(2)

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

AN INVESTIGATION OF REQUIREMENTS TRACEABILITY TO SUPPORT SYSTEMS DEVELOPMENT

by

Gale Alicia Harrington

and

Kathleen Marie Rondeau

September, 1993

Thesis Advisor: *B. Ramesh*
Associate Advisor: *T. Bui*

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704

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

4. TITLE AND SUBTITLE AN INVESTIGATION OF REQUIREMENTS TRACEABILITY TO SUPPORT SYSTEMS DEVELOPMENT

5. FUNDING NUMBERS

6. AUTHOR(S) Gale Alicia Harrington and Kathleen Marie Rondeau

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)

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.
Department of Defense Standard 2167A mandates that requirements traceability be conducted during the development of government systems. This and other standards, as well as current literature, however, do not provide a comprehensive model of what information should be captured as a part of a traceability scheme. The primary goal of this research is to develop a model of requirements traceability at the level of systems design which relates requirements to all system components. An empirical study using focus groups was conducted with various stakeholders involved with the development of large, complex systems. Based on an analysis of the information obtained by the focus group sessions, a model for traceability was developed. This model describes the various relationships or linkages between requirements and system components that must be captured and maintained to support various system development activities. Finally, several issues which must be addressed in successfully implementing a comprehensive scheme for traceability are discussed.

14. SUBJECT TERMS Requirements Traceability, Systems Development

15. NUMBER OF PAGES
95

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited

AN INVESTIGATION OF REQUIREMENTS TRACEABILITY TO SUPPORT SYSTEMS DEVELOPMENT

by

Gale Alicia Harrington
Captain, United States Army
B.S., United States Military Academy, 1982

and

Kathleen Marie Rondeau
Lieutenant Commander, United States Navy
B.S. United States Naval Academy, 1980

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

Authors: Gale Alicia Harrington Kathleen M. Rondeau
Gale Alicia Harrington Kathleen Marie Rondeau

Approved by: B. Ramesh
Balasubramaniam Ramesh, Thesis Advisor

M. Sud for
Tung Bui, Associate Advisor

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

ABSTRACT

Department of Defense Standard 2167A mandates that requirements traceability be conducted during the development of government systems. This and other standards, as well as current literature, however, do not provide a comprehensive model of what information should be captured as a part of a traceability scheme.

The primary goal of this research is to develop a model of requirements traceability at the level of systems design which relates requirements to all system components. An empirical study using focus groups was conducted with various stakeholders involved with the development of large, complex systems. Based on an analysis of the information obtained by the focus group sessions, a model for traceability was developed. This model describes the various relationships or linkages between requirements and system components that must be captured and maintained to support various system development activities. Finally, several issues which must be addressed in successfully implementing a comprehensive scheme for traceability are discussed.

TABLE OF CONTENTS

I. INTRODUCTION

A. THESIS OBJECTIVES

The goal of this thesis is to develop a model for requirements traceability at the level of systems design which relates requirements to all system components. This model should provide the semantics of the various traceability linkages or relationships between requirements and various system components. It should also identify mechanisms for reasoning with traceability information to support systems development and maintenance activities. A secondary objective is to understand the critical issues that relate to the capture and use of traceability information in systems development.

A basic premise in the current research, the results of which are contained in this document, is that development of a model of traceability could be geared toward the needs of stakeholders at various stages of the systems development process. A variety of stakeholders are involved in the systems development process, including project sponsors, project managers, analysts, designers, maintainers, testing personnel, and end users. An empirical approach is used in this research to identify stakeholders' needs. Our study explores the information needs of various stakeholders.

Based on these stakeholder needs, we propose a conceptual model of requirements traceability. We also identify the critical issues to be addressed by various stakeholders when implementing a comprehensive scheme for requirements traceability. This study was conducted with real stakeholders in large scale, complex, real-time systems development efforts.

Given the above objectives, the following questions are addressed:

- What information should stakeholders capture as a part of requirements traceability?
- During what phases of systems development is this information captured, which stakeholders capture it, and then who maintains this information?
- How is this information used to relate requirements to systems components?
- What are the critical issues in implementing a scheme of requirements traceability which supports various stakeholders in systems development?

B. METHODOLOGIES

A thorough literature review of requirements traceability, was conducted to understand the state-of-the-art and current methodologies for requirements traceability. Prior to conducting focus group interviews, the authors attended a course on using focus group methodology for data collection.

Focus group interviews served as the means of data collection for this research. The focus group interview is

recognized as a valid and consistent qualitative marketing technique. Focus groups are highly flexible and appropriate for generating ideas, producing information, measuring potential problems, and encouraging creativity. A focus group is not a rigidly constructed question and answer session. It is a semi-structured exchange among group members which follows a clear agenda.

In this research, a total of 45 subjects in seven focus groups, with 5 to 8 participants each, discussed information requirements of stakeholders for requirements traceability. The groups were under the direction of a moderator (one of the authors of this thesis) who promoted discussion and ensured that the group stayed on the subject. Each focus group interview was between one and one half to two hours in length.

Focus groups were conducted at International Business Machines (IBM), Federal Systems Division, Owego, NY; National Aeronautics and Space Administration (NASA) Langley, Hampton Roads, VA; Naval Surface Warfare Center Dahlgren Division, Dahlgren, VA; Naval Surface Warfare Center Port Hueneme Division, Ventura, CA; and Naval Undersea Warfare Center, New London, CT. These organizations procure and manage the development of aerospace, communications, weapons, and aircraft systems, as well as perform systems integration.

The participants represented many levels of expertise and job responsibilities. The average years of education (after receiving a high school diploma) was 5.5 years, representing the following degrees: PhD, MBA, MS, MA, BS, BA. These degrees were from various academic areas: Electrical Engineering, Aerospace Engineering, Computer Engineering, Mechanical Engineering, Systems Engineering, Software Engineering, Physics, Geophysics, and Mathematics. Further, the participants represented an average of 17 years work experience in systems development. The participants were involved in systems development ranging from several hundred thousand lines of code to over several million.

Projects worked on by the participants included: Networking Technology, Aegis Reengineering Effort, Tomahawk Weapons System Analysis Tools, Hiper-D, Cassandra Data Analysis/ Reduction, Next Generation Computer Resources, Standard Missile, TARTAR, FFG-7 MK 92, Space Station Freedom, F-16, A-12, Real time Embedded Software Science Applications, New Attack Submarine, AN/BSY-2, EASE (Efficient Approach to Systems Evolution), Special Warfare Analysis and Engineering, Integrated Process Working Group, Navy Lamps MK-III, Combat Talon II, Block II Upgrade, and SBIS (Sustaining Base Information Services-Army).

C. SCOPE AND LIMITATIONS OF THE STUDY

Not all of the focus group participants were familiar with current traceability tools and techniques, but all had sufficient interest in the topic to actively contribute to the discussions. This research is designed to identify the information needs of the various stakeholders throughout the systems development process. Further, this research examines how captured information is maintained and used throughout the systems development process. Analysis of data from focus groups is used to develop a model for requirements traceability. It should be noted that this research is not designed to evaluate or compare current traceability tools nor is its purpose to develop a tool to support traceability.

D. ORGANIZATION OF THESIS

Chapter II provides a literature review on the general topic of traceability. This includes a definition of requirements traceability, its goals and objectives, why and how traceability is used, and the role of traceability in DoD/DoN. This chapter also conducts a brief presentation of some tools currently available that support requirements traceability.

Chapter III is based on an analysis of data collected from the focus groups. A model for requirements traceability is presented, concentrating on the semantics of

linkages as identified by the focus group participants. This model focuses on capturing the information needs of stakeholders.

Chapter IV describes the critical issues in implementing requirements traceability as discussed by the focus group participants. These issues must be resolved if a comprehensive scheme of traceability is to be successful.

The final chapter draws conclusions based on research data, and makes specific recommendations resulting from the research effort. This chapter concludes with recommended areas for additional research.

E. ACKNOWLEDGEMENT

The authors would like to thank the participants of the focus group interviews. Special thanks to Dr. Donna Rhodes of IBM, Dr. Susan Voight of NASA, Mr. Rick Stuttler of NSWC DD, Mr. Chuck Hogle of NSWC PHD, and Mr. Tom Choinski of NUWC. We are very grateful for all of their assistance in arranging participation in the focus groups and for providing logistics support during our research.

II. LITERATURE REVIEW

A primary concern in the development of complex, large-scale, real-time, computer-intensive systems is ensuring that the design of the system meets the specified requirements. As part of the systems development and maintenance process, many decisions and tradeoffs are made that affect a variety of system components. The requirements are subject to change and often evolve during the development process.

Due to the size and complexity of these systems, the entire systems development process has become quite challenging to manage. Funding, time, and personnel are often at a premium, as well as technological resources. Advancements in computer technology encourage increased tasking of systems designers.

In such a context, it is essential to maintain the traceability of requirements to various outputs produced during the system's design process, ensuring that the system meets the stated requirements and stays within established resource parameters. It is also necessary to acquire a greater knowledge of the concepts of traceability. The following is a brief review of literature focusing on

requirements traceability--how DoD and DoN view it, what is it, and methods and models of traceability, and some tools that use it.

A. TRACEABILITY IN DOD AND DON

As one of the world's major users of large-scale, computer-based systems, DoD takes a detailed approach to the dilemma of specifying systems requirements through the use of military standards. "Traceability is a key driver in defining and developing software under DoD-STD-2167A" (Walters, 1991, p. 174).

In February 1988, DoD specified its requirements for systems development in its DoD Standard, Defense System Software Development, DoD-STD-2167A. "Government software development standards in general, and DoD-STD-2167A in particular, are only required for military software which is mission critical" (Roetzheim, 1991, p. 12). In general, however, all projects for the military normally require that the standards are complied with. Further, nonmilitary government software for which reliability is a significant concern is typically done in compliance with these standards. "Most large defense contractors require the use of DoD-STD-2167A for all software development, including software developed using internal research and development funds" (Roetzheim, 1991, p. 13).

This DoD standard mandates that requirements be traceable through the entire system. Traceability, as used in this standard, means that "the document in question is in agreement with a predecessor document to which it has a hierarchical relationship." (DoD-STD-2167A, 10.2.3) DoD-STD-2167A formalizes the tracing of requirements (in documents) "from the initial set provided by the customer, to the contractor-written detailed requirements specifications, to the software and hardware design, and to the test procedures and results" (Walters, 1991, p. 174).

DoD-STD-2167A specifically requires the traceability of requirements to design. In accordance with this standard, the contractor must document the traceability of the requirements allocated from the system specification to each Computer Software Configuration Item (CSCI), its Computer Software Components (CSCs) and Computer Software Units (CSUs), and from the CSU level to the Software Requirements Specifications (SRSs) and Interface Requirements Specification (IRS). Additionally, this standard requires traceability of requirements to test cases. For this, the contractor shall document the traceability of the requirements in the SRSs and IRSs that are satisfied or partially satisfied by each test case identified in the Software Test Description (STD). "The contractor shall document this traceability in the STD for each CSCI." (DoD-

STD-2167A, 4.3.4) Finally, "the contractor shall document and implement plans for performing configuration status accounting" (DoD-STD-2167A, 4.5.3). This requires that the contractor generate management records and status reports on all products comprising the Developmental Configuration and the Allocated and Product Baselines. The status report shall, among several demands, "provide traceability of changes to controlled products" (DoD-STD-2167A, 4.5.3).

Traceability, as referred to in DoD-STD-2167A, has five elements:

- the document in question contains or implements all applicable stipulations of the predecessor document.
- a given term, acronym, or abbreviation means the same thing in the documents.
- a given item or concept is referred to by the same name or description in the documents.
- all material in the successor document has its basis in the predecessor document, that is, no untraceable material has been introduced.
- the two documents do not contradict one another. (DoD-STD-2167A, 10.2.3)

By this standard the government insists "that the functional requirements which are identified as part of the functional baseline be traceable directly to specific capabilities within the allocated baseline, which must then be directly traceable to specific capabilities within the product baseline" (Roetzheim, 1991, p. 129). The standard requires only that "the mechanism to ensure the requirements

are traceable throughout the process" (Walters, 1991, p. 174) without much elaboration on the type of information to be maintained to achieve this. A clear definition of the types of information, or relationships between various systems components that are part of a traceability scheme, is lacking.

