Towards Value-Based Requirements Traceability

Thesis

By

Grant Zemont

DePaul University
Chicago Illinois
March 2005

Table of Contents

List of Figures

Acknowledgements

I would like to personally thank Dr. Jane Cleland-Huang for helping me identify the topic in the first place, and working with me through its many iterations. Her ideas and direction were of great value since I had no prior experience with requirements traceability.

It seemed like a long road to complete this paper. During its development I experienced two job changes (one layoff), health problems, extensive employment-related travel, a professional project management certification exam to study for and pass, as well as numerous missed family commitments to perform research and write. Therefore, I cannot express my gratitude enough to my children and especially my wife Michelle for their patience, understanding, and support through the 18 months it took me to develop this thesis.

CHAPTER ONE: Abstract

Over the past decade, several studies have been performed that show the importance of requirements traceability (RT) for supporting critical system development activities. While much attention has been focused on how RT should be performed from a technical and process-oriented perspective, little effort has been made to explore the financial benefits and trade-offs of implementing RT as an integral and value-enhancing component of a software development project. In fact, several studies have shown that many IT organizations do not practice RT and therefore must perform exhaustive searches through requirements, design, and code artifacts in order to understand the impact of a proposed change. Other organizations implement basic traceability practices, perceiving them as necessary overhead, but failing to recognize RT as a value enhancing activity. Only a few organizations have a more mature view of RT and understand its value for validating requirements or supporting impact analysis during change management.

Using traceability on projects and using it effectively are two different things. Since 1994, the Standish Group Chaos Report, which as an annual survey of the successes and failures on IT projects, has identified requirements-related problems as several of the leading causes of project failure. While recognizing the importance of the requirements process, many organizations fail to effectively implement RT practices, therefore suggesting that there is a major disconnect between proper or effective utilization of RT and realizing and capitalizing the benefits that it can provide. Traceability affects all the requirements of a system and the subsequent design, code, and tests that are derived from them. It affects all stakeholders in the RT process such as customers, management, developers, and testers. Therefore, RT cannot be simply viewed as an activity to meet a government standard or as an ancillary process to the software engineering domain. As requirement-related practices are crucial to project success and RT supports the requirements process, it is important to understand the actual value that RT can provide both to an individual project and to an organization as a whole, so that informed and business oriented decisions can be made.

This research introduces a framework for assessing the value that RT can provide to an organization in order to support decisions related to the implementation of RT. The Value Based Software Engineering (VBSE) framework, initially proposed and developed by Barry Boehm, is applied to the RT problem. By using the seven facets of this framework in combination with a blend of manual and dynamic traceability techniques, it is posited that an organization can determine how to exploit the implementation of RT practices to maximize both project and organizational value in order to increase the chance of project success. The managed approach that is proposed through this research is called Value-Based Requirements Traceability (VBRT).

CHAPTER TWO: Introduction

In 1994 and 2004, the Standish Group, which is an independent research group, published their Chaos Reports [Standish]. These reports are surveys of the IT industry and cover the success and failure factors of software development projects. Since 1994, the Standish Group has surveyed 13,522 companies and has classified over 40,000 projects from those companies into three categories:

- Successful projects - Those that have been delivered on time, within budget, and having most if not all features implemented that were initially specified.
- Challenged projects – Those that are operational and have been delivered, but were late, over budget, or missing many initially specified features.
- Failed projects – The project was cancelled sometime during the course of the project.

Based on these definitions, the table below shows the average results of the Chaos Report from 1994 compared to the 2004 report:

Table 1-1. CHAOS Report summary

Project Type	Chaos Report 1994	Chaos Report 2004
Successful	16%	34%
Challenged	31%	51%
Failed	53%	15%

Based on this general data, there are some interesting observations over this 10-year time period. Since 1994, the top reasons cited by companies that projects were challenged or failed centered on poor requirements engineering. More specifically, these reasons were articulated as lack of user involvement, incomplete requirements specifications, and changing requirements specifications. A further disturbing trend is that for challenged projects in 1994, an average of 61% of the defined requirements made it into a final system. In 2004, that number shrunk to only 54%! [Standish]

It is a given that during the software development activity, no matter if an organization follows agile methods, iterative methods, or a waterfall approach, eliciting requirements from stakeholders is a critical activity. Theoretically, resources should not even be committed to a project without knowing the technical and business details of the system, even from a high-level [IEEE 04, PMI 00]. The technical and business details of a system are usually captured in the form of requirements. But what is a requirement? As put forth by the *IEEE Software Engineering Body of Knowledge* (SWEBOK), a requirement is *"a property that a system must exhibit in order for it to adequately perform its function"* [IEEE 04]. The set of activities that surround requirement-related processes are called Requirements Engineering (RE). RE can be defined as *the domain that encompasses all project lifecycle activities associated with understanding a product's necessary capabilities and attributes* [Weigers 03]. There are many processes that

make up RE such as requirements elicitation, analysis, specification, management, and validation [Macaulay 96, IEEE 04, Jodhi 03, Weigers 03].

By using these definitions, we can see that the discipline of Requirements Engineering uses requirements as the medium for understanding a system's necessary capabilities and attributes. Matthias Jarke and Klaus Pohl further refined this idea, and developed what they called their three dimensions of requirements engineering [Pohl 93, Jarke 98]. They did this in a response to the maturing field of RE and how it went from a fuzzy early stage of systems development to a crucial tool in establishing a vision for system-related change. Their three dimensions are as follows:

1. Manage the convergence of stakeholder interests toward agreement on key system goals and constraints.
2. Achieve a sufficient shared understanding of the issues involved in realizing the system vision, such as its functionality, nonfunctional properties, intended, and unintended side effects.
3. Document this understanding in adequate representation formats for human information sharing as well as computerized system development. [Pohl 93, Jarke 93, Jarke 98, Ramesh 01]

Formal frameworks for software development have also incorporated Requirements Engineering concepts. An example is the Software Engineering Institute's Capability Maturity Model Integrated (CMMi), which many companies have used to improve their requirements engineering (and overall software engineering) practices [Leffingwell 96, Tvete 99, Heinz 04, Goldenson03]. In the CMMi, requirement-related activities are divided into the Level 2 Key Process Area (KPA) of Requirements Development and the Level 3 KPA of Requirements Management [SEI 02]. Requirements Management establishes the base requirement elicitation and management activities for a given project. It includes practices such eliciting requirements, managing change, and making sure requirements are incorporated into the project plans and development artifacts. One of the practices defined by the Requirements Management KPA is "Maintain Bidirectional Traceability of Requirements". This is also known as Requirements Traceability (RT), which is the focus of this paper.

On a basic level, RT is considered as the ability to link requirements, design, code, and tests to each other during the development process [Gotel 94]. There has been extensive research on how to make RT possible in a technical sense. Numerous techniques such as manual linking [Weigers 03], Information Retrieval [Antoniol 00, Antoniol 02, Hayes 03], Scenario-Based Traceability [Egyed 00], and Event-Based Traceability [Cleland-Huang 02a, Cleland-Huang 03a, Cleland-Huang 03b] have been developed in order to link artifacts with each other to understand the relationships between artifacts. Several tools, which include TeleLogic DOORS [TeleLogic], Rational Requisite Pro [Rational], TOOR [Pinhiero 96], and several others [INCOSE 03] have been developed in order to manage requirements and the links between them using these techniques.

