirements will not be documented unless it is captured through requirements traceability. For example, a system level requirement may state that an

aircraft shall maintain a certain attitude in level flight based on a specific power setting. As this requirement is decomposed, derived requirements such as: power setting, airfoil setting, and environmental factors must be accounted for. These requirements must be traceable to the higher level, or parent requirement.

One of the main objectives of any contractor is to produce a Type "A" Specification that satisfies the RFP. When dealing with numerous RFPs, an organization needs to produce similar, yet different, Type "A" Specifications. To meet this objective one high end user developed an extension to their traceability CASE tool that enables them rapidly tailor their outputs to the meet the needs of each unique RFP. This added functionality also supported other aspects of project management. Rather than modify the tool for every type of report, they simply created a function in the tool that enabling the generation of any type report, with any type of information.

As the organization completes requirements analysis, focus shifts to the design phase of systems development. Some requirements are viewed as constraints, such as: use of existing hardware, budgetary limitations, and software language to be used, which directly impact the system design. The remaining requirements are used by the design engineer to develop a system architecture that meets the customers needs. As the design develops, issues arise that need clarification before design efforts can continue. The high end user captures design rationale information, the why of design decisions, throughout the design phase. This is perhaps the most critical aspect of capturing information through traceability, as it is used both in the system design, and also in system maintenance.

Design rationale, as viewed by the high end user, is that information that will enable, even the inexperienced engineer, to adequately make and understand design decisions based on the history of the current design of the system. The high end user captures the rationale used in the systems development, including why the design is structured the way it is, and any decisions that effect this design. Design rationale may take on the form of trade studies, meeting minutes, proven design practices, and the engineer's personal design decisions.

Capturing information on the design history of a project is not an easy task. It is very time consuming (upwards of 50% of an engineer's time is spent on documenting traceability) resulting in large overhead. However, the high end user acknowledged and accepted this cost, realizing that the organizational benefits would outweigh the costs in the long run. Those benefits could be realized in future system development (reduced development time and effort) or in modifying existing systems. This "corporate memory," provides the organization with a knowledge tool that could be used for a number of applications, such as change management and development of similar projects, whereas without traceability, the information could be lost if the engineer quits the job or is transferred to another project. It must be noted that much of the rationale collected is in the form of free text which is not well structured. The quality of the information captured is directly related to the effort expended by the engineers.

As the system design is refined, system components are defined (i.e., computer software component, computer software unit, hardware, humanware, etc.) by the actual

design. Requirements are then allocated to the system components. Here, the high end user goes into more detail in their traceability efforts by identifying functions displayed by specific system components, such as computer software unit 3.3.3 controls aircraft aileron movement. The high end user then captures the performance that these functions exhibit to show that performance requirements are satisfied.

The compliance verification procedures illustrated in Figure 2 perform several functions throughout the system development life cycle, such as simulation and testing. A simulation could be a behavior model developed from system components verifying compliance with requirements, for example, a simulation that would test compliance for the previously discussed aileron movement component could be an actual flight simulator. Compliance verification test plans are derived from requirements and provide a means to ascertain compliance and verification of requirements allocated to system components. Information captured in compliance verification procedures would include the type of procedure, the component tested, the requirement allocated to that component, and the results of the procedure.

The customer's needs may also take on the form of an Engineering Change Proposal. In change control management, the high end user uses the requirements information to trace the change through the system design and to the components, thus identifying the impact of the change. By previously capturing information regarding derived requirements and allocating all requirements to system components, determining the impact of a change proposal, be it direct or through residual changes, is a manageable

problem. This not only identifies what hardware and software may need to be changed, it also identifies the impact on the compliance verification procedures, identifying which test procedures or simulations need to be modified.

E. THE IDEAL TRACEABILITY CASE TOOL

Both low end and high end users strongly agreed that a CASE tool supporting requirements traceability must be used in order for an organization's traceability efforts to be successful. Users were very specific concerning the characteristics desired in a traceability CASE tool. The desired characteristics were grouped under three topics, functional, quality, and operational applications. Functional characteristics refers to the ability of the tool to perform specific traceability tasks. Quality criteria refers to issues such as ease of use, robustness, and power. Operational criteria refers to cost, multi-user environment, graphics capability, and hardware platform.

1. Functional Characteristics

The following capabilities were identified as being essential for a requirements traceability CASE tool to be usable:

- interface with other tools.
- input ASCII text files.
- produce and generate ad hoc reports required by the customer.
- automatically check for requirements not addressed, satisfied, or verified.
- create and modify analysis of checks.
- establish and show bidirectionality at all levels.

- group requirements, based on keywords or attributes.
- ability to create attributes.
- generate various reports for requirements, design, and testing analysis.
- support configuration management.
- provide management reports, such as list of requirements in a document, number of requirements in a system specification, and the ability to graph statistics.

2. Quality Characteristics

The quality of a CASE tool was one of the biggest issues of the users, and included the following:

- friendly user interface.
- minimal data entry (no duplicate entry).
- windowing capability for reviewing requirements.
- on-line help.
- easy definition and creation of reports.
- performance considerations.
- data integrity (no loss of data in the event of crash).
- large database with "speed" of performance.
- short query response time.
- good vendor support generally means quality.

3. Operational Characteristics

Operational issues addressed included:

- multi-user support.
- provide a methodology with vendor support.
- good reference manuals.
- updated versions easily obtained and at minimal cost.
- minimal number of people required to maintain the tool.
- within budget.

F. CONCLUSIONS

There is still uncertainty as to what traceability can truly provide organizations. This became evident by witnessing the varying degrees to which traceability is practiced throughout industry. Numerous organizations practice traceability only to the level dictated by the Statement of Work, using a homemade tool or simple spreadsheet. On the other hand, there are several aggressive organizations that carry traceability fully across their entire systems development life cycle, making it a part of their normal daily routine.

It is evident that using a requirements traceability CASE tool greatly improves an organization's chances of properly implementing traceability and taking advantage of the numerous benefits traceability has to offer. Numerous organizations have purchased or developed a CASE tool to assist in the traceability effort. However, since most of the users had little experience with the CASE tools, concern was expressed with the amount of training required to become proficient in the tool's use. The majority of the organizations studied had either hired outside assistance with the tool they had chosen or assigned someone from within to provide the data entry into the traceability tool. The

result was that usually only one or two people had adequate knowledge as to the full capability of the tool which curtailed the degree to which traceability was implemented within the organization. The information captured was limited to what was required to meet the Government standards or as detailed in the Statement of Work. As organizations become more familiar with what CASE tools have to offer, their traceability efforts will increase dramatically.