Having a traceability scheme that provides a precise method for ensuring that requirements are met by the design is vital. DoD currently delineates its requirements to contractors in documents that are developed by numerous specialists in a format that may be thousands of pages long.

One of the foremost issues in developing an efficient and effective system involves the maintenance of consistency between requirements and design. This consistency entails meeting the initial requirements and maintaining requirements, design, and implementation consistently throughout the entire systems lifecycle. A key element included in a Request For Proposal (RFP) must be traceability, guaranteeing that the current set of requirements is met by the evolving system and the "tracing of all system requirements to the requirements stated for the capabilities, constituents, and interfaces." (Walters, 1991, p. 174). With declining defense dollars, systems must

be cost-effective, and be able to adapt to major changes during their lifecycle, without losing contact with the requirements.

The current method used by the Navy to specify requirements uses mostly a narrative, English format with supporting diagrams and charts. Ambiguities are frequent, as English specifications are inexact. If specifications are formally stated and can be transformed into designs in a formal manner, traceability between requirements and designs is a by-product of the design process itself.

It is commonly understood that changes to intricate systems can result in unforeseeable and disastrous effects to important national defense systems. These problems could be avoided if correct traceability methods are used along with proper maintenance of systems.

B. TRACEABILITY EXPLAINED?

There is no universally accepted definition of requirements traceability. Instead, there exists several ideas in which requirements traceability is defined in terms of its function within the system development process.

1. Definition

Hamilton and Beeby (1991) characterize traceability as the "ability to discover the history of every feature of

a system" (Hamilton and Beeby, 1991, p. 1). They go further to say that it is the ability to determine what has resulted from a change request.

Edwards and Howell offer a more generic and inclusive definition of traceability as a technique used to "provide a relationship between the requirements, the design, and the final implementation of the system" (Edwards and Howell, 1991, p. 3-8). These relationships are valuable to both the designer and the testers in that it allows designers to show that the design meets the requirements and that it also allows for early recognition of those requirements not satisfied by the design.

Greenspan and McGowan (1978) cite the use of traceability to effect changes in the entire system at various levels. Their definition of traceability looks at the systems description techniques and how they allow changes to any part of the design--requirements, specification, implementation--to be traced throughout the system. (Greenspan and McGowan, 1978, p. 79)

2. Goals and Objectives

The goal of requirements traceability is to provide a viable technique for linking design elements to the requirements in a bi-directional, complex manner. This linkage should occur across all design stages and design views. (Marconi, RTM Product Overview, 1991,1991) It is

imperative that traceability work in both directions, providing traceability from the requirements to the final design or output (forward) and from the output back to the requirements (backward). Forward traceability allows one to see where requirements materialize in the finished system whereas backward traceability allows reference to be made to the source of a design element or requirement. Finkelstein (1991) asserts that backward traceability is much harder to perform than forward. (Finkelstein, 1991, p. 1) In requirements engineering, for instance, the challenge is the identification and linking of various sources to the requirements.

To effectively prove requirements compliance, requirements traceability should also address how the requirements are arrived at as well as the design rationale that identifies not only the decisions, but also the supporting/opposing reasons behind those decisions (Ramesh and Edwards, 1993, p. 257). In software development, the objective of requirements traceability is "to ensure that the software produced meets the expectations of the user" (Marconi RTM Product Overview, 1991, p. 1). In doing traceability, software engineers will have confidence that their job has been done to the best of their ability and with the needs of the customer as their top priority. Also, the engineers will be working to the same goal from the same

starting point. "(This) will result in a product that meets the customer needs...the customer is happy." (Wright, 1991, p. 1)

Stehle lists some objectives of requirements traceability as to:

- promote a contractor and contracted method of working
- demonstrate that each requirement has been satisfied
- demonstrate that each component of the system satisfies a requirement. (Stehle, 1990, p. 10)

A primary concern in systems development is "ensuring that the performance of the system meets specified requirements" (Ramesh and Edwards, 1993, p. 256). The DoD currently delineates its requirements to contractors in documents that are developed by numerous specialists in a format that may be thousands of pages long. Having a precise method for ensuring that requirements are met by the design is vital; therefore, a key element included in a Request For Proposal (RFP) must be traceability, guaranteeing that the current set of requirements are met by the evolving system. Because of the decisions and tradeoffs made during systems development and the fact that requirements are dynamic, "it is essential to maintain traceability of requirements to various outputs or artifacts" to ensure the system meets stated requirements and that contractual obligations are met. (Ramesh and Edwards, 1993, p. 256) Further, traceability, when included in systems development

methodologies, aims to ensure that systems do only what is specified and no more (Stehle, 1990, p. 7).

3. Why Traceability?

One of the primary uses of requirements traceability is for system developers to prove to the customer that requirements have been understood, the product complies with those requirements, and the product has no unnecessary features (Wright, 1991, p. 1). "Since the Navy typically relies on contractors to design and build large, complex real-time systems, having a systematic way of validating that every requirement is met by the design is important not only to ensure that the system performs correctly, but also to determine whether contractual obligations have been met" (Edwards and Howell, 1992, p. 2-2).

Traceability is needed to ensure the closure of all systems components. It should alleviate problems of maintenance by identifying interdependencies among components and localizing the effects of changes made at various levels of systems design. (Ramesh and Edwards, 1993, p. 256) Also, miscommunication between the customer and systems engineer is a major factor resulting in project delays, cost overruns, delivery of projects not meeting customer specifications, and project cancellations. Traceability should facilitate communications between those involved in the project to alleviate some of these problems.

Gathman and Halker (1990) agree that the absence of clear traceability of design to requirements has created problems for project managers, configuration control and the customer. "Often the reason for delivery schedule slips is mismanagement of requirements traceability" (Gathman and Halker, 1990).

Traceability allows designers and maintainers to keep track of what happens when a change request is implemented. For example, during a development project, if a problem is discovered and if traceability exists the designers can determine the requirements incorporating that feature, the source of those requirements, and "what checks into the suitability and correctness of the design were performed." (Hamilton and Beeby, 1991, p. 1) In the same manner, if a change to the requirements is proposed, the designers will be able to ascertain the parts of the design which will be affected "by effectively relating each requirement to a specific element of the implementation" (Edwards and Howell, 1992, p. 3-8). This enables implications of a requirements change to be determined before system redesign takes place. (Edwards and Howell, 1992, p. 3-8) According to Finkelstein, change is consequent with systems evolution and requires a better

understanding of the requirements "which can only be achieved by going back to their source" (Finkelstein, 1991, p. 1)

Traceability within documents ensures that the source "of the environment's contained information is identifiable" (Cordes and Carver, 1989, p.185). This traceability defines a chain of accountability within the development process. "All information is linked directly to its generating statements within statements within the original document" (Cordes and Carver, 1989, p. 185).

Additionally, traceability enables engineers to locate related requirements at higher and lower levels, that must be reviewed to see if changes are needed. "It refers to the ability to cross-reference items in the requirements specifications with items in the design specifications." (Thayer and Dorfman, 1990) With complete traceability more accurate costs and schedules of changes can be determined than depending on the engineer or programmer to know all the areas effected by these changes. "Requirements traceability adds to what has to be done across the entire life cycle but will reduce the non-productive work by much more" (Stehle, 1990, p. 26).

4. Aspects of Traceability

Requirements traceability has broad applications throughout the systems development process that are a

function of the stakeholders need and the stage of development in the project.

a. Requirements Management

Requirements management is "the process of creating, disseminating, maintaining, and verifying product requirements with regard to performance, cost, manufacturability, maintainability, and other lifecycle concerns" (Fiksel, 1992, p. 1). Fiksel stated that "effective requirements management is a key success factor in integrated product development" (Fiksel, 1992, p. 1). In a survey of product development managers that addressed strategies related to the management of technical and business requirements for new product development, the most common problems noted were the "lack of adequate precision in requirement definition, the inability to communicate requirement changes in a timely manner and product delays due to the late discovery of requirement conflicts" (Fiksel, 1992, p. 1). Requirements traceability is essential for requirements management as it assists systems developers with requirements capture, tracking and verification.

b. Design

Smithers, Tang and Tomes (1991) studied design models that produced, among many deliverables, a design history record in which traceability of the design process is maintained. They state that "design history reflects the

process of design in the form of a design document in which a design result and its justifications are clearly stated" (Smithers, Tang and Tomes, 1991, p. 1). Having design history helps facilitate explaining and documenting final design results as well as repeating and restoring interrupted design sessions which become necessary on long term design projects where there is a need to review the details of previous design sessions. For design history to be meaningful and to support traceability, the following should be documented:

- results of the design
- justifications of the results
- important decisions or assumptions made during design
- contexts of the design solutions. (Smithers, Tang, and Tomes, 1991, p. 1)

These well documented design solutions and the resultant justifications provide a "major source of knowledge for maintaining traceability throughout a design project and for evaluating a design project" (Smithers, Tang, and Tomes, 1991, p. 2).

c. Testing

Brown asserts that effective traceability can also assist in ensuring that test procedures are updated whenever errors are discovered which were undetected by the applicable procedure. According to him, "test procedures

should be traceable to the requirement or design for which they demonstrate product compliance." (Brown, 1987, p.9)

Concerning the verification and validation process, Marconi Systems Technology declare that "traceability is the only technique for assessment of consistency between different lifecycle phases prior to coding" (Marconi Systems Technology, 1991, p. 17). They further explain that acceptance testing is directly used in assessing whether integrated systems meet the requirements statements. "Traceability is thus a major technique for risk management on a project." (Marconi Systems Technology, 1991, p. 17)

d. Maintenance

Schneidewind (1982) describes traceability as a mechanism to support maintenance, focusing on the maintenance phase to discover sources of error. He views traceability as a means of identifying technical information that pertains to a software error detected during the maintenance phase and "thereby trace the error to the applicable design specifications and user requirements" (Schneidewind, 1982, p. 4). Keuffel states that "having a requirements document with traceability to data structure and function design will make maintenance far easier and far more accurate" (Keuffel, 1990, p. 33).

e. Quality

Wright relates requirements traceability to quality as viewed by the customer. Quality "is the degree of compliance to their needs the product exhibits." (Wright, 1991, p.2) What requirements traceability provides is a "change in focus from the product and what it does to what the customers wants--thus, ensuring that the product is of a high quality as perceived by the customer" (Wright, 1991, p. 2).

f. Computer security

From a computer security approach Murine identifies traceability as an essential factor in the establishment of data integrity. He further defines security traceability, one of the criteria for Software Security Metrics, as "those security requirements that provide a thread from the system security requirements to the implementation with respect to software development and security environment." (Murine, 1986, p. 337)

g. Reuse

In a reuse environment, automated traceability assists the user understanding of the component's design and implementation "since it captures the context and the constraints of the development process," therefore assisting the users of the component in reusing it on another application. (Baldo, 1990, pg. xii)

h. Other uses

Other uses for traceability are in "safety analysis, audits, change control and general completeness of the design" (Hamilton and Beeby, 1991, p. 1). Brown advocates that "every software product should be traceable back to the product from which it was derived" (Brown, 1987, p. 9). With a comprehensive scheme for traceability, "it should be possible to identify the requirement or design decision from which each algorithm in the software product is derived" (Brown, 1987, p. 9). Traceability will make it easier for designers to determine phase completion and product completeness. "It supports the accomplishment of reviews and evaluations, and provides for increased confidence in the accuracy of requirements verification." (Brown, 1987, p. 9)

5. Who Benefits With Traceability

A number of stakeholders, each having a different set of goals and priorities, are involved in the systems development process, including project sponsors, project managers, analysts, designers, maintainers, and end users.