However, with all of the academic research and commercial support for RT tools and techniques, the main source of project failure within IT is still centered on requirement-related problems

[Standish, Leffingwell 97]. While RT by itself cannot turn this tide, it can definitely assist in mitigating requirement-related issues. There are no hard figures on the adoption of RT within the industry as a whole, but some research has suggested that even though there are organizations that utilize some sort of RT practices, many still struggle with implementing traceability into their systems development lifecycle, or do not understand RT at all [Jarke 98, Ramesh 98, Ramesh 01, Egyed 02]. These struggling organizations are the focus of this paper and will gain the most benefit from the VBRT framework.

This paper posits that requirements traceability can provide actual value to a project within the software/systems development lifecycle by utilizing a blend of manual and dynamic traceability techniques based on the needs of the project at hand. As previously stated, if organizations can realize this value, then it might be possible to more effectively utilize RT. In order to support this claim, we developed the Value-Based Requirements Traceability (VBRT) for organizations to model, understand, implement, and measure the value that RT can provide within their specific situations. This framework is based on the Value-Based Software Engineering (VBSE) framework developed by Barry Boehm [Boehm 03]. The goal of the VBSE framework is to shun the idea of value-neutrality, where a process or concept is undertaken without understanding the inherent value it does or does not provide. In this value-neutral world, every practice, tool, etc. would have the same theoretical value, and therefore they are valueless in comparison to each other. Within the VBSE model, software engineering activities and concepts must compatibly reinforce each other as well as the overall organizations' activities and goals [Boehm 03]. They must be shown to provide value to each other, the project, and the organization, or they possibly should not be undertaken at all. While the VBSE framework pertains to software engineering as a whole, our framework is obviously centered on requirements traceability.

This thesis will be broken into several chapters in order to present the VBRT framework. The third chapter, "Requirements Traceability", will give a background and foundation into RT. Several concepts and RT techniques that our framework will utilize will be discussed, as well as how applicable they are in certain situations. This overview will cover what research has been done as well as progress in industry of RT activities. Automated and manual techniques will be explained and analyzed, not theoretical processes since there is usually an implementation gap between theory and practice.

Once the stage is set for the role of RT in software engineering, Chapter Four, "Value-Based Software Engineering", will center on the relative newness of VBSE as well as the purpose and goals of viewing software engineering efforts as activities that must provide "value" to an organization. Chapter Five, "Value-Based Requirements Traceability", will describe the VBRT framework model, its seven inherent processes, and the realization of the processes.

After the background and framework has been laid out, Chapter Six, "Case Studies", will describe three theoretical scenarios in which VBRT would be implemented and how it can be accomplished. Chapter Seven will cover "Future Work" that will provide others with further concepts that can be explored on this topic. The final chapter will explain the "Conclusions" from this research.

CHAPTER THREE: Requirements Traceability

Tracing requirements is a subset of the RE activity within a software development project. But what exactly is requirements traceability? The most standard definition of requirements traceability was set forth by Orlena Gotel and Anthony Finkelstein, which defines RT as *"the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)"* [Gotel 94]. This definition has two interesting properties that make up the practice of requirements traceability. First, the definition refers to the life of a requirement. This indicates that a requirement is not a static concept that is just composed of text translated from a stakeholder's mouth. A requirement can be thought of as a dynamic concept that changes and evolves throughout the software development lifecycle. It can be enhanced, taken away from, added to, dropped from the scope altogether, split into multiple requirements, or aggregated with other requirements. Research shows that requirements can and do change throughout the duration of a project. For example, one of IBM's labs reports that 25% of requirements on an average system being developed within the company will change [O'Neal 01]. There is a very extensive body of work on the subject of requirements change, which is discussed in the *Change as Opportunity* section (Section 5.7).

The second concept from the RT definition that deserves attention is the idea of tracing in both a forwards and backwards direction. The CMMI agrees that in order to have well managed requirements, "*traceability can be established from the source requirements to its lower level requirements and from the lower level requirements back to their source"* [SEI 02]. The SWEBOK also states that "*a requirement should be traceable backwards to the requirements and stakeholders that motivated it* and *should be traceable forwards into requirements and design entities that satisfy it"* [IEEE 04]. This means that it is necessary to trace a requirement to the artifacts that implement it, as well as tracing from an artifact to the requirement that the artifact itself implements. Matthias Jarke in [Jarke 98] notes there are four types of traceability links, which are as follows:

1. Forward from Requirements: This indicates links that go from a requirement to an artifact.
2. Backward to Requirements: This indicates links that go from an artifact in the development of the system to a requirement.
3. Forward to Requirements: This indicates links from a source of a requirement to the requirement itself.
4. Backward from Requirements: These are links that go from a requirement to the source of a requirement.

As can be seen, the type of requirement depends on the start and ending point of a link. This is important to understand since current RT research divides traceability into two phases. The phases are Pre-Requirements Specification (Pre-RS) and Post-Requirements Specification (Post-RS) [Gotel 94]. We developed the following model to illustrate the relationship between Pre-RS concepts and Post-RS concepts.

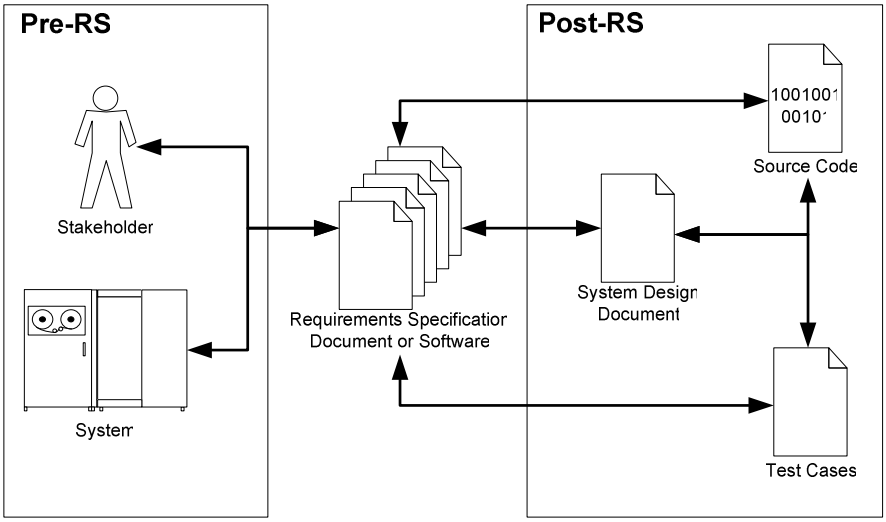

Figure 3-1. Pre-RS and Post-RS Traceability

3.1 Pre-RS Traceability

Pre-RS traceability includes the aspects of the life of a requirement before inclusion into a Software Requirements Specification (SRS) [Finkelstein 91, Gotel 94]. The SRS can be a formal structured document as used in the Rational Unified Process [RUP], or it can be story cards as used in agile methodologies such as Extreme Programming [XP]. The medium does not matter, but the data and context do matter. In essence, the goals of the activities that fall under Pre-RS center on producing requirements for a system by identifying and tracing requirements from the RS to their originating statement and sources. This makes pre-traceability the link between business needs and IT [Jarke 98].