IV. A-10 ADA DEVELOPMENT AND INSERTION PROJECT

A. INTRODUCTION

The case study focused on the use of requirements traceability by the Weapon System Technology Support Branch of McClellan Air Force Base's Science and Engineering Division (WST-SED) in their day to day operations as well as on the A-10 Ada Development and Insertion Project (A-10 ADIP). The main focus of the study was to determine the impact of traceability on the A-10 ADIP and the actual and perceived benefits of traceability as viewed by the Weapon System Technical Support Branch. The information was gathered during on-site interviews with Weapon System Technology Support Branch management personnel and the A-10 ADIP personnel. One-on-one interviews were conducted with the organization's upper management, the project manager, system designers, and test/audit personnel. The scope of the interviews was to determine how the various stakeholders view traceability, to what extent each uses traceability in their daily job tasking, and what are the perceived benefits of using traceability.

The Weapon System Technology Support Branch was chosen as the subject of this case study after an extensive search for an organization that uses traceability, not only to satisfy the project sponsor's requirements, but also includes a thorough traceability practice in throughout management of their systems development efforts. Several

organizations considered as candidates were reviewed, but were not pursued as they were simply producing traceability information to comply with DoD STD 2167A requirements, but not truly using traceability in their day to day operations. In contrast, the Weapon System Technology Support Branch was found to firmly believe in the use of traceability to the extent that it is practiced by all employees in their daily work efforts.

B. WEAPON SYSTEM TECHNOLOGY SUPPORT BRANCH

The mission of the Weapon System Technology Support Branch at McClellan Air Force Base is to provide embedded computer systems (ECS) and software support for projects generated by system program directors, other DoD Agencies, and private industry related to advanced ECS technologies. By March 1994, the organization foresees achieving a Level III status under the Software Engineering Institute's (SEI) Software Maturity Model. Branch responsibilities include: Technology and Ada insertion projects, Chairing the center's Ada Technology Working Group, and ECS software policy. The Branch also supports or is preparing to support advanced software workloads such as Ada 9X, Object Oriented Design, Diagnostics, and Reuse.

The Weapon System Technology Support Branch consists of a Branch Chief, two Section Chiefs, System Engineers and an employee contracted out from the Science Applications International Corporation (SAIC) who provides support for the traceability CASE tool used by the Branch. The CASE tool specialist was hired to help the staff become proficient with the CASE tool.

The Branch acts as an independent contractor when taking on projects for other divisions and branches internal to Sacramento Air Logistics Center (SM-ALC) as well as external organizations. A Statement of Work is developed detailing the specification requirements and deliverables as well as budget items and a work schedule. The Branch must seek out and bid for projects similar to what an independent contractor must do to economically survive. Funding for labor and materials (hardware and software) is detailed in the Statement of Work.

The Weapon System Technology Support Branch is starting to operate on a zero-based budget system, relying on incoming funding from their various projects to operate. Funding for personnel, hardware, and software is directly tied to specific projects. If the Branch does not perform satisfactorily, projects could be terminated which could result in a loss of funds and economic disaster.

C. A-10 ADA DEVELOPMENT AND INSERTION PROJECT

The A-10 Ada Development and Insertion Project (A-10 ADIP) was contracted out to the WST-SED. The initial Statement of Work (SOW) required that the present Operational Flight Program (OFP) for the A-10 attack jet be redesigned from Jovial programming language to Ada, as well as provide enhancements to the existing system. Included as a requirement in the SOW was for the Branch to provide traceability from the Product Specification to the actual Ada coded modules. The A-10 ADIP contains approximately 75,000 lines of code and over 3,000 requirements.

1. A-10 ADIP Statement of Work

The decision to modify the Operational Flight Program in the A-10 aircraft was prompted by the limitations of the original program, written in Jovial, to provide the desired system upgrades while still maintaining acceptable system throughput. The original processor was operating near its capacity, thus, incorporating the desired upgrades required a new processor. The Statement of Work detailed five primary requirements:

a. A complete rewrite of the OFP from Jovial to Ada

The complete rewrite of the OFP from Jovial programming language to Ada programming language was intended to provide an identical base OFP written in Ada to which upgrades will be made. This was a result of the Department of Defense directive requiring Ada to be used for all significant system upgrades.

b. Functional equivalency to the V.40 Jovial based OFP

The new system was required to be functionally equivalent to the existing Jovial based OFP. Functionality must be directly mapped from the existing system to the new system so that the pilots will not require total retraining. It is deemed mandatory that when the pilot pushes a button in the plane's cockpit, the new system will yield the same results as the old system. However, retraining will be required when additional functionalities are incorporated into the system.

c. A complete redesign of the OFP using Object Oriented Design philosophy

The new system must be redesigned using an Object-Oriented Design philosophy which is highly suitable with the use of Ada.

d. Incorporate a time slicing scheme with tasks optimally distributed among the slices

A time slicing performance scheme must be incorporated in the new design. This requirement is a performance related requirement and a detailed discussion of this is beyond the scope of this study.

e. Implement total "Dual Redundancy" for both processors

Dual redundancy is required between the new processor and the existing processor. This requirement is also a performance related requirement and is not elaborated further here.

2. Requirement for Use of Traceability

Traceability of specifications was required 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 the Interface Requirement Document and Software Requirement Document, Computer Software Component, and Computer Software Unit. A CASE tool was procured to help ensure that traceability was properly implemented throughout all phases of the project.

D. USE OF REQUIREMENTS TRACEABILITY BY THE ORGANIZATION

The Weapon System Technology Support Branch has embraced requirements traceability methodology as vital in their day to day operations as well on specific projects. The organization's view of using traceability in their daily operations is best summed up by the Branch Chief, who firmly stated that "Traceability is a must to our Branch's survival. We must use it in our daily operations and take full advantage of the benefits that traceability provides in developing systems." This strong support for the use of traceability was enlightening.

The Branch views traceability as a methodology that will ease the task of life-cycle maintenance. By providing traceability from the Product Specification requirements to the Ada code modules resulting in a traceability matrix, the maintainers will have a tool that will help quickly pinpoint the location of problems in the source code, significantly reducing the amount of time and effort commonly experienced in trouble shooting such large, complex systems. The Branch even foresees the actual users of the system, the A-10 pilots, participating in the troubleshooting of problems found during flight.