Stehle describes how requirements traceability is of value to the end user in the following ways:

- Providing an understanding of the purpose behind each component of the specification or design
- receipt of design documents specific to those parts of the system they are responsible for

- ability to determine that requirements are satisfied through workable and acceptable solutions. (Stehle, 1990, p. 9)

Stehle also lists the benefits of requirements traceability to those stakeholders who are active participants in the systems development process as:

- ensuring a problem oriented approach to logical design and physical implementation
- better estimating required resources, milestone dates and development costs through greater understanding of the factors involved
- providing additional focus for quality assurance of the development process and proposed system
- ability to assess the impact of proposed changes to the requirements because the chain of system components that will be affected can be traced. (Stehle, 1990)

Maintainers benefit from traceability once a change is required. "A maintainer needs to be able to trace that change back to the requirements that necessitated or triggered it, and to pinpoint which parts of design/implementation are affected by the change." (Ramesh, et al, 1992)

C. METHODS AND MODELS OF TRACEABILITY

One of the foremost issues in systems development entails the maintenance of consistency between requirements and design. Such consistency involves meeting the initial requirements and maintaining requirements, design, and implementation consistently throughout the entire system

lifecycle. In the past, system malfunctions caused by changes in the requirements, resulted in unforeseeable and sometimes disastrous effects. Some of these projects involved critical national defense systems. With proper methods and models of requirements traceability, these problems might have been avoided.

1. METHODS

Before a model of traceability can be developed it is important to first develop a method for capturing and using traceability information. Schneidewind (1982) provides several formats for achieving traceability. These include: event tables which relate modes, events and actions; condition tables which relate modes, conditions and actions or values; and, selector tables which relate modes and mutually exclusive characteristics of modes. These tables provide a clear means of identifying and capturing information needed for traceability.

Macmillan and Vosburgh (1986), speaking on Software Quality Indicators, state that a major management concern for software development is that all requirements for a system are translated into detailed specifications of software functions. To facilitate this they recommend that a requirements traceability matrix be created to map system requirements to software functions and serve as "the basis for a completeness indicator" (Macmillan and Vosburgh, 1986,

p. 47). The completion indicator measures progress, "because the percentage of requirements addressed is an indication of the amount of work accomplished" (Macmillan and Vosburgh, 1986, p. 48).

West (1991) also suggests the use of matrix analysis to "develop design requirements from customer requirements, product functions from design requirements, test requirements and process requirements from customer and design requirements, and so on" (West, 1991, p. 1). The matrices are then linked, with the output on one as the input of another, therefore incorporating the whole development process...The rationale for decisions can be traced back all the way from shipping the product back to the actual customer requirements" (West, 1991, p. 6). West uses these matrices in his Quality Function Deployment (QFD) Methodology for software development. QFD is not a requirements gathering process, but rather is "a methodical way to ensure product functions address customer needs and that no requirements are ignored or overlooked in the development process" (West, 1991, p. 6).

A method recommended by Greenspan and McGowan (1978) emphasizes the careful definition of reliability requirements through the use of Structured Analysis and Design Techniques (SADT). This technique is a means of structuring/documenting the development process with a major

emphasis on traceability that allows changes to any part of the design to be traced throughout the entire system. Traceability is "the property of a system description technique which allows changes in one of the three system descriptions--requirements, specification, implementation--to be traced to the corresponding portions of the other descriptions." (Greenspan and McGowan, 1978, p. 79)

"Hierarchical Progression Analysis (HPA) is a methodology which provides traceability through a software system by relating the data and control flows between the various disciplines needed to develop and verify a system" (Andrews, 1984). It also addresses the problem of maintaining traceability during the translation of requirements into design, design into code, and the "subsequent hierarchical proof of the correctness and completeness of that translation" (Andrews, 1984). HPA fills the traceability void in the systems development process in that it enables the developer to maintain continuity throughout implementation, providing testers with a method of ensuring each requirement is tested, providing managers with a tool to evaluate the impact of proposed changes, to measure work to be done and to establish a high level of confidence in the correctness of the completed system. (Andrews, 1984)

Jackson presents a method of traceability using keyphrases to express relationships between entities. The keyphrases, extracted to a file, are compared and traceability information is provided. A Relational Information Management System (RIMS), centered on an Oracle DBMS, loads the keyphrases into a database and provides early warning of impact and helps to forecast the cost of changes. (Jackson, 1991, p. 3)

Edwards and Howell (1992) maintain that a key issue in developing a correct long lasting system is to ensure consistency between requirements and design. They have developed the requirements specification and traceability methodology (ReSpecT). Under the ReSpecT methodology the requirements aspect of system development is divided into three parts: conceptual view, formal representation, and design elements and system to which requirements are traced. (Edwards and Howell, 1992, p. 3-1)

This methodology is vastly different from previous, informal methods consisting mainly of English text. ReSpecT methodology allows for the designer to identify ambiguities while also checking for consistency of information. These capabilities would be beneficial in the maintenance of these systems. ReSpecT techniques ensure the final design meets the requirements specified by the customer.

2. MODELS

Liu and Horowitz (1989) noted that there is great concern about the lack of an appropriate model of the software development process for the development of large-scale software and stated many reasons why it is important that a good model be available. Briefly these reasons are:

- A model is a means of communication between the stakeholders.
- A model provides assistance in the management of the process by providing milestones that can be examined to determine the rate of progress of the project.
- A model gives software engineers a foundation for building tools that support and enhance the software process. (Liu and Horowitz, 1989, p. 1280)

Whereas a single model might not provide all the angles of software development, Liu and Horowitz (1989) suggest the following essential features for a successful model:

- Describe an evolutionary design process.
- Provide for a large-scale project that functions with parallel processes.
- Indicate all artifacts produced at various points in the process.
- Indicate activities and resources affected by failed activities and then allowing for their reevaluation.
- Indicate that activities exist with diverse conditions.
- Indicate the extent and nature of resources involved in a subtask, including people, consumable, and nonconsumable resources. (Liu and Horowitz, 1989, p.1281.)

Liu and Horowitz propose the DesignNet model that attempts to satisfy all of the above criteria. DesignNet is an open system model which is designed for describing and monitoring the software development process using AND/OR graphs and Petri net notation "to provide the description of a project work breakdown structure and the specification of relationships among different project information types (activity, products, resource and status report information)." (Liu and Horowitz, 1989, p. 1280) It provides an automated information tracking mechanism that records project history. With this information the project manager can query the project database to analyze and reason about the project's progress. Liu and Horowitz show how the waterfall life cycle model maps onto a DesignNet and the Implications for project planning, cost estimation, project network construction, re-initiation of activities, and traceability across the life cycle. (Liu and Horowitz, 1989, p. 1292)

Horowitz and Williamson (1986) describe the data abstraction model used in Software Documentation Support (SODOS), a computerized system which supports the definition and manipulation of documents used in developing software. SODOS "permits traceability through all phases of the software life cycle, thus facilitating the testing and maintenance phases" (Horowitz and Williamson, 1986, p. 849).

Most system development life cycle models consist of the following stages: analysis, design, implementation, test production, testing and reviews. Wright (1991) suggests that a model should augments this traditional lifecycle with a new stage which he describes as "allocation, the task of mapping requirements onto something that is a necessary evil to that which is vital to both the life cycle, and the success of requirements traceability" (Wright, 1992, p. 2). This new model would rely on the review phase to ensure the correctness of the allocations by determining when a deliverable--something that is produced--becomes an acceptable--something that is produced and that is wanted--product. (Wright, 1991, p. 2)

Ramesh and Dhar (1992) present a conceptual model, REpresentation and MAintenance of Process knowledge (REMAP) that relates process knowledge to the objects that are created during the requirements engineering process. This model includes as part of it the Issue Based Information Systems (IBIS) designed to record argumentation related to deliberations. Support for various stakeholders involved in software projects can be provided by capturing the history about design decisions in the early stages of the systems development life cycle in a structured manner. (Ramesh and Dhar, 1992, p. 1)

Ramesh and Edwards (1993) state that a model that represents traceability information at the level of systems design relating requirements to all system components needs to be developed. "A comprehensive scheme for maintaining traceability, especially for complex, real-time systems, requires that all system components (not just software), created at various stages of the development process, be linked to the requirements" (Ramesh and Edwards, 1993, p. 256) This model should include semantic models that support the various stakeholders in systems development activities. Models, supported by reasoning mechanisms to support various stakeholders, that capture essential features of systems development process, such as the design rationale are components of a useful traceability scheme. (Ramesh and Edwards, 1993, p. 259)

D. TRACEABILITY TOOLS AND CURRENT EXPECTATIONS

One important aspect of any software development environment designed to assist with software development to government standards is the ability to automatically document requirements traceability. (Roetzheim, 1991, p. 129)

A number of traceability tools with a wide variety of capabilities have been developed by the industry, both for in-house use and as commercial products. Since these tools vary widely in their applications, and as yet there are no

industry standards for them, we do not attempt to make comparisons nor appraisals of their features. Rather, a brief description of capabilities of a few tools is presented to provide an overview of features available in tools today.

1. ARTS

One of the earliest systems to capture and use traceability data was Automated Requirements Traceability System (ARTS), a bookkeeping program developed to manage the requirements of a large, error-prone aerospace system. ARTS operates on a data base that includes systems requirements and their characteristics. It allows for automated tracking of requirements as they are partitioned and apportioned to lower-level requirements. ARTS provides database management and output operations on requirement-related attributes selected by the user. The major function of ARTS "is to provide rapid and accurate traceability, upward and downward, in a requirements hierarchy or tree." (Dorfman and Flynn, 1984, p. 63) Like ARTS, other tools often focus on the database management issues related to maintaining links between requirements and the different components of the system.

2. Teamwork/RQT

Some of the traceability tools provide for manual parsing and grouping of functional requirements. One such

tool is Cadre's Teamwork/RQT. Other capabilities include point-and-click allocation of requirements to targets, navigation through allocation channels to integrate the entire life cycle, and the ability to propagate allocations between parent and child entities. Teamwork/RQT is a comprehensive package providing automatic model consistency checking that "provides a simple means of automatically generating a traceability report on (those requirements) which have been satisfied by which parts of the model" (McCausland, 1991, p. 1).

Cadre describes Teamwork/RQT and its concept of requirements traceability as being able to reveal the "mapping between requirements and the deliverable components which are intended to satisfy them" (Cadre Technologies, Inc., 1990, p. 6). They further state that compliance is proven in a two-step process:

- Show the correspondence of requirements to deliverable components. A table which shows this correspondence is called a traceability matrix.
- Show that the corresponding deliverable components correctly satisfy the requirements. (Cadre Technologies, Inc., 1990, p. 6)

3. RT

Teledyne Brown Engineering's Requirements Tracer (RT) is designed to be used throughout the entire system life cycle to "define, analyze, and trace system requirements." (Teledyne Brown Engineering, 1991, p. 1) RT

uses a database of natural language requirements that, through the use of pre-defined criteria, allow for the establishment of relationships between requirements. A requirements traceability matrix is created that provides assistance in verifying the proper allocation of all requirements. The user of the tool then generates customized reports which provide user-selected sets of information to support requirements traceability. (Teledyne Brown Engineering, 1991, p. 1)

Capabilities of RT include such tracing mechanisms as parent/child relationships (and how to determine them), functional hierarchy, keywords, attributes, querying, requirements extraction, and customized report generation. Requirements can be allocated to functions or subfunctions by either direct entry or selection from a previously defined list.

4. RTM

Marconi Systems Technology's Requirements and Traceability Management (RTM) is a traceability tool that provides project configurability (specifying where traceability is wanted), requirements engineering, requirements traceability, and documentation. By using more than one type of link between objects, RTM assists in ensuring that the software produced meets the expectations of the user.