It is interesting to note that the trend of most RT problems identified through research stem from Pre-RS concepts such as identifying sources or rationale for requirements [Gotel 94, Gotel 95, Gotel 97, Pohl 96, Ramesh 95]. This is mainly due to the lack of systematic techniques and methods to record and trace information related to RS production [Gotel 94, Haumer 99] . Even though these techniques are still being researched, it is a goal of this paper to show how Pre-RS efforts fit into the overall value of a project.

3.1.1 Contribution Structures

One of the areas of research that has received much attention in the context of Pre-RS is Contribution Structures. This area has risen from the fact that based on industry observation and study, organizations seem to have a general lack of knowledge about the stakeholders and subject matter experts that give rise to requirements for a system in the first place [Gotel 95]. This lack of knowledge is very problematic since people are usually considered the ultimate source for requirement validation, examination, and refactoring. In order to avoid this lack of knowledge, one must link (and later trace) sources or contributors of requirements to the artifacts that they contribute to. However, it has been shown that a basic notation of defining the origin of a requirement is not enough. In [Gotel 95], Gotel and Finkelstein note that contributors to the

requirement generation process as well as their roles in the process need to be understood and documented. Once the basic contributors have been identified, there can be further qualification of the contributors of requirements, which involves their relation to each other as well as the requirements themselves. This can also assist in identifying stakeholder value propositions and conflicts as described in the *Stakeholder Proposition Elicitation and Reconciliation* section (Section 5.2). The contributor, the level of knowledge the person has, and other aspects can be captured within the traceability link for each artifact for a system being developed as well. Therefore, if a project member is unsure about information related to a requirement, the developer can trace backward from the requirement to understand who documented the requirement, who initially gave the requirement, and can ascertain who the best person would be to contact to answer any questions based on the strength of knowledge or involvement of the person who provided the requirement [Finkelstein 91]. This concept is also important for change management purposes (See Section 5.7). By using traceability links, any associated artifacts and requirements that may be affected can be discovered, and the appropriate sources can be contacted and informed.

3.1.2 Rationale

Another aspect of Pre-RS traceability that should be considered is understanding the rationale of a requirement. Contribution structures provide information on who provided a requirement, when they provided it, and how the person relates to the artifacts that are impacted by the requirements they provide. Rationale provides the project team the information of why a requirement exists at all. For example, the Phillips Medical Systems MIMIT (Medial Imaging Information Technology) Group captures requirement rationale to better understand why a requirement exists in order to help separate the needs of clinical users and commercial users [Higgins 03]. The idea of capturing rationale can be as informal as just defining the reason that a requirement exists, as provided by the source of the requirement. It is important to note that rationale-related information that is captured is usually a result of human interaction in providing requirements for a system, so this type of simple rationale capture requires little overhead. A more advanced approach to RT and rationale management can be to document various issues, alternatives, assumptions, and arguments (with resolutions) that led to the creation of a requirement [Ramesh 01]. Capturing rationale can also focus on information related to external constraints that need to be considered such as mandates, standards, or policies that will affect the development of a system.

Understanding rationale can also be very beneficial during the development process when there are conflicts between requirements, understanding requirements when traced back from an artifact, or documenting requirements that are derived from other requirements. It has been noted that the information known about derived requirements is often sparse or unrecorded which can lead to development misunderstandings and errors. To minimize this, rationale information should be captured and understood for parent requirements, somewhere within a high-level specification, or in attributes within links to the artifacts that led to the creation of the requirement in the first place [Ramesh 01, Domges 98].

3.2 Post-RS Traceability

Post-RS is defined as the ability to trace requirements from, and back to, a baseline (the RS), through a succession of artifacts in which they are distributed. Furthermore, any changes to the baseline need to be re-propagated throughout the chain of traceability links [Gotel 94]. Post-RS activities mainly focus on traceability forward from requirements and backward to requirements. There are several tools and techniques that have been developed over the years to support the concept of Post-RS traceability. The range of techniques goes from completely manual concepts such as using traceability matrices to fully automated technologies such as Event-Based Traceability and Information Retrieval methods. There are several techniques that a user can employ when undertaking an RT effort. Some of these techniques can be used standalone, or within a requirements management tool such as Telelogic Doors [TeleLogic] or Rational Requisite Pro [Rational]. While this section does not contain an all-inclusive list of traceability techniques, these methods represent the major advances in traceability research and practice. As will be seen, these techniques will be used to feed into the *Business Case Analysis* section of the VBRT framework (Section 5.3).

3.2.1 Simple Links

One of the most basic forms of performing RT is using a hard-coded requirements traceability matrix within a tool such as Microsoft Excel. An RT matrix is "*a table that illustrates logical links between individual functional requirements and other system artifacts"* [Weigers 03]. An example of such a matrix is shown below:

Table 3-1. Example Requirements Traceability Matrix

User Requirement	Functional Requirement	Design Element	Code Module	Test Case
Requirement 10	orders.query	Class orders	orders.lookup() orders.list()	Test Case-7
Requirement 11	orders.sort	Class orders	orders.sort()	Test Case-24 Test Case-46 Test Case-47

This table shows a link to the requirement as given by the user, the requirement as outlined in a requirements document, the design artifact that it references, the sections of code that it is implemented in, as well as the test case that tests the requirement. This simple type of RT meets the requirement of showing the links in both a forward and backward direction. However, the process of developing and maintaining a traceability matrix is completely manual. While this results in very precise links since a user can make exact determinations as to the correct linkages between artifacts and requirements (as opposed to other dynamic methods) [Cleland-Huang/Zemont 04], there must be discipline to maintain the complex web of linkages throughout the course of a project in order for the matrix to be of any value. Since requirements change often, it is very easy for a matrix to fall out of sync with the current requirements and other artifacts of a system.

3.2.2 Event-Based Traceability

Event-Based Traceability (EBT) is a dynamic method of traceability using executable models and simulations that establish links between requirements and artifacts. It is based on the "event-notifier" design pattern, and its main goal is to *"notify dependent artifacts and developers of changes that have occurred in order to support the developer in the task of updating an artifact"* [Cleland-Huang 03a]. Once a requirement or set of requirements change, the change events trigger performance models to be re-run with the new changed values to test the performance aspects of a system. In order to accomplish this, EBT uses a "publish-subscribe" model. Requirements are published and the performance models subscribe to the system. The EBT system ties the two together so a change in a requirement causes a model that is subscribed to that requirement to understand the new values [Cleland-Huang 02b]. As previously shown, the cost of a change or defect found during the test phase of a project is much more significant compared to understanding and catching potential defects during the requirements phase [Boehm 01a, Shull 02, Leffingwell 97]. The main advantage of EBT is that when changes are introduced into the system, they can be adequately propagated throughout the system of artifacts [Cleland-Huang 03a].