All of the personnel working on the A-10 ADIP use requirements traceability in conducting their daily tasks. Many of the uses and benefits of using traceability that the workers perceived were similar to those identified by the authors during their literature review. The following sections describe the requirements traceability employed by the WST-SED staff, as viewed by the research team, and detail how various Branch personnel view and use requirements traceability in their daily work routine.

1. Traceability Model Employed by WST-SED

Though WST-SED has not developed a formal model of 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 3 provides such an information model. This model identifies various types of traceability information used in areas such as requirements rationale capture, design rationale capture, allocation of requirements to system components, and allocation of resources to various system components. A detailed discussion of these aspects of traceability is provided in chapter three.

The segment of the traceability model that addresses verification of compliance of the system to the requirements (including testing) is shown in a dotted box indicating that it is not yet practiced by WST-SED. However, it should be noted that ADIP has not reached the stage of "testing" yet. Based on our discussion with management, the model represents the traceability mechanism WST-SED 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 address, and so on. In this effort, therefore, it is near impossible to "trace back from requirements" to their sources. WST-SED, however, maintains rationale behind requirements whenever feasible.

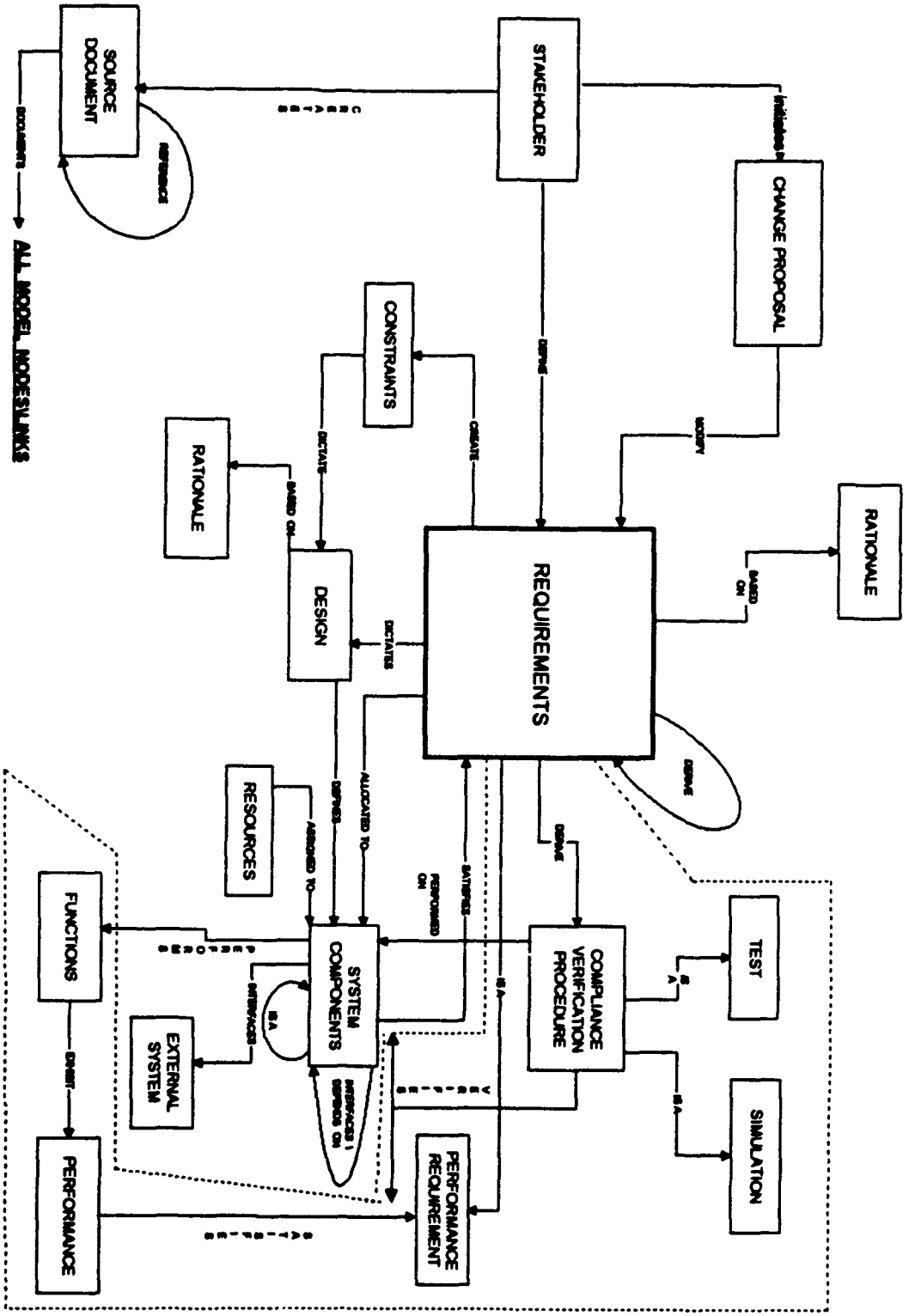

Figure 3. A-10 ADIP Traceability Model

Throughout the branch, stakeholders create source documents supporting all of the nodes throughout the model. Such source documents include design rationale and requirements rationale through the engineer's notebook and requirement tracing through the traceability matrix. Though WST-SED requires the capture of design and requirements rationale, this information is not captured in any pre-defined format. How the various issues get resolved is entered in free form through the engineer's notebook, which could result in missing vital information and nonstandard forms.

The model reflects how requirements are defined by the stakeholder or modified by change proposal requests. Furthermore, requirements are an iterative node in that upper level requirements derive lower level requirements through several layers of refinement. Once the requirements are refined to the lowest possible level they are used to create system constraints and dictate the system design. Also, the requirements are allocated to the system components providing traceability throughout the system design.

2. Upper Management

The upper management of the Weapon System Technology Support Branch viewed the use of requirements traceability as a must for survival. The branch is operating under a cost reimbursable system within the command (very similar to the Department of Defense's proposed unit costing system). If the Branch does work for another division within McClellan, a Statement of Work is developed with all efforts being costed out and charged to the other division.

Management summed up their use of traceability stating:

> Requirements traceability is not an option, it is essential to keep the customer happy. Requirements traceability is needed to survive in today's acquisition process. We must stay on the leading edge in systems development, especially in today's times where work means jobs. How are we to know when the job is done, if not for requirements traceability? 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 possibility of missing a requirement or missing a derived requirement in the process of developing large, complex systems. In the case of the A-10 ADIP, missing one requirement could be catastrophic to the pilots flying the aircraft.