5. RDD-100

The objectives of traceability with RDD-100 include to: indicate which portions of the system design satisfy specific requirements, record issues and resolutions as the foundation for subsequent changes to originating requirements, identify the set of decisions that led to the baseline requirements, link verification methods to design elements, and provide consistency between documents.

6. Shortfalls With Current Tools

Current traceability techniques tend only to provide mechanisms to represent relationships among objects without providing any guidance on what useful relationships are and how this information will be useful during the lifecycle of a system. Contemporary methods yield some traceability through simple linking techniques that relate requirements to design.

According to Edwards and Howell (1992), shortfalls in today's technology for requirements traceability techniques include that current tools:

- do not capture how the requirement is satisfied by the design just the fact that some relationship exists
- lack the ability to trace back from the actual pieces of design and implementation to the requirements
- do not have a method for tracing from a particular piece of hardware or humanware back to the requirement. (Edwards and Howell, 1992, p. 2-4)

E. CONCLUSION

A comprehensive model for maintaining traceability, especially for complex, real-time systems, requires that all system components (not just software), created at various stages of the development process, be linked to the requirements. To achieve this it is essential that traceability be maintained through all phases of the systems development process, from the requirements as stated, or contracted, by the customer, through analysis, design, implementation, and testing to the final product.

Additionally, it is critical that consistency between the requirements and the design is maintained, especially in situations where an organization relies on outside contractors for developing systems. Having a systematic way of validating that each requirement is met by the design is important, not only to ensure that the system performs correctly, but also to determine whether contractual obligations have been met.

The need to provide traceability is recognized in standards that regulate the development of systems for the U.S. Government; yet, there is no clear definition of the types of information or relationships between the various system components that are part of a traceability model. Neither the standards that require traceability as a part of any systems development effort nor the current literature elaborate on the specific types of traceability linkages to

be maintained. Though current tools provide mechanisms to represent various types of linkages between system components, the interpretation of the meanings of such linkages is left to the user. Finally, the focus of the majority of the literature reviewed catered to traceability at the level of software design, rather than at the level of system design.

III. REQUIREMENTS TRACEABILITY MODEL

A. INTRODUCTION

A principal challenge in this thesis is the development of a model that represents and provides the semantics of various traceability linkages or relationships between requirements and system components. A basic premise in our research is that development of a model of traceability could be geared towards the needs of stakeholders at various stages in the systems development process. Using focus groups, we conducted a study to identify the traceability information needs of various stakeholders during systems development.

In this chapter we discuss results from the analysis of data collected during our study. First we reviewed the context in which traceability information is likely to be used during systems development; i.e., from the perspectives of key stakeholders in terms of the different types of traceability information that would be of interest/use to them. Then we looked at specific traceability linkages or relationships between various system design products and processes that would support these needs.

A traceability information model has been developed based on this analysis. Following is a discussion of the semantics of various types of traceability information

represented as linkages, mechanisms for their capture, and their use in systems development activities. We have segregated the model into three parts for clarity of presentation: Requirements Management, Design/Implementation and Allocation, and Compliance Verification. Each part is further segregated by functional features. In this chapter, traceability linkages will be identified by uppercase, bold faced letters (**LINKAGES**), while components that they link are indicated with uppercase, italic letters *(COMPONENTS)*. For every link in the model an inverse may be defined.

B. REQUIREMENTS MANAGEMENT

A recurring theme among the subjects was that traceability, when implemented correctly, would greatly benefit requirements management facilitating requirements understanding, capture, tracking and verification. The following is a discussion of the Requirements Management Model presented in Figure 1.

1. Organizational Objective

ORGANIZATIONAL-NEEDS must **JUSTIFY** *SYSTEM-OBJECTIVES* as systems are built primarily to satisfy these needs. They are further broken down into *OPERATIONAL-NEED*, the functional needs of the organization and *MISSION-NEED*, the strategic needs, as denoted by the **IS-A** hierarchy. These two needs **SUPPORT** each other and are the foundation used to corroborate the Organizational-Needs.

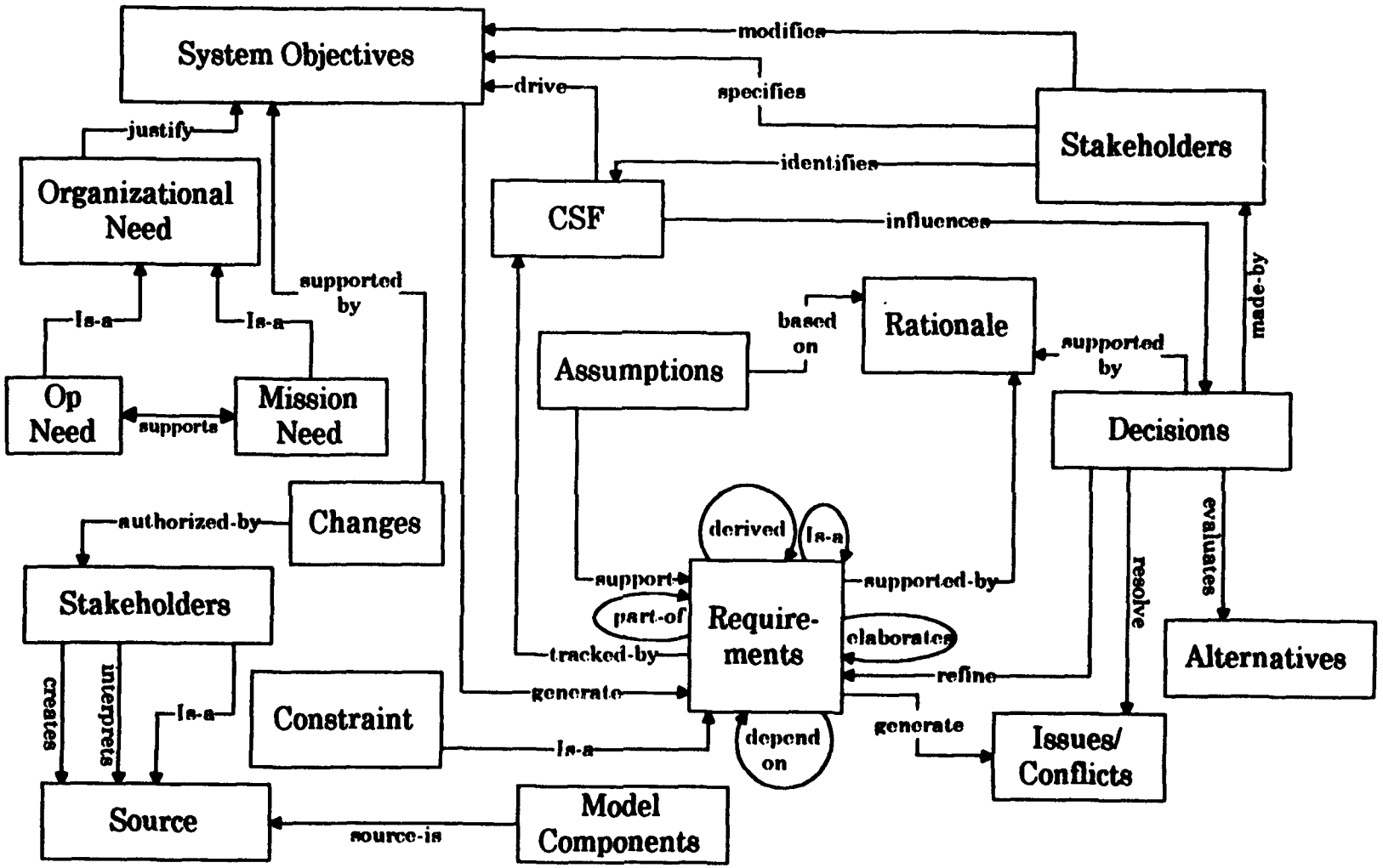

Figure 1: Requirements Management Model

2. System Objective

STAKEHOLDER, be it the customer, program manager, program sponsor, etc. **SPECIFIES** the *SYSTEM-OBJECTIVES*. The *REQUIREMENTS* for the system are **GENERATED-BY** these System Objectives.

3. Critical Success Factors (CSF)

STAKEHOLDERS **IDENTIFY** various *CSFs* for the system. These stakeholders could be anyone, but most likely would be the Customer, System Developer, and/or Project Manager. ("Stakeholders have some issues or values like, no mistakes or system errors, performance is not the most important thing, it's no mistakes.")[1] These CSFs could be whatever is considered critical to the projects' development: Mission Criticality, Cost, Time, Weight, Voltage, etc. They **INFLUENCE** *DECISIONS* made during requirements management. Further, the Systems-Objectives are constrained or **DRIVEN** by the *CSF* (Critical Success Factors). The overall requirements traceability focuses on the CSFs **IDENTIFIED** by the Stakeholder, and facilitates requirements being **TRACKED-BY** those CSFs.

Mission criticality (a CSF) of requirements could be used in classifying and monitoring them. Similarly, CSFs such as weight and cost can be used in tracking and managing

---

[1] This is a direct quote from a subject participating in a focus group. Henceforth, all quotes from a subject will be enclosed in parentheses and quotation marks, but no specific reference will be made.

requirements. Due to the large number of requirements associated with many of today's projects, it is important to determine those which are critical to the success of the project and to trace them and others they relate to. It is necessary to decide which requirements should be tracked and which are of lesser importance. ("...just try to get that information, and not get the rest of the requirements that may not be as important.")

("There's a dependency of total dollars that are spent or the total of whatever that is spent.") ("Basically, you've got schedule margins, cost margins.") ("A lot of decisions have been made...because of cost and weight, mostly cost.") ("It will be a combination of technical and cost trades.") *DECISIONS* are always made considering the CSFs to determine, among many things, whether they are feasible, cost effective, or desirable.

As part of the negotiation process among stakeholders when making decisions, there are many tradeoffs made depending on the availability of resources and whether they are considered to be CSFs. ("Typically we enter into a complex negotiation process of what's in scope, what's out of scope, what's affordable, what's not affordable. And I believe this is where traceability earns its keep.")

As all requirements are not equal in significance or criticality, various requirements may be traced through the

lifecycle at different levels or granularity of detail in order to optimize resources spent on such capture. It may be unnecessary or even undesirable, considering the overhead involved in maintaining traceability, to maintain linkages between every requirement and every output created during the systems design process.

As *REQUIREMENTS* are **GENERATED-BY** *SYSTEM-OBJECTIVES*, they can be prioritized throughout the lifecycle to provide stakeholders with a view to understand and evaluate whether the system supports these CSFs. ("...the smaller subset is the driving, high risk set of requirements that merit more visibility in the tracking of the program...we have an attribute of a requirement that we call technical performance measure (TPM). Is it high risk, do we want to put a TPM against this requirement? The TPMs are all programmed, tracking, giving them higher visibility to see how we're doing.")

4. **Change**

One of the biggest challenges in managing today's large, complex systems is the way that requirements are constantly evolving and changing. Each focus group agreed that to accurately reflect this volatility, a dynamic model of requirements traceability is needed. For military applications, not only is the nature of systems requirements

dynamic, but even the organizational-needs (mission and operational) and the systems-objectives also.

Systems that were considered successful during the Soviet threat era are no longer considered useful as strategic and mission needs are evolving rapidly. It is not possible, however, to abandon all investments in military systems and develop new systems to meet current needs. The ability to identify components that are linked to various objectives and modify them to suit the current situation is needed. *STAKEHOLDER* **MODIFY** *SYSTEM-OBJECTIVES* based on revised mission needs. When this happens, the modified *SYSTEM-OBJECTIVES* **GENERATE** changed *REQUIREMENTS*.

It is very important that the model captures such changes. We have identified basically two types of changes, those that fix a deficiency in system design (as identified say, during testing) and those that occur because of the dynamic nature of the system that is being developed. The former aspect is covered in the discussion on design and allocation procedures.

When new or deleted requirements enter the system, they are **AUTHORIZED-BY** at least one *STAKEHOLDER* and **SUPPORTED-BY** the *SYSTEMS-OBJECTIVE*. *CHANGES* to *REQUIREMENTS* follow such revised needs.