EBT accounts for two main types of changes: contextual changes and functional changes [Cleland-Huang 02a]. Contextual changes are those of the types described above where a quantitative change has been made within an existing requirement. These changes will update the EBT system, and trigger the automated re-execution of relevant performance models with the new values from the change. The other type of change is a functional change. This type of change results when a new requirement or functionality is introduced to the system. This type of change's impact is harder to predict since it involves changes in functionality and performance, while contextual changes primarily impact the performance aspect of a system [Cleland-Huang 03a].

Both types of changes can also be used to perform impact analysis [Cleland-Huang 02a]. As a system evolves, developers can use different quantitative values within requirements, or introduce new requirements, and re-run the performance models in order to see what potential changes as well as new requirements *may* do to a systems performance. The result of running the performance models can show the impact of the changed requirements in terms of cost and effort. This enables developers to understand the effects of changes within a requirement in relation to other requirements and the system as a whole and allow project managers to analyze the risks of implementing changes or not implementing them. [Cleland-Huang 03a]

EBT is a semi-automatic process. The "publish-subscribe" model used within EBT allows automatic linkage and validation of values within requirements that are established with the EBT system. This type of automation eases the manual intervention to maintain linkages and update artifacts based on changing requirements. The EBT system can then act as an automated change notification agent. When changes are made, the affected artifacts and models are logged, and the developer can determine which artifacts to update, and which to ignore, based on factors such as the criticality of the requirement changed. Criticality can be determined by the strength of the relationship between objects being reviewed, which is stored in the subscription database. However, the publish and subscription relationships must be setup by the user within the EBT system. Once links are established, changes can then be monitored [Cleland-Huang 03a].

3.2.3 Information Retrieval

Automated Information Retrieval (IR) techniques are used to link artifacts to each other as well as requirements via a mechanism that queries a set of artifacts. By using an indexing process based on pre-processing documents, identifying attributes, or developing a thesaurus of keywords, artifact information and semantics are parsed and used in order to rank the artifact on its relevance to a given query. The rankings are then used to create links between artifacts, which are retuned to the user. The user can use the rankings in order to understand relationships between artifacts and even requirements in order validate the links that have been generated by the IR algorithm [Antoniol 00, Antoniol 02]. Therefore, IR techniques are ideal for recovering and identifying links when requirements already exist as do the artifacts that have been produced or are being produced by the system development process. This can be referred to as "after-the-fact" requirements traceability. [Hayes 03]

The effectiveness of any IR technique is measured against two main metrics: recall and precision [Antoniol 00, Hayes 03]. For a given query, recall is the ratio of the number of correct links retrieved over the total number of correct links. Precision is the percentage of the matches within the recall set that are relevant to the query. It is important to understand that while an IR query mechanism is an automated mechanism in which to retrieve artifacts, IR techniques can require manual intervention to ensure that the retrieved linkages are of significant relevance to the query and intent at hand. This is because there can be a significant error rate, or false positives, in the links that have been retrieved from a query. For the links between artifacts that are retrieved, it is often necessary for a developer to manually review the link association before determining that the formal link is valid between artifacts or requirements. The lower the precision, the more manual intervention is needed to review the returned results. [Antoniol 00, Antoniol 02]

There are two main IR models (techniques) that have been researched for use with requirements traceability. In a probabilistic IR model, documents (artifacts) are ranked according to the probability of being relevant to a query. The other type of model is a vector-based IR model. In this model, each document and each query are mapped onto different vectors. Documents are then ranked against queries by computing distance functions between the corresponding vectors. [Antoniol 00, Antoniol 02].

Even though one of the drawbacks for using IR is the manual intervention needed for low precision queries and brute force impact analysis required for low recall queries, research has shown that retrieving and associating links using IR is still faster than a 100% manual approach to RT. This has been shown on a system of thousands of requirements and millions of lines of documentation, as well as systems having only 20-70 requirements and a few artifacts [Tryggeseth 97, Hayes 03]. Research has also shown that recall and precision increases as larger and larger document bases are used even though there is no lower cutoff on the minimal artifact set needed for proper IR usage [Hayes 03]. However, an increase in precision and recall can have an adverse affect on time for retrieval. As a general rule, probabilistic algorithms are more precise [Hayes 03]. Therefore, there is a tradeoff using IR based on speed of retrieval or high levels of precision.

As previously mentioned, IR can provide great benefit is by recovering links between artifacts when the base of requirements and artifacts is very large, and manual traceability is not feasible

due to time or the cost of traversing thousands of documents in order to develop linkages. This can be done during the system development process to better perform impact analysis or during the maintenance process in order retrieve links from source code to documentation [Tryggeseth 97]. If developers who maintain a system can link source code to documentation, they can better understand the requirements of a system that they may have to modify in the future. By developing such links, they can better assess the impact of potential code changes, as well as understanding the system itself, by retrieving all documents associated with the source code [Antoniol 00, Antoniol 02].

3.2.4 Scenario-Based Traceability

Another traceability technique that is used to recover links between source code and artifacts is called Scenario-Based Traceability (SBT). The impetus behind the evolution of this type of traceability that RT users often cannot trust the links that are retrieved from an RT tool by techniques such as IR. Even if there is a 30% error rate (inclusion or exclusion) in potential links returned from a system based on a given threshold, if the user does not know which 30% are in error, the set of returned links is rather useless unless there is time to manually validate them all [Egyed 00]. In order to increase the accuracy of retrieved links, especially during reverse engineering or maintenance of systems, scenario-based RT generates traceability information based on observing the results of test scenarios that are executed on a working system.

There are 3 pre-requisites that must be met in order for this technique to work. First, there must be a running system to test against. The system need not be complete, as prototypes or complete sub-sections are appropriate. The second pre-requisite is that there must be a corresponding software model for the system such as Unified Modeling Language (UML) class diagrams. Finally, there must be test cases or scenarios that can be executed [Egyed 00].

Once these three requirements have been met, an automated tool can be used to match the software models against the actual source code implementation of a running system via executed test scenarios. Through this, dependencies between source code and artifacts are created, and those relationships can be used to establish RT links. The source code that is executed during the running of a test scenario is called its “footprint” [Egyed 00].

While there is a tool called Trace Analyzer [Egyed 02] that can be used to create the links between artifacts and source code, other tools are needed to observe the footprint that the scenarios make such as Rational PureCoverage [Rational]. By using the tools in concert, SBT can validate four different types of traces:

1. Traces between scenarios and the system.
2. Traces between model elements and the system.
3. Traces between scenarios and model elements.
4. Traces between model elements.

These are the trace types that can be returned from an SBT tool. Since the links retuned are generated from the system actually running and being tested, research shows that the error rate, or number of false positives, is rather low [Egyed 00, Egyed 02]. Errors do occur, but they

mainly result from incomplete models or erroneous or deprecated source code. Apart from that, SBT has other uses that make it a viable technique. For example, if it is determined that a model or other artifact has an extensive number of links to another artifact, the concept of the strength of a relationship between two artifacts can be introduced. For larger systems, SBT can help reduce ambiguity in manual links by introducing dynamic verification, where a user may have thought an dependency or link existed, but it did not. Furthermore, by utilizing dynamic linkages SBT is a viable solution for requirements change management activities as well as impact analysis of changing requirements [Egyed 02].