Management uses traceability in evaluating and accepting potential projects. The Branch first identifies the requirements for a proposed system from the customer's project management plan. Once this is done, the requirements are referred back to the customer to ensure that the interpretation by the Branch is accurate. This initial step of traceability provides management with a complete list of validated requirements. Management uses the available CASE tool to help estimate 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.

Management also uses traceability as a work management tool in the daily operation of the office. The traceability matrix provides the manager an automated 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. Figure 4 is a sample traceability CASE tool output used by

TITLE: WPNS Status Matrix 09/09/92 (page 1 of 3)

The latest completion date listed on this matrix is 9/23

CSC/CSU Name	ASG	FLOW DIAGRAM	pre-decl spec	pre-decl body	post-decl spec	post-decl body	DDE
WPNS	DONE	N/A	none 2135	N/A	N/A	N/A	N/A
.BMB	DONE	N/A	9/23	N/A	none 1264	9/23	DONE
..CALCULATE_BALLISTICS	DONE	9/22	N/A	9/22	N/A	N/A	N/A
..CALCULATE_CCIP	9/23	9/23	N/A	9/23	N/A	N/A	N/A
..CALCULATE_MIN_RANGE	9/22	9/22	N/A	9/22	N/A	N/A	N/A
..CALCULATE_PBIL	DONE	9/11	N/A	9/11	N/A	N/A	N/A
..EXPANDED_MODE	DONE	N/A	9/22	N/A	9/22	9/22	9/22
...CALCULATE_COMPRESSION	DONE	9/11	N/A	9/11	N/A	N/A	N/A
...CALCULATE_TDC	DONE	9/11	N/A	9/11	N/A	N/A	N/A
...INIT	DONE	9/11	N/A	9/11	N/A	N/A	N/A
..INIT_CCIP	9/22	9/22	N/A	9/22	N/A	N/A	N/A
..INIT_MIN_RANGE	DONE	9/21	N/A	9/21	N/A	N/A	N/A
..INIT_PBIL	DONE	9/21	N/A	9/21	N/A	N/A	N/A
..INTEGRATE	9/21	9/21	N/A	9/21	N/A	N/A	N/A
..PERFORM_RUNG_KUTA	DONE	9/21	N/A	9/21	N/A	N/A	N/A
.GUN	DONE	N/A	9/9	N/A	9/9	9/9	9/9
..CALCULATE_AMIL	9/18	9/18	N/A	9/18	N/A	N/A	N/A
..CALCULATE_CCIP	9/18	9/18	N/A	9/18	N/A	N/A	N/A
..CALCULATE_FUNNEL_DELTAS	9/18	9/18	N/A	9/18	N/A	N/A	N/A
..CALCULATE_MRGS_DELTAS	9/18	9/18	N/A	9/18	N/A	N/A	N/A

Figure 4. CSU Progress Report

management, providing a quick-look report on the status of a CSU's development. The matrix shown provides management with status of the Weapons Module of the system being developed. The columns of the matrix depicts the CSC/CSU, while across the top the various project tasks are detailed. Completion status, or projected completion date for the various tasks are detailed in the matrix.

Traceability also provides management a tool for tracking and projecting budget information. Figure 5 is an output from the traceability CASE tool detailing manpower expended, for a specific work period, on various stages of the system's development. By capturing this information, management has an asset that will assist in change management, as well as projecting future workloads. In change management, the hours expended would be used to project the estimated time it would take to effect a change on that specific section of code, thus providing management a gauge to estimate the cost of implementing a change. Data captured in this matrix can be used to determine workloads of similar modules. . The matrix shown provides management with hours expended in developing the Weapons Module of the A-10 ADIP. The left hand column of the matrix depicts the CSC/CSU, while across the top the various project tasks are detailed. Hours expended in completing the various tasks are detailed in the matrix.

3. Project Manager

Throughout the system development life cycle, the project manager uses requirements traceability for more than simply producing links between requirements

15:04:47 16 Aug 93 Note 9406 page 1

TITLE:
WPNS Status Matrix 27 Apr 93

BODY:

CSC/CSU Name	ASG	Flow Diag	pre-decl spec	pre-decl body	post-decl spec	post-decl body	ECE
WPNS			1 h				
.BMB			1 h				
..CALCULATE_CCIP	0.5h						
..CALCULATE_MIN_RANGE	0.5h						
..EXPANDED_MODE	0.5h						
..INIT_CCIP	0.5h						
..INTEGRATE	0.5h						
..PERFORM_RUNG_KUTA	0.5h						
.GUN			1 h				
..CALCULATE_AMIL	0.5h						
..CALCULATE_CCIP	0.5h						
..CALCULATE_FUNNEL_DELTAS	0.5h						
..CALCULATE_MRGS_DELTAS	0.5h						
..INIT_MRGS_DELTAS	0.5h						
..INIT_MRGS_INTEGRATION	0.5h						
..INTEGRATE_MRGS_DELTAS	0.5h						
.LIBRARY			1 h				
.MAV			1.5h				
..CALCULATE_DATA	1 h	2 h	1 h	1 h			
..INIT	0.5h						
.MGR			1 h				
..INITIALIZE_IN_REAL_TIME	0.5h		2 h				
..SCHEDULE_PRIORITY_BACKGROUND_UNITS		1 h		1 h			
.SIM			1 h				
..CALCULATE_POSITION	0.5h						
..INTEGRATE	0.5h						
.TBL			1 h				
..DETERMINE_MIN_TIME	0.5h						
..DETERMINE_WPN_CHARACTERISTICS	0.5h						

Figure 5. CSU Time Card

requirements and source code. The project manager believes that proper use of traceability proves his worth as a manager by providing him a means of showing he is in full control of the project. Requirements are first captured from the product specification and linked to the Software Requirements Specification (SRS) and Interface Requirements Specification (IRS) documents. The system is then further refined, being broken down into more descriptive modules, until ultimately, the CSU level is reached. Requirements are linked from the initial SRS or IRS all the way to the CSU to ensure reliability of the system.

The project manager uses the Relational Information Data Base System in the available CASE tool to track the requirements from the SRS and IRS to the CSU. The project manager traces not only the initial system requirements, but also must identify, document, and trace derived requirements as the system is more defined. By using requirements traceability in this fashion, the project manager is able to prove to 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 how upper management did, 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. Figure 6 is a GANTT chart generated from the traceability CASE tool that is used by the project manager in

TITLE: Top-Level Gantt Chart - 2/26/93

OPR: David Bertioux

	Mar					Apr				May					Jun	
	1	8	15	22	29	5	12	19	26	3	10	17	24	31	7	14
Status Reporting & Meetings	4	T	4	4	4	4	4	4	4	4	4	4	4	4	4	4
EXEC Design	20	R	10													
WPNS Design	6	A	16	26	12											
Transition Plan		I		8												
Documentation Review		N		AS	R	E	Q	U	I	R	E	D				
CASE Tools	3	I N	3	3	3	4	4	4	3	3	3	3	3	3	3	3
ATIC	3	G	3	3	3	4	4	4	3	3	3	3	3	3	3	3

Figure 6. Quarterly GANTT Chart

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. The chart is completed using typical GANTT chart format.