5. Requirements Evolution

Traceability links the different levels of requirements as they evolve through the various stages of the development lifecycle. This provides for recursive links between the requirements as they change and evolve through each phase, providing a historical record of requirements information. Linkages map the requirement back to the original requirement thus allowing the users to have complete understanding of where the requirement comes from.

REQUIREMENTS are often **DERIVED** from other requirements, often based on assumptions made by stakeholders in their interpretations. These derived requirements need to be tracked carefully as they are likely to be a major source of conflicts and issues, and subject to changes as the interpretations and assumptions vary. They are also the cause of an "explosion" in the number of requirements. ("...if you can start with 30 pages of requirements, top level, then you can go to your prime item development specifications and you could consider those 9000 requirements derived from those 30 pages, but then you go in those 9000 and there's going to be derived requirements between those second and third generation...you have an explosion of derived requirements and relationships.")

Requirements that are derived ("create a collection of requirements at a different level which defines this new

entity, or new thing at a lower level. Then you do a one for one traceability of the originating requirements with the requirements that partially satisfies it.").

Some *REQUIREMENTS* are **ELABORATED** by others, providing further explanation or clarification. This added detail assists in understanding the assumptions and interpretations made. ("Sometimes you add requirements really because you find them. They are refined, and become more specific in detail then in fact are more demanding than the original requirement might have been.") Requirements also **DEPEND-ON** others. An example would be an application requirement may depend on a specific software that in turn depends on a specific hardware. Identifying these dependencies is important, especially when implementing changes, so that the impact of the change to the entire system can be determined.

Breaking down larger requirements may generate smaller *REQUIREMENTS* that form a **PART-OF** it. This provides information of how the pieces fit together and is especially important on large, complex systems that require *REQUIREMENTS* to be further developed. ("Higher levels are just abstractions of lower levels.") Finally, **IS-A** links show the parent/child hierarchical relationships between requirements. ("Initially it's a top-down, hierarchical decomposition starting with the most general goals of the

customer, flowing down to a set of systems performance requirements...you now can have children with multiple parents, whereas originally we were probably looking at children with a single parent.") An example of an **IS-A** hierarchy would be: Computer Software Unit (CSU) is a Computer Software Component (CSC) is a Computer Software Configuration Item (CSCI) is a Segment is a System. (Roetzheim, 1991, p. 23)

CONSTRAINT may be treated as a type of *REQUIREMENT* (denoted by the **IS-A** link). Every focus group stated how various constraints become hard requirements because they set limits within which the system can be developed. Systems are designed and decisions made with constraints considered.

6. Decision Making

REQUIREMENTS **GENERATE** (or lead to) *CONFLICTS/ISSUES*, that often occur due to differing interpretations, assumptions, interests, viewpoints, experience and objectives of the stakeholders. Information about the decisions made, assumptions in place, status of issues, and change activity must be maintained throughout the system lifecycle to ensure that customer requirements are understood and satisfied. ("Traceability pays for itself by making sure you understand how the requirement is interpreted, how it's being

implemented, what were the issues that were considered in the design to satisfy that.")

DECISIONS **MADE-BY** *STAKEHOLDERS* provide a basis with which requirements are **REFINED**. During decision making, to **RESOLVE** *CONFLICTS/ISSUES*, various *ALTERNATIVES* that could lead to their resolution are **EVALUATED**. Though the chosen alternative is well documented in many cases, the other alternatives considered are not recorded. It is important to capture this information as it may become relevant in a changed context. Availability of the information may avoid expensive rework on the same decisions. ("We debated that for two weeks now. Why was it thrown out years ago when we got one camp arguing to put it back in and another to keep it out. And if we could have gone back and assessed why we made that decision two years ago it would have saved us some time.")

It is important to track the *RATIONALE* that **SUPPORT** the *DECISIONS*. This *RATIONALE* is **BASED-ON** the *ASSUMPTIONS*. Also, it is desirable to keep track of the assumptions as explicitly as possible for these are the most likely candidates for reevaluation and change, leading to changed requirements. Besides the *RATIONALE* that support decisions to resolve issues, the rationale behind various requirements may also be represented to identify their justification.

Further, the *ASSUMPTIONS* that **SUPPORT** *REQUIREMENTS* must also be explicitly captured.

Over the lifecycle of a project, the personnel, requirements, assumptions, etc. change and this change leads to a lot of rework as this decision history information is typically not maintained. ("We already looked at that tradeoff, that approach, but the person that did it is going and the person that reviewed it is gone, so we churn the whole thing again.") ("This is a lengthy look at all these phases with a lot of people that contribute to, touch, change, add, use this information. If you just have an understanding about the linkages and relationships, that to me is just the information model that you're going to maintain. But who produces it, when do they produce it, in what order, how's the information shared intelligently are all equally important issues.")

7. **Source**

A source is the person or thing that "documents" each component in the model. There are many manifestations of *SOURCE*: documents, manuals, persons, meetings, conversations, standards, contracts, etc. *SOURCE* is traceable to every node and link of the model. It is desirable to be able to go back to the document, or whatever the **SOURCE-IS** and examine the actual text or quote from a stakeholder. ("Normally what happens is you have a set of

requirements that go back to one source and you'd interpret what they said, so you need to have that original source to be able to go back and verify that your interpretation was correct.") ("There are all kinds of customers out there...so it's real important to know which of those stakeholders has that requirements because you have all kinds of conflicts.")

One would like to get back to the source for understanding, clarification, etc. *STAKEHOLDERS* **CREATE** *SOURCES* of requirements. It is necessary to know who created these sources for future reference as clarification might be needed. ("You want to know who, when, where and possibly why that's a requirement.") *STAKEHOLDER* **IS-A** *SOURCE* since, at any time during development, they can be a source of a requirement, rationale, etc.

Also, during the systems development process they often **INTERPRET** items of various sources by stakeholders, be it a requirement, a standard, a need, etc. Even when two different stakeholders use the same source (say, standard or meeting minutes) they may interpret it differently. Often the difference in interpretation is the cause of conflicts between stakeholders, even in a cooperative setting. Therefore it is important to capture this information to provide clarification. ("You can look at a given set of requirements and they might be meaningless if you don't

really know what the customer needs because you can interpret them wrong, or at least two or three different ways.")

The source of the requirement oftentimes is not its originator, but may be a intermediary who makes the interpretation, and therefore provides another twist to the requirements. It is important to capture information about this intermediary to help identify the history of requirements.

Another reason to have the source of the requirement is to provide accountability. This provides the stakeholders with information about how the requirements and design have evolved, what changes have been made, why and how these were made, and the status of their development. ("If you don't have requirements traceability to link back to that high level requirement, it's going to be very difficult to give the customer an impact statement of that change.")

C. DESIGN/ALLOCATION

In this study of traceability we have looked at the whole systems development lifecycle. ("For the system to work you have the hardware issue, software issues, system issues, everything has to work together.") The Design/ Allocation Model is presented in Figure 2 and Figure 3.

1. **Design/Implementation**

We view design as any activity that creates artifacts. The implementation process may be seen as a lower level design activity. Further, design defines the system, while implementation creates it. Therefore, the traditional distinction between design and implementation is not made.

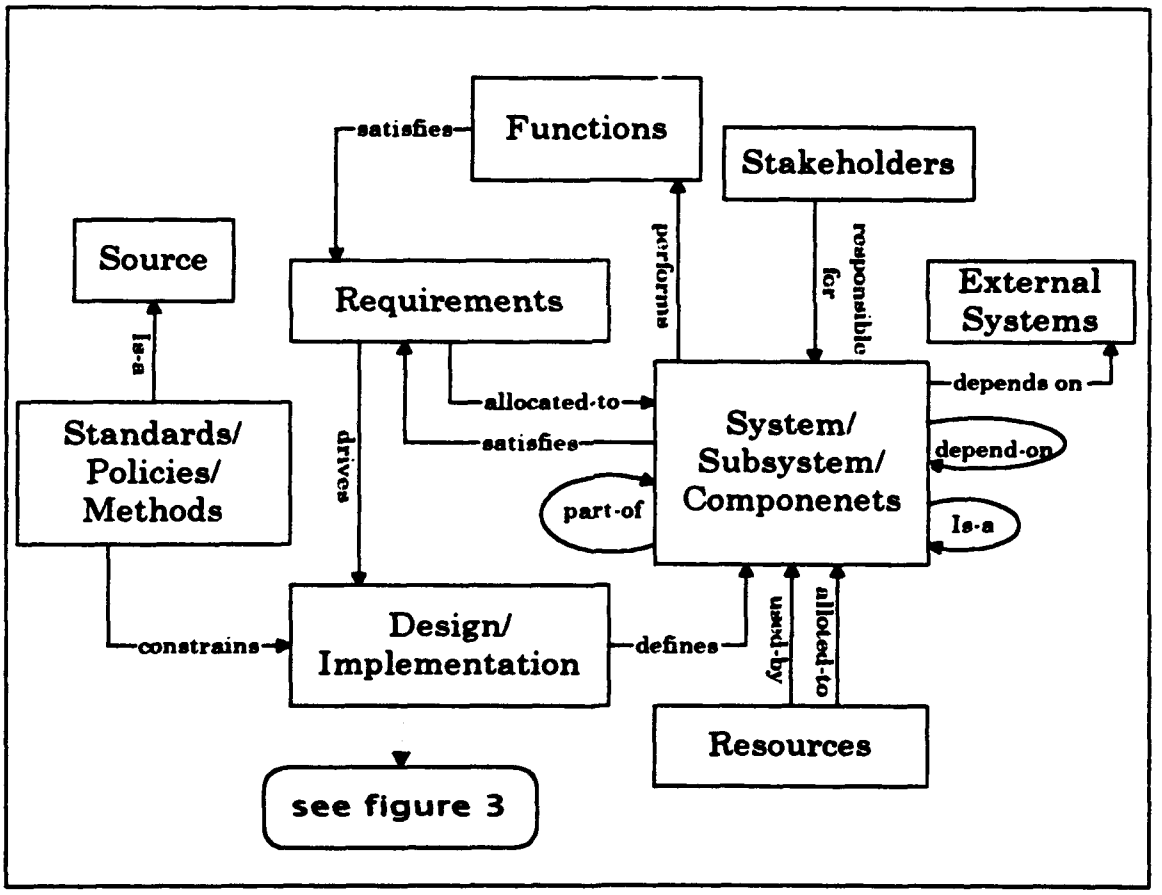

Figure 2: Design Allocation Model

REQUIREMENTS **DRIVE** *DESIGN/IMPLEMENTATION*. These are **CONSTRAINED** by *STANDARDS*; e.g., military standards, policies, procedures, methods, etc. These constraints may be documented in various sources (as noted by the **IS-A** hierarchy).

2. System/Subsystem/Components

SYSTEM/SUBSYSTEM/COMPONENTS (here on referred to as *COMPONENTS*) are the building blocks of the system. They are **DEFINED** by the *DESIGN/IMPLEMENTATION* process. *REQUIREMENTS* are **ALLOCATED-TO** *COMPONENTS*. The components could be anything, a piece of hardware, software, humanware, etc. ("At the allocation step you're going to allocate to people, to processors, to memory, to hardware, etc.")

COMPONENTS **DEPEND-ON** other components, in that the performance of one may depend on another. Some of these dependencies are more obvious than others. Those that actually interface with other components, say actually talk to each other, have explicit dependencies and are easier to recognize than those that do not directly interface. For instance, an operating system on one component that may affect the choice of hardware on another even though there may be no direct interfaces between them. This information is needed so that when a component is modified, deleted, and/or added than there is a way of determining how the system is affected by these dependencies. **PART-OF** maps the

components that are pieces of a larger component which, when put together, make a whole. This information allows the designers to know inter-component dependencies. ("As I change a part or break it into sub-parts, I would like to have some sort of traceability or mesh structure which gives some sort of relationships...I would like to know what is the characteristic of each component.") Finally, an **IS-A** hierarchy specifies that components are decomposed at different levels; e.g., from system to subsystem to item and to unit.