3.2.5 Heterogeneous Traceability

Heterogeneous Traceability is not an RT technique per se, but rather a model for implementing RT as proposed and explained by Cleland-Huang, Zemont, et. al. in [Cleland-Huang/Zemont 04]. Research has suggested that an effective traceability solution would not necessarily adhere to only using one technique throughout a project, but that an effective traceability solution can utilize the strengths of more than one technique. This can be accomplished by using a proper blend of manual (Simple Links) and dynamic traceability techniques (EBT, IR, SBT) in order to establish proper levels of traceability for a certain subsets of requirements based on factors such as criticality, volatility, impact of change, and amount of risk to the system [Cleland-Huang/Zemont 04]. For example, this approach may use manual traceability for critical requirements, EBT for volatile requirements, IR for large quantities of static requirements, and potentially leaving only a small number of non-critical requirements that have a low probability of change untraced. Decisions on which techniques are best for a given set of requirements need to be made within the context of project-level trace objectives [Domges 98, Cleland-Huang/Zemont 04].

3.3 Types of Traceability Users

The previous concepts and techniques, while studied extensively, are not de facto standards within industry. Different organizations obviously have myriad approaches to implementing and using RT based on their experience and maturity. Between 1998-2000, Balasubramaniam Ramesh conducted surveys and research on the level of usage of RT practices in 26 different organizations from U.S. government systems development industry to the pharmaceutical industry to electronics and software consulting. As a result of the industry surveys which are outlined in [Ramesh 98] and [Ramesh 01], he noticed that users of RT can be divided into two main categories: low-end and high-end users.

Low-end users typify a more immature attitude and approach toward RT. Nine of the companies surveyed fell into this category. Ramesh found that these types of users tend to see traceability as a mandate forced on them by upper management or the project manager. Low-end users tend to use simpler forms of traceability to model dependencies, such as manual traceability matrices. They also tend not to implement Pre-RS practices in any form, and therefore lose all rationale and contribution structure analysis information. In essence, they capture the links themselves, but not the semantics surrounding the links. One user from his study said that *"often we have no idea who made these decisions, and how they impact the rest of the effort. Simply trying to do these at the end of the project or after the fact does not work"* [Ramesh 01]. Furthermore, this type of

user tends to view RT as limited in scope. Links primarily focus on linking requirements to system components such as design and links to test cases are not usually defined. Another comment from the study was that "when you want to know which part of the system satisfies which requirement, you can go to the allocation (matrix) table to identify it. Whether it does indeed or not is another matter – for testing to worry about." [Ramesh 01] Lastly, low-end users view RT linkages as static one-time concepts. As a result they are not well maintained and tend to fall into obsolescence during the system's development lifecycle. This view toward RT supports his previous findings that low-end users consider RT as "expensive overhead" that is often one of the first processes to be eliminated when a funding crunch comes along [Ramesh 98]. The *Business Case Analysis* section (Section 5.3) will show that this financial attitude toward RT is flawed.

High-end users typified the rest of the companies studied. These users tend to implement much richer traceability schemes which allow them to understand more about the traces and the requirements themselves. These users tend to use more sophisticated RT techniques for developing links, but they also understand the importance of link maintenance throughout changes in the lifecycle of the project. This is especially true in safety critical systems built in industries such as aerospace and pharmaceuticals, where documenting each and every activity detail has been a long established practice [Jarke 98, Watkins 94]. High-end users also utilize Pre-RS concepts as a critical element to defining their requirements and links. They have processes surrounding the logistics of managing requirements such as RT tool and technique implementation, defining and maintaining rationale links, defining and maintaining design and system component links, and defining and maintaining compliance verification links to test cases or other verification mechanisms [Ramesh 01]. As a result, the systems that these users build tend to have a higher probability that it will meet all the customers' needs and will be easy to modify and maintain [Ramesh 98].

Summary

In this chapter, we covered the concepts of Pre-RS, which is composed of Contribution Structure analysis and capturing Rationale. Analyzing Contribution Structures involves understanding the people who contribute requirements as well as their relationship to each other, which can help enable the Value-Based Requirements Traceability aspect of *Stakeholder Value Proposition Elicitation and Reconciliation* (Section 5.2). Post-RS is composed of techniques to trace requirements from a requirements specification to the design, code, and tests that implement them. There are four individual techniques for performing Post-RS RT: Simple links, EBT, IR, and Scenario-Based Traceability. Simple links allow the user fine-grained control over linkages, but is rather labor intensive. EBT is a dynamic technique that can enable change management and impact analysis for requirements during the development lifecycle. IR is another automated technique that can enable links between mass amounts of requirements and artifacts on an as-needed basis. Finally, Scenario-Based Traceability is a dynamic technique that uses test cases as the mechanism to create linkages. These four techniques feed into the concept of Heterogeneous Traceability, which purports that using a blend of manual and dynamic techniques, based on the project at hand, offers the most value to the organization while minimizing costs. This is will detailed in the *Business Case Analysis* section (Section 5.3). As will also be seen in the next

chapter, each of these techniques will help derive the benefits in the *Benefits Realization Analysis* section (Section 5.1).

CHAPTER FOUR: Value-Based Software Engineering

The definition of value as put forth in the Marriam-Webster dictionary states that value is something's "*relative worth, utility, or importance"* [Webster].The term *relative worth* is important since in order for something to have value, it needs something else to compare itself against. In his paper on VBSE, Boehm explains that in many organizations, software engineering practices are done in a "value-neutral" setting, whether they center on requirements traceability, defect tracking, or configuration management [Boehm 03a].. Therefore, it can be stated that software engineering practices in many organizations are often performed without understanding their relative worth to the business. The relative-ness of the worth is important because it means that in a value-neutral setting, the activity is done in isolation without anything to compare itself against, such as the bigger picture in which the activity exists. The result is that the connections between technical aspects and value-creation are vaguely understood, if at all [Boehm 00b]. Symptoms of this can be seen in the following examples that we developed:

1. Defects or requirements are all treated equally as important as the next. The result is that critical requirements or defects may not be attended to properly.
2. Measurements of project progress might only center on how much time has been spent or money has been burned, rather than how much value has been completed or achieved.
3. People doing the actual work may be so far removed from understanding why they are doing specific work, that they don't comprehend the bigger picture, and therefore cannot communicate or get the most value out of their work.[Boehm 03b]

These are profound statements, since in value-based settings upper management can now ask "If I don't know an activity's relative business worth, or value, then why are we doing it?" To upper management, an employee or activity can obviously be seen as a liability or an asset. In today's rapidly changing IT environment, dollars need to be spent very wisely in order for a company to succeed. No longer can software engineering and IT in general work in value isolation. It is crucially important that organizations tie software engineering activities to business value so they can stay competitive [Boehm 00b, Boehm 03a, Boehm 03b].

In order to accomplish this, Barry Boehm describes that the VBSE agenda *"accepts the challenge of integrating value considerations into all of the existing and emerging software engineering principles and practices, and of developing an overall framework in which they compatibly reinforce each other"* [Boehm 03a]. There are seven key principles that software engineering activities, such as requirements traceability, can use as a foundation for framing within a value-based environment. These elements are the foundation of the Value-Based Requirements Traceability (VBRT) framework.