Also, traceability provides the project manager with an early warning means of detecting project delays. By tracing requirements to the CSU level, derived requirements become readily apparent and are entered into the system. The number of derived requirements may be such that an increase in time, and resources may be required to keep the project on track.

Once the project reaches the testing phase, the project manager uses traceability to verify to himself and to the customer that the system meets the stated requirements and is complete. By using traceability to write acceptance test plans for every validated requirement, including derived requirements, the project manager is able to prove to the customer that the system is complete. Though A-10 ADIP has not yet reached this stage, the project manager has stated that the traceability CASE tool in use will greatly assist his staff in ensuring that the acceptance test plan tests all requirements.

Upon completion of the A-10 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.

4. System Designer/Engineer

The most significant use of traceability throughout this project was observed through the system engineer's "Engineer's Notebook." Figure 7 is a sample page from an 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 concerning why the system was designed as it was. This information could prove invaluable throughout life cycle maintenance and on the development of similar systems.

Although the project is not yet to 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.

15:00:31 16 Aug 93 Note 9010 page 1

TITLE:
Lessons learned Jun 92

BODY:
ADA DESIGN
Variable_A is declared in Procedure_A
Variable_B is declared in Procedure_B
Procedure_A calls Procedure_B
Variable_A is not visible to Procedure_B
Variable_B is not visible to Procedure_A
Any variable which needs to be visible between procedures will have to be passed as a parameter or declared in a package.

Sam Dugan understood the importance of set/use info. He spent a lot of time extracting this info from the GE SDDD. Now that I am trying to design packages and procedures, this info determines where variables are defined.

There are a lot of Jovial variables which are related but are declared as separate variables. For example,
CCF'VIPBFX IMPACT POINT VELOCITY X-AXIS
CCF'VIPBFY IMPACT POINT VELOCITY Y-AXIS
These variables are the x and y components of a velocity vector.
In Ada, I could combine these variables into 1 variable. I would declare a 1 dimensional array with 2 elements: Impact_Point_Velocity: array (1..2) of UTIL.TYPES.Feet_Per_Second;
Then, Impact_Point_Velocity(1) would represent CCF'VIPBFX and Impact_Point_Velocity(2) would represent CCF'VIPBFY
But, in an array, the elements have to be of the same type. In this example, I declared the type to be UTIL.TYPES.Feet_Per_Second.
In the GE Jovial design, the variables CCF'VIPBFX and CCF'VIPBFY have the same units but the range of possible values is different for each variable. These range constraints are important because they determine if the value for a variable is valid.
Therefore, I can't use the array declaration because the elements have different range constraints.
However, I could declare a record because record components can be of different types. For example:

```
type Impact_Point_Velocity_Type is
   record
      X_Component: UTIL.TYPES.Feet_per_Second range 0.0..1_000.0;
      Y_Component: UTIL.TYPES.Feet_per_Second range -200.0..200.0;
   end record;
```

PROJECT MANAMGEMENT
The more places a person is required to update information, the more chances there are for error. When I started designing packages I had a note which listed all the jovial names and the corresponding Ada names for the variables in that package. Then I had a separate note for the Ada variable declarations and type definitions for that package. My reasoning was that the note that had just the jovial and ada names would be a concise way to see all the package variables and would be easy to visually search for a particular variable. The other note would have all the Ada syntax. But the variables in a package would change constantly. This required updating 2 notes. I soon saw that I was wasting time and making mistakes. Now I have 1 note with the Ada type definitions and variable declarations with the corresponding jovial names right justified. The extra time it takes to find a specific variable is offset by the time saved in maintaining only 1 note.

Figure 7. Engineer's Notebook

Additionally, a software designer was interviewed to determine the extent that he used traceability on the A-10 ADIP. The software designer's task was to ensure that A-10 weapons software written in Jovial was successfully and completely rewritten in Ada. There were no new requirements in this aspect of the project.

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 CASE tool, the original system functional requirements were mapped to the ASGs, providing a level of confidence that the Ada system design was complete, providing the "fit and function" of the translated code.

5. Tester/Auditor

The A-10 ADIP has not yet reached the test phase, however, the authors interviewed the system testers concerning their planned use of requirements traceability in developing and conducting system testing. The system testers plan on using traceability in writing the acceptance test plan. Making use of the CASE tool, the testers will verify that the Acceptance Test Plan tests all of the system requirements, thus ensuring completeness. This will allow validation that the system has been completely tested and that it operates as it is designed to, while meeting all of the customer's requirements.

E. USE OF CASE TOOLS BY THE ORGANIZATION

The Weapon System Technology Support Branch had recently purchased a CASE tool to assist with their traceability efforts. At the time of the case study, Branch 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 Branch personnel on its use. These were viewed as sunk costs or necessary evils that would prove beneficial "down the road" but not on the initial project.

Branch personnel were using the tool to assist in ensuring that all of the initial requirements from original OFP were addressed and allocated to components of the system. By allocating the requirements in this manner, they were able to isolate requirements within a specific component, yet still track dependencies across the system. Additionally, the CASE tool was used to assist in the formulation of traceability documentation required by DoD STD 2167A.

F. SUMMARY/LESSONS LEARNED

The Weapon System Technology Support Branch has an excellent grasp on how traceability can be used to enhance the systems development process. It was encouraging to see the Branch think of the use of requirements traceability as a must in its daily operations. From the upper level management (also pictured as the customer

interface personnel) to the system tester, 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. Their productivity will increase over the long run due to their dedicated use of traceability, especially in aspects such as the historical benefits, and as they become more familiar with the CASE tool. As the organization becomes more familiar with the CASE tool being used to assist in its traceability efforts, benefits such as reduced development and maintenance time will be experienced, which directly relates to reduced costs.

1. Definition of Traceability

The first step in implementing a traceability scheme is the definition of an information model that details the content of the traceability information that the organization intends to capture and use. Such a model is needed to facilitate ease of capture in desired detail and ease of reuse in a standard fashion. In the ADIP, the design 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. An information model that standardizes the form and content is required so that the data captured will be consistent and useful. ADIP has addressed the issue by defining templates for several reports that are produced as a part of the traceability scheme.