The subjects of our research stressed that identifying *STAKEHOLDERS* **RESPONSIBLE-FOR** the various *COMPONENTS* of the system was important to ensure accountability. This information provides an ownership to the system components that satisfy specific requirements. Here, ownership does not refer to the originator as a requirement, but the person who is responsible for satisfying it. ("If I've got a high level requirement and I'm going to allocate that down to the next level of specification then I have assigned ownership for at least the satisfying of at least part of that requirement to that lower level specification. And I would expect that whoever owned that specification to take responsibility for that allocation.") ("The customer gives us requirements to do a program or a system, then we take those requirements and

parse them out to the components of the system. Then that group can look at that component in the same way we look at the system.")

It is also important to track the allocation, distribution and utilization of resources. *RESOURCES* are **USED-BY** and **ALLOTED-TO** *COMPONENTS*. This allows the actual utilization of resources to be tracked. ("At the allocation level there needs to be parameters that associate budgets of resources that are going to be consumed...and then you're going to have to test to see that you're somehow compatible with, complying with and not grossly out of bounds with your budgeting.") ("I want traceability to give me some level of assurance of particular steps in the design process that I'm meeting or likely to meet system goals or to help me to allocate resources or capabilities in such a way that I maximize the likelihood of meeting system goals.")

RESOURCES are allocated in line with *CSFs*, such as time, money, weight, criticality, long term maintainability, etc. ("If cost happens to be the major system driver then what I am really interested in is keeping track of actual or supposed cost of the components. If on the other hand memory space is the primary driver they might not really pay as much attention to cost, as how my decisions impact the allocation of memory space.")

Our subjects also mentioned that it was important to understand what *FUNCTIONS* the *COMPONENTS* **PERFORM** and how these **SATISFIED** *REQUIREMENTS*. The *COMPONENTS* work together to **SATISFY** the non-functional requirements also. A component may partially **SATISFY** requirements, and as the design progresses, requirements must be fully satisfied.

3. External Systems

The *COMPONENTS* **DEPEND-ON** *EXTERNAL SYSTEMS* in that they often interfaces with other systems. As already discussed, direct interfaces are easier to recognize than those that have ambiguous dependencies. However, information about all interfaces is important to maintain. ("What's happening now is that instead of being a single quantity, a single product line, the change in technology that we see is bringing about a new paradigm...a particular product line is going to be brought on board a ship and it's going to have to plug into an already existing system...the things they need to know is what is the functionality that the system needs to have, what is the demand on these services that they would have and the range of the demand.")

4. Design/Allocation Decisions

Similar to the requirements model presented in Figure 1, there is a decision process for design and allocation. This is presented in Figure 3. The components and links of this process are the same as that presented in

the Requirements Management Model and the discussion presented in paragraph III.A.6 is applicable, except that they relate to *DESIGN/IMPLEMENTATION*. ("Traceability would come in handy for determining the reasons for your design.") ("That's the design knowledge capture that we were supposed to maintain. What is basically here is the design the way it is and here's the decisions, the thought that went into that deci-sion. And then when you changed them, here's how we changed them.") ("Another use of traceability in design is allocating functions in such a way that you're sure that

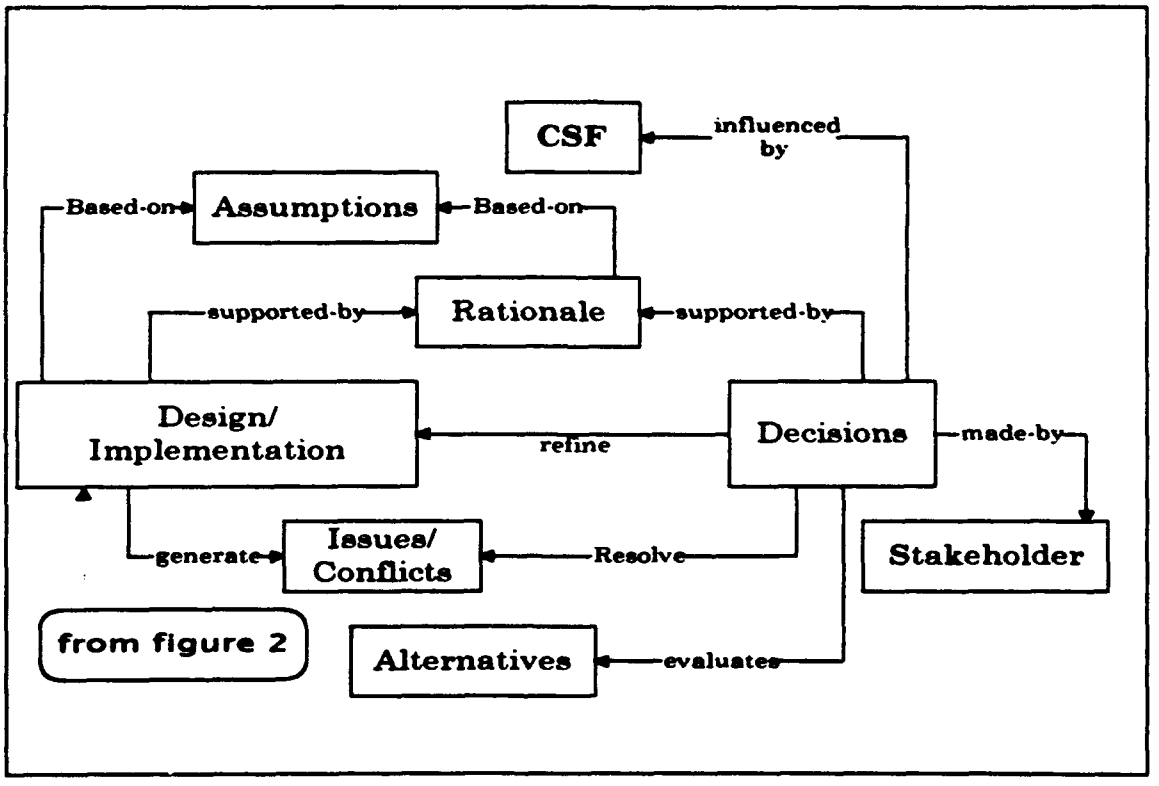

Figure 3: Design/Implementation Decision Making Model

they support each other and don't interfere with each other.")

D. COMPLIANCE VERIFICATION

Throughout our research it became very obvious that the "testing" phase of systems development was affected by all requirements. Each requirement usually has a set of "testing" parameters associated with it that are used to determine whether they have been satisfied by the system. The Compliance Verification Model (Figure 4) is discussed below.

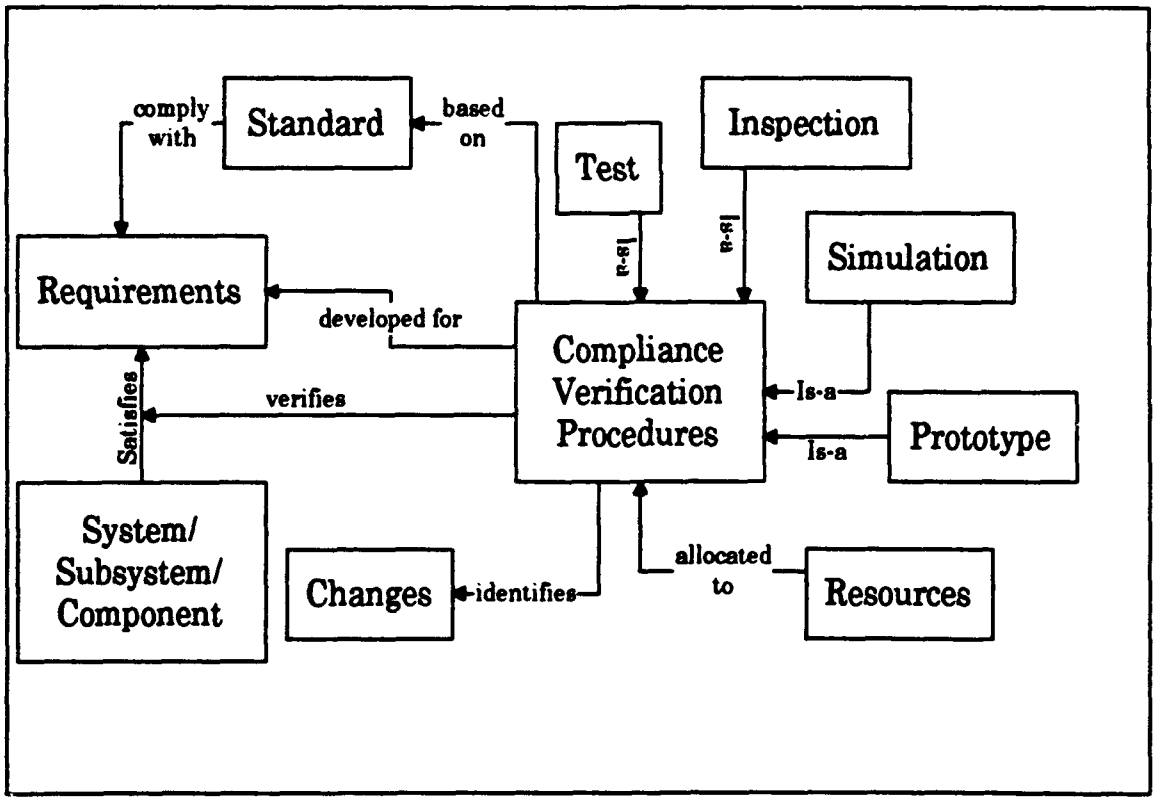

Figure 4: Compliance Verification Model

1. Compliance Verification Procedures

Procedures for verifying the compliance of the system components with requirements are defined; the plans, approach, and methods are described. *TEST*, *INSPECTION*, *SIMULATION*, and *PROTOTYPE* are all *COMPLIANCE VERIFICATION PROCEDURES* (here on referred to as CVP) as noted by the **IS-A** link. ("We write the test specifications by taking apart the SRS and Component Specifications.") ("You've got to be able to specify what you're going to test to. You've got to be able to calibrate in terms of if the performance is what you need.") These *CVPs* are **DEVELOPED-FOR** each *REQUIREMENT*. ("Once you determine what requirements need to be tested it requires test procedures. If something is untestable or we elect not to test, then it becomes no longer a requirement or at least we all agree that it is unverifiable.")

The utilization, assignment and availability of *RESOURCES* is important to testing and they are **ALLOCATED-TO** *CVPs*. This often determines whether one can even perform a compliance test, or if the cost-benefit analysis of performing it justifies it. CVPs need resources so that quality systems are produced. But, if CVP resources decrease, resources for design may increase. There is a tradeoff between the resources allocated to design and CVPs that must be made in order to optimize utilization to attain the best system possible.

2. Standards/Procedures/Methods

STANDARDS, military or otherwise, affect testing procedures in that they provide the **BASIS** for *CVPs* and determine which CVPs are required and how they are to be performed. Often, *REQUIREMENTS* are written so that they **COMPLY-WITH** *STANDARDS*. It is important that the CVPs for a requirement use the same interpretation of the standards as were used while developing requirements. ("There is a lack of a way to firmly assure that the tests that you are generating really reflect the same thing as the standard.") ("So if something meets the test it may not meet the standard and vice versa.")

3. Completeness and Accuracy

The *CVPs*, when performed, produce test results that either **VERIFY** how the *COMPONENTS* are functioning or **IDENTIFIES** deficiencies (*CHANGES*) that need to be resolved. *COMPONENTS* **SATISFY** *REQUIREMENTS* and the results of *CVPs* **VERIFY** that components satisfy them. This whole model is used to certify completeness and correctness of the system and identify changes that may be necessary to meet the objectives. ("As I change my design, my design continues to support my objectives.")