Boehm's seven principles [Boehm 03a] are introduced below with brief definitions that we derived:

- **Benefits Realization Analysis**: Understanding the goals and benefits of undertaking an activity or set of activities.

- **Stakeholder Value Proposition Elicitation and Reconciliation**: Understanding the stakeholders involved in the activity, their needs, and how they interact with each other.
- **Business Case Analysis:** Developing a sound qualitative and quantitative picture of the costs and benefits for undertaking an activity.
- **Continuous Risk and Opportunity Management:** The process of constantly looking for risks within the activity in order to properly handle them, as well as potentially realize them as a value-creation opportunity.
- **Concurrent System and Software Engineering**: Developing a system in a non-linear (non-waterfall) fashion.
- **Value-Based Monitoring and Control:** Creating mechanisms to understand the state of an activities value creation progress in order to make adjustments as necessary.
- **Change as Opportunity:** Understanding that change is a constant in any activity, and it can be embraced as an opportunity to make an activity or concept better.

While these seven aspects present the foundation of the VBRT framework, we will not address them all equally since they do not all have equal value to the purpose of this research. *Concurrent System and Software Engineering* (Section 5.5) is still an emerging concept within the arena of RT, and while it is a very interesting topic, it warrants it's own separate in-depth study. It is therefore outside the scope of this paper. Also, the aspect of *Change as Opportunity* (Section 5.7) will not be extensively explored in this paper since the topic, within the context of RT, has extensive research behind over the past 15 years.

Summary

According to Boehm, the seven aspects of the VBSE framework can help organizations understand that software engineering activities should be considered in terms of value creation for the business. VBSE can also assist businesses in realizing the value that they are getting out of software engineering activities as well. Research into Value-Based Software Engineering concepts has already been explored in a few areas such as software design [Boehm 01b], ROI analysis of software development [Boehm 03b], and product-line engineering where software engineering activities are tied to the concept of *customer value* [Faulk 00]. Our paper will add to this body of knowledge. While we will not focus on all seven aspects equally, we will propose a VBRT framework to apply these concepts to requirements traceability. The next section will present the VBRT framework and show VBSE can be applied to RT.

CHAPTER FIVE: Value-Based Requirements Traceability

In his VBSE paper, Boehm describes several software engineering practices that can and should be tied to the business value of a systems development initiative. One such example is Value-Based Requirements Engineering (VBRE). VBRE is described as principles and practices for identifying critical stakeholders and constantly matching their needs to the functionality of a system, thereby developing a mutually agreed set of objectives for a system [Boehm 03a, Boehm 03b]. By using a VBRE frame of reference, requirements engineering can be thought of as a living entity that reflects a mutually agreed set of objectives for a system (between users/customers and the implementing team). Requirements, and therefore stakeholder objectives, are constantly evolving elements that need to be managed and tracked in order to make sure the system being developed is providing value to the customer and accurately reflects the customers' needs at all times [Lam 98, Anderson 02, Cleland-Huang 03a]. RT is the mechanism that enables the tracking and management of requirements over the lifecycle of a project, and beyond [Weigers 03]. There are two objectives for VBSE and requirements traceability:

1. Show **that** RT can provide value
2. Show **how** RT can provide value

In order for RT to be understood as an essential software engineering activity, it must be shown to provide value within the context of a project, not just an activity in and of itself. This avoids looking at RT in a value-neutral setting. But simply showing that RT provides value to an organization does not address how it can be done. Direction needs to be given on how RT can provide value that can accommodate flexibility in order for it to be useful to different organizations with different requirements for understanding value. This is the foundation of the Value-Based Requirements Traceability framework outlined in this section.

As previously mentioned, the VBRT framework consists of the seven aspects of VBSE. They do not act as independent entities, but rather as interrelated concepts.

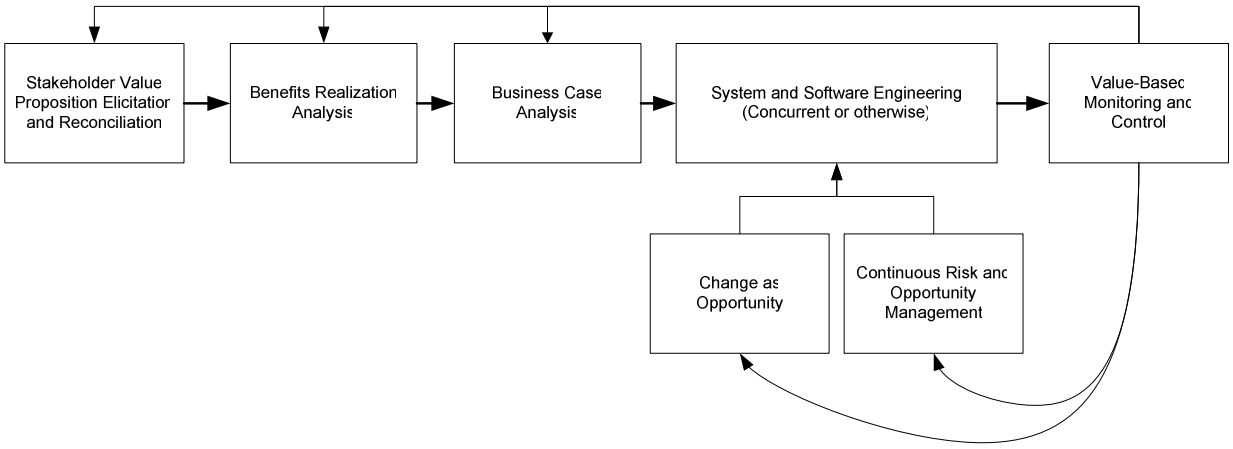

Figure 5-1. Value-Based Requirements Traceability Model

Since the framework can be thought of, and is presented as, a process flow, we will use the term process from now on to describe the seven aspects of the framework. The first process of the framework is *Stakeholder Value Proposition Elicitation and Reconciliation* (Section 5.1), whereby the organization will identify the stakeholders in the RT process and assess their needs and conflicts. Using the needs of the stakeholders as a basis, the organization will then perform a *Benefits Realization Analysis* (Section 5.2) in order to understand the benefits the company would derive from RT and map them out as goals to be implemented. These goals would then be verified and financially framed by performing a *Business Case Analysis* (Section 5.3). These three process would then feed into the *System and Software Engineering* (Section 5.4) process, where RT is utilized in practice. The VBRT framework does not specify the manner in which this activity is carried out. We propose the framework can be used in either a concurrent/agile or waterfall-type approach. *Value-Based Monitoring and Control* (Section 5.7) takes all of the cumulated processes as an input, and introduces the concept of measurement, monitoring, and reacting to change and risk during the *System and Software Engineering* process. If there are risks that arise, the framework utilizes the *Continuous Risk and Opportunity Management* (Section 5.5) process and the *Change as Opportunity* (Section 5.6) process to feed alterations of the RT process back into the *System and Software Engineering* process. Finally, *Value-Based Monitoring and Control* also acts as a general feedback loop into the rest of the framework processes if general adjustments needs to me made. As an example, the cost of managing requirements can change, so there would need to be alterations made in the *Business Case Analysis* process. Or, new stakeholders might have been identified, and their needs might need to be incorporated into the *Stakeholder Value Proposition Elicitation and Reconciliation* process.