The absence of a comprehensive model of traceability is a common problem throughout industry. DoD-STD-2167A, the document that has pushed the practice of

requirements traceability throughout much of industry, does not provide an informative model of traceability. At WST-SED information on areas such as budgeting, design rationale, and project management was captured and used, but were not perceived by some as part of the a traceability scheme. This reinforces the need for development of a comprehensive traceability model for organizational use throughout the systems development industry.

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 initial 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 Branch to requirements traceability and the use of a CASE tool in systems development, the A-10 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-SED established a good model for implementing requirements traceability into an organization's management philosophy. WST-SED

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.

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 team completed trade study reviews, reviewed operators manuals, and examined code line by line in an effort to understand the original requirements.

Ultimately, the staff had to back-hire engineers from the initial Jovial project, at an estimated cost of about $150,000, to assist in determining the detailed requirements from Jovial code. Had the initial system been developed using a traceability methodology, 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, due to the Jovial system not being developed using traceability.

With the present shrinking budget legacy systems are becoming more and more common throughout the U. S. Government, Also, with the mandate to use Ada

in any significant system rewrite, the problems experienced by the WST-SED (having to "uncover" lower level requirements and missing detailed design rationale) will become common. Developing systems that provide comprehensive requirements traceability will greatly enhance the efforts of re-engineering those systems at a later date.

4. Traceability for Hardware Upgrades

The A-10 system is already under consideration for hardware upgrade. The re-engineering traceability effort has been capturing detailed software interface requirements that the current system is subject to. This effort is expected to provide immediate payoff during the hardware upgrade.

Like A-10 ADIP, numerous other legacy systems within the U. S. Government and Department of Defense were not originally developed under a traceability methodology. The ADIP experience suggests that it may be advantageous to even *re-engineer* some of the requirements traceability information during a system upgrade. With rapid increases in the hardware technologies, frequent hardware upgrades are becoming common in Defense Systems. 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.

5. CASE Tool Compatibility

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 A-10 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 very large scale systems development.

6. Functionality's of CASE Tools

The functionalities needed to support various stakeholders should be carefully examined in choosing a CASE tool for traceability. Also, difficulties arise when first using a traceability CASE tool in project development. In the A-10 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 levels to higher levels and vice versa is 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-SED 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.

7. "Political" Problem of Ownership

With the detailed amount of documentation associated with requirements traceability, concerns arise over a "political" issue: this information may be used for performance evaluations. Management at WST-SED handles this by instituting a "Team Responsibility" philosophy throughout the shop. As an engineer completed a portion of the project, the other engineers would review the work and documentation as a team. The result was that all products were considered team products and no individual had to worry about "being hung" for a mistake in design rationale.

8. High Costs of Traceability

Planning for the increased documentation required of requirements traceability, the initial budget for the A-10 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, throughout development. It is anticipated that these costs will be more than recovered throughout the project's life cycle due to development of a higher quality product and reduced maintenance costs.

The overhead associated with training the various members of the organization in use of the CASE tool was both time consuming and expensive. However, WST-SED 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.

9. Traceability in Process Improvement

Management viewed implementing traceability into the organization's systems development methodology as "an important concept of improving the process of systems engineering activity and overall project quality." Additionally, management viewed requirements traceability as an important component in increasing their SEI Process Maturity level 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 SEI Process Maturity rating, implementing a traceability methodology has provided WST-SED a critical review of the Systems Engineering process and an opportunity to modify those processes that resulted in an improved SEI Process Maturity level.

V. CONCLUSIONS AND RECOMMENDATIONS

A. NEED FOR A MODEL

As detailed in our case study, in order to fully realize the benefits that requirements traceability has to offer, traceability *must* be practiced throughout the entire systems development life cycle. However, before traceability can be properly incorporated into an organization's corporate vision, the organization needs to fully understand the information that needs to be captured through their traceability efforts. In the absence of guidelines regarding this, chapter three provides a model that could be used by organizations in their requirements traceability efforts.

B. NEED FOR A METHODOLOGY

In conjunction with a model, which details *what* needs to be captured, a traceability methodology must be incorporated into an organization's way of doing business, explaining *how* the information should be captured and *what should be done with it* once it is captured. This methodology needs to support all of the stakeholders throughout the systems development life cycle.

C. NEED FOR INCENTIVES

As the end-user of systems being developed, the Government should *require* a requirements traceability methodology that provides detailed documentation as part of the final product. This would ease life cycle maintenance of systems, especially when the

follow-on contract for updating the system is awarded to someone other than the initial designer. Among other documents, the engineer's notebook should be included, which should contain the rationale used throughout the system's development. However, this introduces the facet of intellectual property issues, who owns the rationale used in developing the system. A comprehensive scheme needs to be developed addressing this as well as detailing specific guidelines for systems development organizations to follow.

D. NEED FOR CASE TOOL USE

In order to fully exploit the benefits of requirements traceability, a traceability CASE tool should be employed throughout the system's development. System development organizations will experience a substantial front-end cost associated with procuring and becoming proficient with a CASE tool, but the long-term benefits of this approach to systems development far outweigh these costs. Organizations should identify a loss leader project, as detailed in chapter four, for the initial project to be developed using requirements traceability and a CASE tool.

E. NEED FOR A LIVE (EVOLVING) DOCUMENT

With the Department of Defense dedicating four percent of a system's development costs to requirements traceability, the product of these efforts must be of benefit to the Government. One such way to encourage this is to treat requirements traceability documentation as a living or evolving entity. As the system development efforts progress through the systems development life cycle, the requirements traceability documentation should be constantly evolving to reflect the current system status. The information

captured and maintained in the traceability documentation needs to be historical, from "cradle to grave."

F. SUGGESTIONS FOR AREA OF FUTURE RESEARCH

Follow-on research to this thesis should include:

- Extend the size of the systems development organization sample.
- Conduct a sample focusing of industry category (i.e., private industry vs. government industry, large vs. small organizations, private industry vs. U. S. Government systems development organizations).
- Focus on organization's determinants to determine success factors for implementing requirements traceability.
- Revisit the WST-SED to continue analysis of the A-10 ADIP.

APPENDIX A

WHO PRACTICES TRACEABILITY?

Numerous stakeholders are involved systems development throughout the Systems Development Life Cycle. Each of these stakeholders views and uses traceability differently. In this Appendix, the authors will attempt a first cut at who uses traceability at the various stages throughout the Systems Development Life Cycle. Each of these stakeholders will be examined addressing their use of requirements traceability in the systems development life cycle.