IV. FOCUS GROUP ISSUES

This chapter examines the results from the analysis of data collected by focus group research. While the results of research provide a framework for a model of traceability, many issues remain. This discussion centers around those issues that need to be addressed while implementing a comprehensive traceability scheme.

A. CUSTOMER INVOLVEMENT IN SYSTEM DEVELOPMENT

The customer is an invaluable source of information in the system design process and should be included as an integral member of the development team. This is because they are the final authority on completeness, accuracy, and quality. Therefore, customer involvement is key in developing requirements specifications which clearly describe their needs. In the development of many systems, the customers present their needs and intent, in the form of requirements, to the designer. The designer then develops a product which all too often must go through several reworks before it is accepted by the customer. This occurs because customer feedback is limited in the various walkthroughs and reviews of the development process.

In addition to reviews, the customer is also important when modeling is used, because ("the customer and the

contractor have to agree on what the model will tell us").[1] Some organizations have incorporated the customer into every aspect of the development process with much success. ("They've (the customers) been there as the requirements are written, they've given their interpretations, they've put them in the context of the operational environment.")

To make the most from including the customer in the process, traceability can capture the requirements rationale (the why) of the customers requirements at the inception of the project. Furthermore, an effective traceability scheme will also include the identification of critical requirements. This will make even the limited opportunities for feedback more effective by allowing the customer to concentrate on those critical success factors that need to be looked at. Traceability will promote customer relations by providing important information to the customer in a timely, organized fashion. This can assist the customer in conducting tradeoffs and cost benefit analysis when trying to assess the impact of changes.

B. TALENTED PEOPLE NEEDED ON PROJECTS

One of the key elements to success in any project are the people who work on it. Not only is there a need for people who are highly skilled and able to understand the

[1] This is a direct quote from a subject participating in a focus group. Henceforth, all quotes from a subject will be enclosed in parentheses and quotation marks, but no specific reference will be made.

complexity of the project but there is also the need for personnel who will stay with a project over time. The quality of traceability information in a system is dependent on the quality of the people producing it.

Systems have become so vast in size and scope that they can no longer be designed or managed by one entity. As a result, systems are divided into independent components and developed separately. To ensure that these components do not develop too independently of each other it is absolutely essential that the people involved with the system be knowledgeable of their component, other components, and the larger, parent system which they compose.

Traceability serves as an excellent means of augmenting the skills and knowledge of the people associated with a project. It captures what is happening in a process and serves to compile the history of the project. Some important aspects of the expertise of project teams can be captured by traceability information such as Design Rationale. On the other hand, there are limitations to how much and in what detail this information can be captured for traceability cannot capture what goes on in the mind of a designer. It can, however, capture decisions and assumptions as well as provide a link to the identity of the person who made them.

In addition to providing linkages, traceability can also be used as a mechanism to facilitate information exchange among project participants. Establishing linkages as well as capturing information enable stakeholders timely access to critical information and therefore speeds up the decision making process.

C. STAKEHOLDERS HAVE DIFFERENT INFORMATION NEEDS

Every requirement in a system originates from some stakeholder and each of these stakeholders have varying needs and desires from traceability. One of the desirable features of traceability is its ability to facilitate information sharing among stakeholders. Throughout the hierarchy of the organization each stakeholder has information needs that are unique to their level of responsibility. ("On the top level you think about completeness and correctness and on the bottom level you think about resource utilization.") Some of these information needs include, sources, assumptions; in short, whose need is being met and why.

As well as identifying their information needs, stakeholders should also be identified by the requirements they are associated with. This allows changes to be evaluated by the concerned stakeholders. Furthermore, identifying stakeholders will enable designers to determine the validity of a requirement if its "owner" changes.

Traceability should, at a minimum, be able to capture why a requirement exists, who made that decision, what additional requirements are related to that requirement, and which stakeholders and system components are affected by the requirement. The level of granularity of information captured by traceability needs to be such that each stakeholder has full access to the information they require.

D. PROBLEMS WITH LANGUAGE INTERPRETATION.

There needs to be a formal language which stakeholders can use to communicate with each other. This is because ("the first requirements design review between the government and the contractor is to understand what the requirements are"). System designers are constantly faced with the task of accurately translating the customers' objectives and needs into system requirements. ("Part of the problem is to write a functional specification and to ensure that when the contractor implements it he builds a system that meets the functional specification.")

Problems like these stem from the difficulty involved with translating English text in such a way as to capture the intent of the customer. Because the human mind is able to reason and judge, interpretation of English text may be solely individual. ("The problem with traceability here, is there is no way to map unambiguously from English statements, which is the way standards are typically

written, to any other language.") By having no standard method for translating text into unambiguous requirements, it becomes difficult for the designers to clearly interpret the customers needs and expectations.

To assist in translating requirements, traceability should be able to relate the textual description of a requirement, by the customer, to the actual system requirement, developed by the designer, and to subsequent derived requirements throughout the system lifecycle. For traceability to be fully functional, there must be a consistent protocol for capturing information from English language text. Then it must be clear that the language be used consistently throughout the system design, for example, so that tests and standards say the same thing.

As one means of translating requirements, traceability tools use a variety of methods, such as identifying key words, for capturing information contained in English text. However, these tools are subject to unique language interpretations which affect the type and intent of the data and information they capture.

E. RESOURCE CONSTRAINTS

Always an issue is the allocation and budgeting of resources and their impact on the overall system. Time, funding, and available manpower have a great impact on any project. ("It takes time to document information and to

actually sit down and document all that information takes a great deal of time.") As a non-functional requirement, traceability tends to slip to the end of the project life-cycle and is one of the first things to be eliminated or scaled down when a funding crunch comes along. Because, ("if nobody pays you to document and trace then you don't do it").

As a compromise, in lieu of elimination, traceability is often poorly conducted and this is attributed to a lack of available time, money, and personnel. As a result, what information is captured may prove to be no longer timely or adequate. This has a negative impact because timely and accurate information are crucial to the decision making process.

Cutting traceability in response to resource constraints may prove to be a poor choice. When you lose traceability you also lose some management decision aids, such as the ability to perform impact analysis. Traceability provides the auditing framework necessary for monitoring resource allocation and use. Therefore, it can be of benefit during resource cutbacks by providing a means of identifying those things that need to be cut while preserving those identified as critical success factors. This information will enable stakeholders to carefully weigh alternatives as they implement budget cuts and reallocate resources.

F. MISSION CRITICALITY

("Today the requirements are evolving on almost a daily basis because the threat is evolving.") As a result, developing systems need to be dynamic and responsive in order to comply with constantly changing threat, missions, and technologies. To keep pace with today's volatile climate, there needs to be prioritization of requirements. The key to prioritizing is the identification of mission critical requirements. These are the requirements that are essential to the project and cannot be eliminated. Such requirements are found throughout the system hierarchy and have linkages which extend across all levels of the system. Consequently, the identification of and relationships between mission critical requirements needs to be captured in order for the effects of changes and cutbacks to be accurately assessed by stakeholders through cost benefit analysis.

In today's world change is no longer trivial and can quickly alter the basic nature of a system. Complete traceability will help to manage the evolution by enabling designers to quickly assess the impact of changing requirements throughout the system. To support a dynamic system, traceability must provide a means for capturing the source and rationale behind changes.

Traceability allows system engineers to identify and prioritize mission critical requirements and trace their

linkages throughout the system lifecycle. This enables quick assessments of the impact of change on mission critical requirements. Complete traceability that identifies critical requirements enables stakeholders to salvage what they need from a rapidly changing system with minimal loss.

G. DERIVED REQUIREMENTS

System designers need a method to identify both derived requirements and their subsequent linkages throughout the system hierarchy. ("They (requirements) are refined, and become more specific in detail and in fact are more demanding than the original requirement might have been"). What is unique about derived requirements is their origin, they are not found in the original specifications. Derived requirements tend to evolve in response to design constraints and implicit assumptions made as part of the design solution. Because they are based on interpretation, derived requirements tend to be dynamic and should therefore be closely monitored.

Traceability enables system designers to identify and capture the source and rationale contributing to a derived requirement. This information is important because derived requirements are usually not explicitly specified by the customer and therefore, ("those are the ones we can revisit and change").

H. COMPATIBILITY BETWEEN TOOLS

The use of tools to conduct traceability is becoming more popular with contractors and customers alike. Traceability tools are a big investment for organizations. And many organizations spend large amounts of time and money developing and implementing traceability tools to better serve the needs of their customers. The problem arises when customers and contractors rely on different tools to conduct traceability. Contractors feel the crunch because they deal with ("a wide variety of customers and those customers want outputs in their set of tools"). Hence there is a need for compatibility among the available traceability tools.

To ensure compatibility, traceability tools should focus on capturing information, in a similar format, to a common database. Differences in tools should not be in the type of data, but rather the manner in which data is manipulated and presented. Stakeholders want a versatile tool because, ("if you have a really good, robust database, or engineering tool and you can get data out, hard copy in different formats you're well ahead. We want a requirement stored in one place to go out in as many documents as it needs to").

I. CAPTURE OF INFORMATION

How to capture information as the system development process evolves is a great challenge facing systems designers. Traceability information must be captured by the

people doing the design rather than someone else whose job is to create the documentation. The best way is to capture information as part of the development process, for then, information is captured as it is created. The problem here is, it's very difficult to get the system engineers to ("sit down long enough to really think about what it is they want because a lot of cases they'll walk into the simulator and think of something new while they're flying"). Keeping in mind that information is only as good as the people creating it want to make it, an incentive mechanism should be built into systems. Designers must feel that complete traceability is necessary because of either some type of reward or penalty.

To capture the information created by the designer requires a continual process. This process should facilitate the capture of information on-line, from the source, functionally organizing it, reviewing and sharing the information, and verifying that the stakeholders information needs have been met. Using traceability enables you to look at all aspects of a decision and examine all the areas affected by it.

J. FUNDING PROFILE OF PROJECTS

In many projects today, especially in DoD, the funding provided for system development is apportioned in phases. As part of this funding profile, initial funds for

requirements management and design tend to be rather limited with the bulk of the funds allocated to latter phases of development. This is difficult for the developer who must define and identify requirements as well as establish a scheme for traceability information capture in the early stages of the system lifecycle.

A traceability scheme, with its high startup costs, becomes a problem when only limited funding is available at the start of a project. Traceability, to be effective, must be comprehensive in the early stages of a project. Project sponsors should realize that without adequate funds to support it, a complete traceability scheme cannot be implemented at the start of the project.

K. LIVING TRACEABILITY DOCUMENT

Traceability information is not something that is captured at one point in the system lifecycle and stays stable. On the contrary, a good traceability scheme is one that is constantly updated throughout the lifecycle of the system. Information needs to be captured as early as possible for use later in the project.

To capture information, many organizations use daily notebooks or unit folders which serve as a repository for all information concerning a project. These are maintained by the individual or manager and contain documentation reflecting decisions and changes as they occur within the

system development process. Such information should be captured and maintained in a more formal traceability environment.

L. EMBEDDED ASSUMPTIONS

As designers refine requirements in the system development process there are assumptions which they make unknowingly. These assumptions are not explicit, they may be obvious to the person developing them but not to others. ("Even if you write it down you might make some unstated assumptions that over time might change. And the other person comes back and looks for that information, they could be working on a totally new set of assumptions.") Even if you have a traceability model, you may not capture all the required information as some stakeholders may not even realize that some information obvious to them may not be obvious to others.

To counter this problem of capturing assumptions, there needs to exist a detailed model of what information needs to be captured. The more detailed the traceability model, the better chance of capturing the true intent and assumptions behind a decision.

M. ACCOUNTABILITY

Some decisions and requirements have little technical or functional justification. Because they are of this type, it

is important for system designers to capture the information contained in their rationale and assumptions. Designers need to identify the stakeholder responsible for the requirement, because sometimes you can not explicitly capture all the information. This is because; for example, the need for the requirement may go away if the stakeholder leaves the project.