Each process in the VBRT framework relates to and can feed into other processes. We have divided each process into three major components similar to many formal frameworks, such as the Project Management Body of Knowledge (PMBOK) [PMI 00]. Each process has *Inputs, Techniques, and Outputs*. *Inputs* are the concepts that feed a process. Some of these are derived from other processes, others are unique to that process. *Techniques* are the tools used in which

the process is undertaken. Lastly, *Outputs* are what the process creates or enables. These three components allow us to see each process from a macro view and understand how they relate to each other.

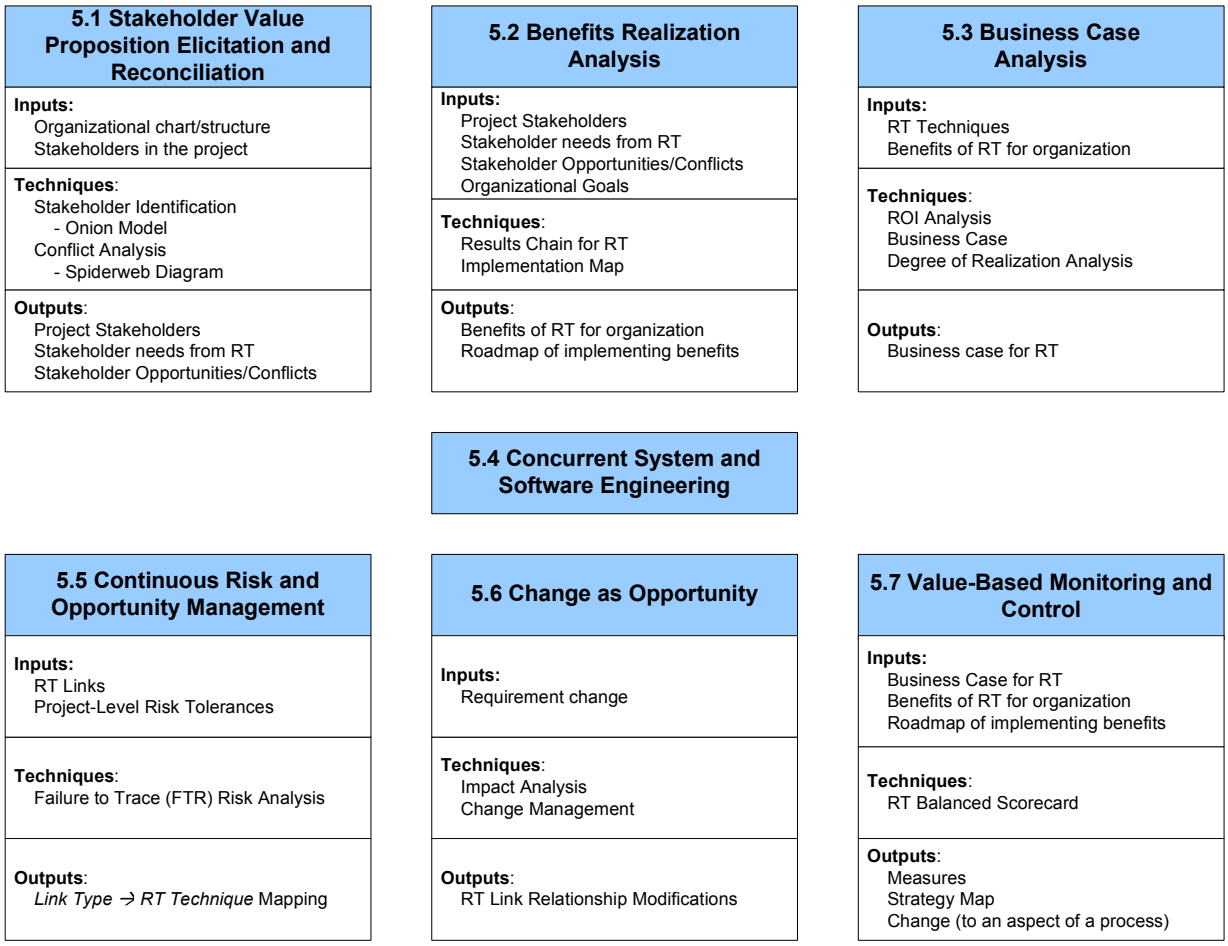

Figure 5-2. The VBRT Framework Components

5.1 Stakeholder Value Proposition Elicitation and Reconciliation

A project stakeholder can be defined as "*someone who gains or loses something as the result of a project*" [Alexander 04]. Research has shown that all stakeholders and participants in a systems development activity do not always agree with each other 100% of the time. Different people have different needs, based on their roles in a project. This can cause conflicts and problems if not adequately understood. There is an extensive body of literature on stakeholder identification, needs analysis, and conflict resolution (further analysis is out of scope for this paper) [Collins 94, Sharp 99, Preiss 01, Giesen 02, Alexander 04].

The activities that are executed within the RT domain are usually not done by one person in a vacuum. Several different members of a project and an organization as a whole have an interest in the RT process. This is due to that fact that system artifacts and concepts such as documents, code, policies, tests, plans, and designs are all linked to requirements. As a result, the RT activity

touches many aspects of an organization. It is necessary to understand the various stakeholders in the RT process and be able to meet their needs in order for RT to provide the most value for an organization and its customers.

We derived our set of stakeholders as well as their needs, or value propositions, from personal experience and research into the environments of high-end RT users [Ramesh 95, Ramesh 01]. This was done for two reasons. As previously described in Chapter Three, high-end users typically employ more mature RT practices, and therefore have a more comprehensive view of RT and the value that it can provide. Most low-end users have not yet realized the value of RT. Second, most studies and research have focused on high-end users, so the data is a little more extensive and complete. In practice, deriving stakeholders could be done through organizational charts, or by analyzing common roles on project teams.

As Boehm originally proposed with VBSE [Boehm 03a], stakeholders in RT operate and have influence on both the organizational and project level. We propose the following "Onion Model" [Alexander 04] that illustrates the various stakeholders that interact in the RT domain. While these roles are the most common, they are by no means all inclusive. There can be many other more specific roles that can be a part of the RT domain, and their needs would have to be accounted for and met as well.