A. PROJECT SPONSOR (CUSTOMER)

The project sponsor is the stakeholder that provides the funding for the system being developed. As such, he is most concerned with cost overruns and the finished product. By taking advantage of compliance with requirements that traceability provides the project sponsor is afforded a mechanism to ensure unnecessary features are eliminated and that required items are properly addressed in the system. Also, the benefit of completeness provided by traceability provides the customer with the satisfaction that all requirements are functionally implemented in the system. This will help the project sponsor minimize cost overruns, keep him aware of schedule slippage and prevent delivery of an incomplete system.

B. PROJECT MANAGER

The project manager is the person responsible for the overall project, including milestone management, from beginning to end, ensuring complete and timely project completion. Traceability provides the project manager with a means to view the entire system design at any stage in development and to monitor the progress of program development.

C. SYSTEM DESIGNER

The system designer needs to trace requirements from the original requirements documentation to design objects or source code, and from the source code back to the original requirements documentation. Properly implemented traceability (successfully mapping all requirements from the requirements documentation to the system design) allows the system designer to quickly verify that requirements will be or are met by the system design, thus providing a tracking mechanism for project management.

D. TESTER/AUDITOR

A relationship must exist between each requirement and the individual tests being conducted to verify that requirements to allow test teams to validate that each requirement has been tested. By determining these relationships and using them to design tests, the tester can verify that all requirements are met and validated by the system design.

E. SYSTEM MAINTENANCE PERSONNEL

"The systems engineering team can use the web of relationships among requirements, design, and implementation to analyze the impact of a change among requirements, design

and implementation." (Nejmeh, et al, 1989, p. 1) Using requirements traceability when implementing changes to a system will provide the maintainer with a tool that can readily trace what code modules are directly effected by a requirement change. This includes identification of residual changes and changes to documentation.

APPENDIX B

TRACEABILITY CASE TOOLS

The purpose of this Appendix is to increase contact, awareness, and understanding of traceability CASE tools. Use of this report should be the first step in putting effective traceability processes, methods, and tools into practical use. The main users of these CASE tools are organizations responsible for the development and maintenance of complex computer systems, although several other applications of traceability have been identified in Chapter 3.

The authors familiarized themselves with four of the leading CASE tools available today: Requirements Driven Development (RDD-100), Requirements & Traceability Management (RTM), Teamwork/RQT, and Requirements Traceability System (RTS). Each of these tools will be discussed to familiarize the reader with its capabilities and limitations. No effort will be made to compare the tools to one another.

A. REQUIREMENTS DRIVEN DEVELOPMENT (RDD-100)

1. Background

RDD-100 was developed by Ascent Logic Corporation in San Jose, CA. Unlike traditional tools which seem to focus on specific parts of the development phase, RDD-100 supports systems engineering throughout the life cycle from Mission Needs and Requirements Analysis to system architecture design to specification of all aspects of the

system including design, test, production, deployment and support. RDD-100 supports traceability during the total development as well as upgrades to existing systems.

2. Platforms/Operating Systems Supported

The tool operates on most workstation platforms including SUN, DEC, Apple, HP and IBM. Newly available is the Requirements Editor that runs not only on the above platforms but 100% compatible PC's and Macintosh.

3. Highlights of Operational Characteristics

RDD-100 provides tools that facilitate the entire Engineering Process and include the following:

- RDD-100/SD System Designer - allows the users to approach the solution of any complex systems engineering problem by breaking it down into manageable pieces. This allows the user to analyze the requirements and trace those requirements to specific behavior which is then allocated to components (e.g. hardware, software, people and environment). The System Designer also provides an executable graphical model of the behavior model allowing the user to simulate performance or resource specification of a system even at a very high level of design.

- RDD-100 System Description Database - tracks requirements, behavior and component architecture with objects in the database which are based on the Element-Relationship-Attribute (ERA) model of data. The ERA model has built-in attributes such as Name, Description, Creation Date, Modification Date, and others. The database can provide the crucial "traceability" information necessary for managing and tracking large, complex systems.

- RDD-100 Extensibility - The database is capable of being extended with new elements, attributes, relationships and even functions. This gives the user's of RDD-100, and it's supporting tools, the capability to customize their system's development needs.

- RDD-100 Integrated Views - enables the user to view the database in either a graphical view or a textual view. The textual view includes a powerful template capability to define specific elements in a specific view. The graphical view

enables the data to be viewed in a variety of representation including hierarchy views, behavior diagrams, function and data flow diagrams.

- RDD-100 Behavior Modeling Notation - provides the users with notation to describe function flows and message sequencing, including concurrence, looping and replication. Since RDD-100 allows multiple graphical and textual views, a change made in any view will always be reflected in any other. --> RDD-100 Dynamic Verification Facility - provides the user with the ability to create an executable simulation model with which to demonstrate the actual operation of the target system.

- RDD-100 Data Sharing - provides for the sharing of data among groups of engineers. The Multi-User Merge (MUM) allows users to define specific elements in the database to be partitioned out and assigned to engineers. All subsets can then be exported back into the master database.

- RDD-100 Reports - provides for the generation of MIL-STD-490A and DoD-STD-2167A reports as well as custom reports. RDD-100 allows for the creation of templates for accessing data in the database.

- RDD-100 Interfaces - allows for the passing of data between RDD-100 and several CAE and CASE tools. RDD-100 serves as the design integrator assuring consistency and coordination during the entire development project.

B. REQUIREMENTS & TRACEABILITY MANAGEMENT (RTM) - Marconi Systems Technology

1. Background

RTM is a tool which was developed in the United Kingdom to support requirements traceability. It has the ability to be customized to meet the user's preferred system development life cycle and methodology. By allowing the user to define what information (object) is relevant for the chosen life cycle and what inter-relationships exist between the objects RTM provides flexibility to meet various development methodologies. This benefit permits the user to tailor the RTM tool to meet the specific traceability needs of the project.

2. Platforms/Operating Systems Supported

The tool runs on various systems, including: Sun SPARC station and Sun-3 (version 2), DEC VAX station 4000 (VMS) (version 1.2 only), Hewlett Packard 9000/700 (version 1.2 only), and IBM RS/6000 (version 1.2 only).

3. Highlights of Operational Characteristics

- Requirements Stripping - captures requirements by paragraph or ID in an automatic or manual mode. Allows for the insertion of attributes manually and puts a placeholder in the original document where the text was.