N. ACCOUNTABILITY VERSUS PERFORMANCE APPRAISAL

Traceability helps to determine who is accountable for requirements by identifying stakeholders and the rationale they used for developing requirements. ("I think you need to be sensitive to it (traceability) when you conceptualize your requirements so you know why they are requirements and so that you know as you are writing them that you are going to be accountable down the road to see that those can actually be carried out.") As a word of caution though, traceability should not be used for performance evaluation. Rather it should be used as a way to understand and correct mistakes. To prevent the fear of being held culpable, a good idea is to evaluate decisions and make them as a group.

O. LEGAL/PROPRIETARY

Certain design procedures may contain information that the developer may feel is confidential/proprietary. The question then arises, if the designer uses traceability to

capture their design rationale on a project, are they required to provide that information to the customer who funded the project?

Traceability provides the mechanism of ensuring that the customer is getting what they have paid for. However, the idea of full disclosure of information by the designer is one that many feel is on the verge of revealing trade secrets. The requirement to divulge traceability information is something that should be included in the contract negotiation process and issues of proprietary information must be sorted out.

V. CONCLUSION AND RECOMMENDATIONS

A. MODEL

Department of Defense Standard 2167A requires that requirements traceability be conducted during systems development. However, this standard (and other standards that require traceability) offer no comprehensive model for use by developers that states what information should be captured as a part of a traceability scheme. Our research developed a model of traceability based on the information needs of various stakeholders in systems development. This model, as described in Chapter III, provides a basic framework for use in conducting traceability.

B. IMPLEMENTATION

To successfully implement requirements traceability in systems development, it needs to be integrated into the systems development process. A systems development methodology must recognize that the capture and use of requirements traceability information must be done as the system is created and maintained. Much of the traceability information can be automatically captured if properly supported by tools. CASE tools that automatically capture as the system evolves are essential.

C. DEVELOPMENT OF MODEL COMPONENTS

While the proposed model provides a framework for conducting requirements traceability, there are component parts which require further refinement. For instance, a comprehensive design rationale model, such as REMAP, with support mechanisms may be used to include some of our model primitives. Depending on the project needs, components that concentrate on specific issues such as reliability and/or change management may be developed. By elaborating on each of these processes, traceability will provide more accurate and useful information to stakeholders.

D. DEVELOPMENT OF MECHANISMS

Combined with our model, which outlines what information to capture, there needs to be mechanisms which specify how information can be combined and used by stakeholders. An example would be a mechanism which identifies how critical success factors can be tracked through various system components to support cost benefit analysis. Mechanisms that facilitate reasoning with traceability information should be developed to support various stakeholder needs.

E. LIVING (EVOLVING) DOCUMENT

Current practices of producing static traceability documents that become obsolete almost as they are created is wasteful. As a system develops through its lifecycle, the

traceability documentation should constantly evolve to reflect the current status of the system. Without an existing requirements traceability scheme during the entire lifecycle of the system, certain key pieces of information may not be captured. This results in a breakdown of the traceability scheme and the information available becomes useless.

F. LIFECYCLE COST BENEFIT

While required, traceability is all to often not included into the funding profile of a project, and in those where funding is available, it is rarely sufficient. Furthermore, funds for projects are allocated by phase with the initial phases receiving the smallest allocation. Requirements traceability can reduce system costs by enhancing decision making and streamlining the development process. Therefore, requirements traceability should be included in a project's funding profile by looking at the benefits it provides throughout the entire lifecycle.

G. SUGGESTIONS FOR FUTURE RESEARCH

Follow-on research to this thesis should include:

- Validation of our Requirements Traceability Model using focus groups consisting of stakeholders that are familiar with traceability.
- Personal interviews and additional focus groups to refine and expand component processes of our Requirements Traceability Model.

- Exploration of tools and techniques to resolve issues raised in the development of a model for requirements traceability.

LIST OF REFERENCES

Andrews, Betty J., "Hierarchical Progression Analysis (HPA)," *Information Processing & Management*; Vol. 20, No. 1-2, 1984.

Baldo, J., "Reuse in Practice Workshop Summary," Institute for Defense Analyses, April 1990.

Brown, B. J., "Assurance of Software Quality," SEI Curriculum Module SEI-CM-7-1.1 (Preliminary), Carnegie Mellon University, Software Engineering Institute, July 1987.

Cadre Technologies, Inc., *Teamwork/RqT Primer*, by C. Kelley, 1990.

Cordes, D. W. and Carver, D. L., "Evaluation method for user requirements documents," *Information and Software Technology*, v. 31, no. 4, May 1989.

Dorfman, M., and Flynn, R. F., "Arts -- An Automated Requirements Traceability System," *The Journal of Systems and Software 4*, 1984

Edwards, M. and Howell, S., "A Methodology for Systems Requirements Specification and Traceability for Large Real-Time Complex Systems," Technical Report, Naval Surface Warfare Center, August 1992.

Fiksel, Joseph Dr., "New Requirements Management Software Supports Concurrent Engineering" with Requirements Management Survey, CimFlex Teknowledge Corporation, Washington, DC, June 2, 1992.

Finkelstein, Anthony, "Tracing Back From Requirements," in proceedings of the Colloquium by the Institution of Electrical Engineers Professional Group C1 (Software Engineering), London, UK, 2 December 1991.

Gathmann, T. and Halker. D., "Towards a Manageable Solution to the Iterative Development of Embedded Knowledge-Based Systems," Rockwell International Corp., September 1990.

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

Hamilton, V. L., and Beeby, M. L., "Issues of traceability in integrating tools," in proceedings of the Colloquium by the Institution of Electrical Engineers Professional Group C1 (Software Engineering), London, UK, 2 December 1991.

Horowitz, E., and Williamson, R. C., "SODOS: A Software Documentation Support Environment -- Its Definition," *IEEE Transactions on Software Engineering*, Vol. 12, August 1986.

Horowitz, E., and Williamson, R. C., "SODOS: A Software Documentation Support Environment -- Its Use," *IEEE Transactions on Software Engineering*, Vol. SE-12, November 1986.

Jackson, Justin, "A Keyphrase Based Traceability Scheme," in proceedings of the Colloquium by the Institution of Electrical Engineers Professional Group C1 (Software Engineering), London, UK, 2 December 1991.

Keuffel, W., "Extra Time Saves Extra Money," *Computer Language*, December 1990.

Liu, Lung-Chun, and Horowitz, E., "A Formal Model for Software Project Management," *IEEE Transaction on Software Engineering*, Vol. 15, No. 10, October 1989.

Macmillan, J., and Vosburgh, J. R., "Software Quality Indicators," *Scientific Systems, Inc.*, September 1986.

Marconi Systems Technology, *RTM Requirements and Traceability Management* (Product Overview), Arlington, VA, 1991.

Marconi Systems Technology, *RTM Requirements and Traceability Management*, (User Course Notes), Arlington, VA, July 1991.

McCausland, C. D., "A Case Study in Traceability," in proceedings of the Colloquium by the Institution of Electrical Engineers Professional Group C1 (Software Engineering), London, UK, 2 December 1991.

Military Standard, *Defense System Software Development*, DoD-STD-2167A, 29 February 1988.

Murine, Gerald E., "Secure Software's Impact on Reliability," *Computers and Security*, Vol. 5, 1986.

Naval Postgraduate School, Technical Report NPS-AS-92-022, *An Initial Model of Requirements Traceability: An Empirical Study*, Balasubramaniam Ramesh, Ann Abbott, Mona Busch, and Michael Edwards, 1992.

Naval Postgraduate School, Technical Report NPS-54-82-005, *Useability of Military Standards for the Maintenance of Embedded Computer Software*, Norman F. Schneidewind, June 1982.

Ramesh, B., and Dhar, V., "Supporting Systems Development using Knowledge Captured During Requirements Engineering," submitted to IEEE Transactions on Software Engineering, June 1992.

Ramesh, B., and Edwards, M., "Issues in the Development of a Requirements Traceability Model," in proceedings of *IEEE International Symposium on Requirements Engineering*, San Diego, CA, January 1993.

Roetzheim, W. H., *Developing Software to Government Standards*, Prentice Hall, Englewood Cliffs, NJ, 1991.

Smithers, R., Tang, M. X., and Tomes, N., "The Maintenance of Design History in AI-based Design," in proceedings of the Colloquium by the Institution of Electrical Engineers Professional Group C1 (Software Engineering), London, UK, 2 December 1991.

Stehle, G., "Requirements Traceability for Real Time Systems," in proceedings of EuroCASE II, London, UK, 2 April 1990.

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

Walters, N., "Requirements Specifications for ADA software under DoD-STD-2167A," *J. Systems Software*, 1991

West, M., "The Use of Quality Function Deployment in Software Development," in proceedings of the Colloquium by the Institution of Electrical Engineers Professional Group C1 (Software Engineering), London, UK, 2 December 1991.

Wright, Simon, "Requirements Traceability -- What? Why? and How?," in proceedings of the Colloquium by the Institution of Electrical Engineers Professional Group C1 (Software Engineering), London, UK, 2 December 1991.

BIBLIOGRAPHY

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

Alford, M., "Strengthening the Systems Engineering Process," Ascent Logic Corp., San Jose, CA., in proceedings of NCOSE, October, 1991.

Cooke, J., and Stone, R., "A Formal Development Framework and Its Use to Manage Software Production," ...

Fischer, Aaron, "CASE Tool Gets Friendly Enhancements," *Electronic Engineering Times*, 12 September 1988.

Howard, S. G., *"Requirements and Traceability Management,"* in proceedings of the Colloquium by the Institution of Electrical Engineers Professional Group C1 (Software Engineering), London, UK, 2 December 1991.

Keys, Ellen, "A Workbench Providing Traceability in Real-Time System Development," in proceedings of the Colloquium by the Institution of Electrical Engineers Professional Group C1 (Software Engineering), London, UK, 2 December 1991.

Krueger, R. A., *Focus Groups: A Practical Guide for Applied Research*, Sage publicatoins, Inc., 1988.

Masselle, Eric L., and Rhodes, Donna H., "Mechanized Mechanisms, The Case for Process Enactment," published by IBM Federal Systems Company, 1993.

"Military Fine Tuning Urges Development Standards," *Software News*, November 1980.

Morgan, D. L., *Focus Groups as Qualitative Research*, Sage Publications, Inc., 1988.

Nejmeh, Brian A., "Designs On Case", *UNIX Review*, Vol. 6 No. 11, date unknown.

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

Oman, P. W., Cook, C. R., "Design and Code Traceability Using a PDL Metrics Tool," *J. Systems Software*, 1990.

Schindler, Max, "Software Engineering: Still a New Frontier," *Electronic Design*, August 6, 1987.

Shepperd, M., and Ince, D., "Multi-Dimensional Modeling and Measurement of Software Designs," in proceedings of the 1990 ACM Computer Science Conference, Washington DC., February 1990.

Stewart, D. W., and Shamdasani, P. N., *Focus Groups: Theory and Practice*, Sage Publications, Inc., 1990.

"System Hierarchy, Allocation of Requirements, Flowdown, Traceability, Interface Definition," in proceedings of the Software Productivity Consortium, Herndon, VA, 1989.

Templeton, J. F., *Focus Groups: A Guide for Marketing and Advertising Professionals*, Probus publishing Company, 1987.

Wuebker, F. E., "The Impact of Nebula, MCF, and ADA on Real-Time Embedded Computer Systems," RCA Government Systems Div., November 1982.

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.	Department of Administrative Sciences Attn: Code AS/RA (Prof. B. Ramesh) Naval Postgraduate School Monterey, California 93943-5000	10
4.	Department of Administrative Sciences Attn: Code AS/BD (Prof. T. Bui) Naval Postgraduate School Monterey, California 93943-5000	1
5.	CPT Gale A. Harrington, USA Box 625 Hemphill, Texas 75948	2
6.	Commander Fast Sealift Squadron ONE 4400 Dauphine St. Bldg. 601-4A New Orleans, Louisiana 701446 Attn: LCDR K. M. Rondeau, USN	2