- Requirements Editing - Allows for the selection and editing of any requirement, attribute, Query, or clarification text.

- Requirements Expansion - Allows for the breakdown of requirements that are actually multiple requirements into several pieces. Allows traceability to be supported from each piece and also allows decomposition into different classes.

- Requirement Focus - Focus multiple requirements into one so allocation, testing, and implementation is done only once.

C. TEAMWORK/RQT - CADRE Technologies Inc.

1. Background

Teamwork/Rqt was developed for supporting requirements traceability throughout all phases of the development life cycle. The tool tracks progress and completeness by showing relationships between project requirements and actual deliverables.

2. Platforms/Operating Systems Supported

The tool operates on workstations running UNIX, ULTRIX, AIX, VMS, HP-UX, DOMAIN, and OS/2.

3. Reported Operational Characteristics

Automated tracing and reporting of requirements against deliverables keeps projects on target, on time, and within budget.

Consensus reporting helps establish system requirements.

Customized keywords and attributes for requirements, allocations, and targets simplify the allocation process.

Impact analysis demonstrates the effects of change to requirements, allocations, and targets before the change is made.

Custom rules-based parsing and WYSIWYG (What You See is What You Get) display makes documentation quicker and easier.

Automatic linkage to the Teamwork CASE environment improves communication and helps manage large systems development.

Open architecture supports integration with other tools.

D. REQUIREMENTS TRACEABILITY SYSTEM (RTS) - System Design Automation

1. Background

RTS is a tool which was developed to support requirements management. It is multi-user, network-compatible application designed to run on an IBM-compatible PC.

2. Platforms/Operating Systems Supported

IBM-compatible PC using DOS.

3. Reported Operational Characteristics

- Written in Paradox, a relational database, developed by Borland. The user-interface consists of a Paradox-style menu bar and is only three layers deep.

- Parsing Capability - RTS provides a parsing capability that automatically reads in ASCII files and assigns each requirement an ID number.

- Requirements - RTS tracks requirements to the "shall" statement level. A user-defined requirement type can be assigned to each requirement, allowing differentiation between requirements and non-requirements.

- Allocations - each requirement in the database can be assigned a user-defined allocation. Multiple allocations can be assigned to each requirement, if necessary.

- Traces - to track the flow down of requirements, the user can link source requirements to many lower-level requirements.

- Issues - Each requirement in the database can be assigned multiple issues. The issue feature allows the user to highlight problems areas at the requirement level. Each issue contains a description of the issue, the issue date, the issue status, and issue status date.

- Verification - Each requirement in the database can be assigned verification information consisting of test method, test level, test procedure, and procedure step. Each procedure defined in RTS contains the procedure name, test plan, applicable procedure dates, responsible person, and procedure status.

LIST OF REFERENCES

Department of Defense Standard DoD-STD-2167A 10.2.3, *Traceability to Indicated Documents*, 1987.

Hamilton, V. L., and Beeby, M. L., "Issues of Traceability in Integrating Tools," paper presented Colloquium by the Institution of Electrical Engineers Professional Group C1 (Software Engineering), Savoy Place, London, UK, Dec. 2, 1991.

MacLean, A., Young, R. M., and Moran, T. P., "Design Rationale: The Argument Behind the Artifact," in *Conference Proceedings of Human Factors in Computing Systems*, Austin, TX, May 1989.

Nejmeh, B. A., Dickey, T. E., and Wartik, S. P., "Traceability Technology at the Software Productivity Consortium," in Riter, G.X. ed., *Information Processing '89*, Elsevier Science Publishers B.V., 1989.

Thayer, R. H., and Dorfman, M., *Systems and Software Requirements Engineering*, IEEE Computer Society Press, 1990.

Wright, S., "Requirements Traceability - What? Why? and How?," paper presented Colloquium by the Institution of Electrical Engineers Professional Group C1 (Software Engineering), Savoy Place, London, UK, Dec. 2, 1991.

BIBLIOGRAPHY

Agusa, K., Ohnishi, A., and Ohno, Y., "A Verification Method for Formal Requirements Description," *Journal of Information Processing*, Vol. 7, 1984.

Baldo, J., "Reuse in Practice Workshop Summary," Institute for Defense Analysis, Alexandria, VA, Apr., 1990.

Brown, B., "Assurance of Software Quality," SEI Curriculum Module SEI-CM-7-1.1 (Preliminary), Carnegie-Mellon University, Pittsburgh, PA, Jul., 1987.

Greenspan, S. J., and McGowan, C. L., "Structuring Software Development for Reliability," *Microelectronics and Reliability*, Vol. 17, 1978.

Hadfield, S. M., "Interactive and Automated Software Development," (Master's Thesis), Air Force Institute of Technology, Wright-Patterson AFB, OH, Dec., 1982.

Jackson, J., "A Keyphrase Based Traceability Scheme," paper presented Colloquium by the Institute of Electrical Engineers Professional Group C1 (Software Engineering), London, UK, Dec. 2, 1991.

Keys, E., "A Workbench Providing Traceability in Real-Time Systems Development," paper presented Colloquium by the Institute of Electrical Engineers Professional Group C1 (Software Engineering), London, UK, Dec. 2, 1991.

Liu, L., and Horwitz, E., "A Formal Model for Software Project Management," *IEEE Transaction of Software Engineering*, Vol. 15, No. 10, Oct. 1989.

Pirnia, S. and Hayek, M. J., "Requirements Definition Approach for an Automated Requirements Traceability Tool," IEEE, New York, NY, 1981.

Ramesh, B., Abbott, A., Busch, M., and Edwards, M., "An Initial Model of Requirements Traceability," Technical Report NPS-AS-92-022, Naval Postgraduate School, Monterey, CA 93963.

Schneidwind, N. F., "Software Maintenance: Improvement through Better Development Standards and Documentation," Naval Postgraduate School, Monterey, CA, Feb. 2, 1982.

Walters, N., "Requirements Specifications for Ada Software Under DoD STD-2167A," *J. Systems Software*, 1991.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Professor B. Ramesh Naval Postgraduate School Monterey, California 93943-5000	2
4.	Professor T. X. Bui Naval Postgraduate School Monterey, California 93943-5000	2
5.	LCDR Timothy P. Powers 1026 Arctic Circle Juneau, Alaska 99801	3
6.	LT Curtis D. Stubbs 6 Burning Ember Lane Palm Coast, Florida 32137	2
7.	Commandant (G-TPP/HRP) U. S. Coast Guard 2100 Second St. S.W. Washington, D.C. 20593-0001	3