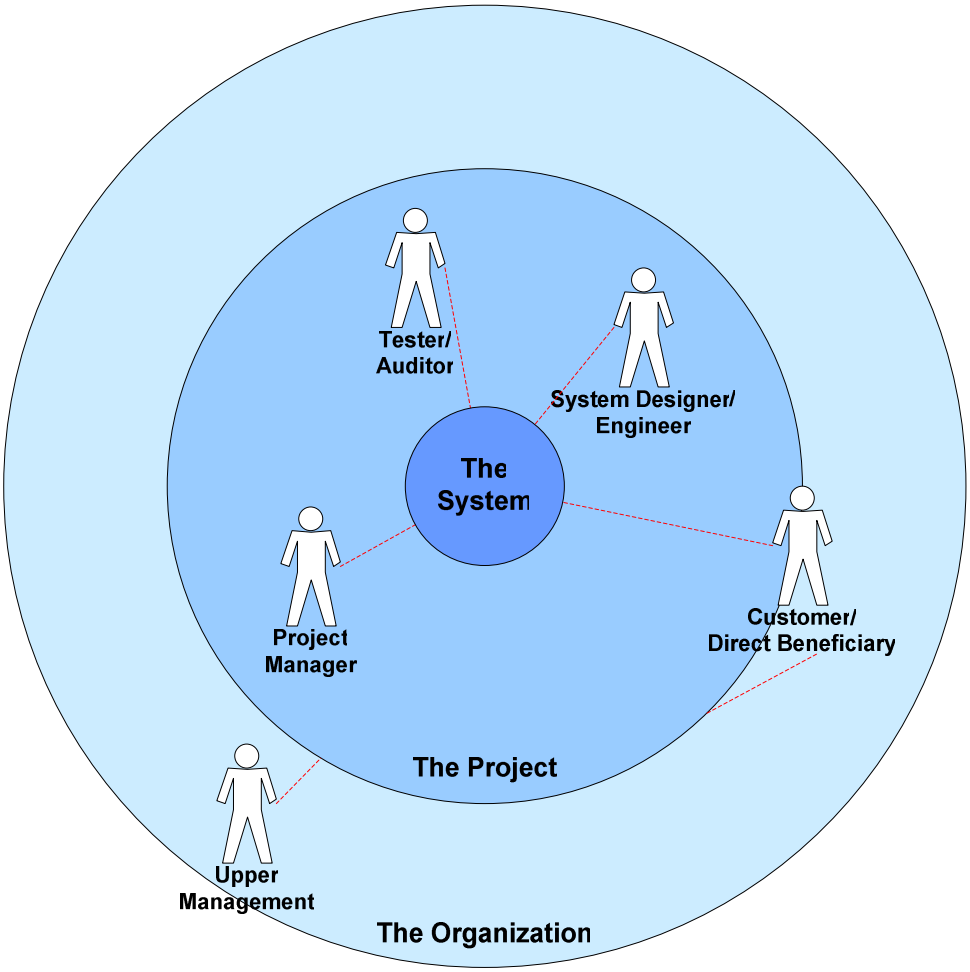

Figure 5-3. Onion Model of RT Stakeholder Relationships

As the diagram shows, the Project Manager, Testers/QA, and System Designers/Engineers operate on the project level and directly interact with The System being built. Customers/Direct Beneficiaries interact with The System as well as the roles on the project level, but they operate within the project domain and the organizational domain (or external to the organization). Upper Management operates within the organization level and usually does not directly interact with The System, but does communicate with all roles within the project domain.

Now that the main stakeholders within RT have been established, it is important to understand their needs and requirements of the RT process. Each role has a unique set of goals that need to be met on order to benefit from the RT process and gain any value from it.

Based on the stakeholders proposed in the figure above, the following value propositions were derived for requirements traceability:

Upper Management Value Propositions

One of the main roles of upper management is to keep the project sponsor or customer happy with the details of the project that the organization has undertaken for them. This can include schedule status of the project, quality, cost, or any technical details for which they are ultimately responsible. All of these aspects are usually based on the assumption that the organization has limited resources to get the job done, and that needs to be taken into consideration to help minimize costs and deliver the system on time. Therefore for requirement traceability activities, these limited resources need to be taken into consideration. Upper management is also accountable for any external constraints that need to be satisfied, such as government mandates or regulations.

The main goal that upper management is concerned about is to ensure that the customer and sponsor, whether internal or external, will receive a system that is complete and meets the customer's needs. This completeness can be based in part on verifiable and documented links from the requirements that were originally given by the customer or project sponsor to the designs, code, and tests that verify that the requirements were met. Furthermore, from upper management's standpoint, it is necessary that the occurrence of missing an explicit or derived requirement is minimized. This is especially crucial in safety-critical systems for unaccounted for requirements or unverified requirements can result in injury or loss of life.

Upper management can also rely on RT as a mechanism for estimating the size and budget of a project, maintenance effort, or re-engineering activity. Traceability on historic projects can be used to determine the size and effort of similar projects for future enhancements based on the time and effort to implement the artifacts within a system based on the traceability information. Finally, RT can also provide upper management with a manner in which to measure progress within a project. If the traces to code, tests, etc. are granular enough, tasks can be assigned and completion status can be measured based on the requirements being addressed as they are implemented in code [Ramesh 95, Ramesh 01].

Customer/ Direct Beneficiary Value Propositions

Customers want to get the best system for their money with all or as many features as possible based on a limited budget usually as soon as possible. Using the example of help desk software, the system has to be useful to the customer (Tech Support staff) and the person on the other end of the phone (direct beneficiary) who will have a ticket created and tracked in the system. If new features are required or existing features evolve, customers and direct beneficiaries need to know that they have the flexibility to do so, and the system that is delivered will reflect the changes. Customers need to know that every requirement that they have put forth has been addressed in some way, especially with critical systems. Failure to properly implement requirements in those situations can result in injury or loss of life [Boehm 00a, Boehm 03a].

In some cases the organization at hand is building a system that will be handed over to the customer at some point near the project's completion. In these situations, the customer then has a need to understand the workings of the system they are about to inherit and maintain. To meet

this need, RT can provide proactive insight into how the artifacts of a project and the requirements have been implemented and are linked together. For example, the customer can use techniques such as Scenario-Based traceability in order to at least understand how code and tests are linked, or even IR techniques to retrieve links between system artifacts and requirements in a reverse-engineering capacity [Ramesh 95, Ramesh 01].

Project Management Value Propositions

A project manager needs as many methods as possible to show that they are in control of the project. Above and beyond standard measurements of cost and schedule, they are normally the ones who are responsible for minimizing project costs and balancing the work of limited resources.

On a more granular level, they need to be able to constantly verify that the stated requirements are reflected in design, implemented in code, and verified in tests. They also have the same responsibility as upper management to ensure the project, code, etc. complies with external constraints such as policies or government mandates. While upper management uses the same information to show milestones have been met, project managers can use traceability information on more of a daily basis to measure daily progress. For example, if a certain set of requirements compose a package in a work breakdown structure, traceability information can be used to measure % complete for the work package. If there are 10 requirements in the package, and only 5 of them can be traced to code, then the project manager can assume the package's implementation is 50% complete. If certain developers are assigned to implement the requirements, then the project manager can understand who has completed the 50% of the package, and who is responsible for completing the rest. At this granular level of control, derived requirements can become more apparent and therefore can be added to the system.

Project Management also needs the ability to perform impact analysis. One of the duties of a project manager is to manage change requests and scope creep. For example, RT can allow a project manager to understand the impact and trade-offs of new or changed requirements by using techniques such as EBT [Ramesh 95, Ramesh 01, Cleland-Huang 03a].

System Architecture/ Engineering Value Propositions

System architects or software engineers need to be able to link code with design and documentation. This enables them to know where and how requirements are implemented within the system. Also, since architects and engineers develop the designs and code that compose a system, they need to fully understand how changes are reflected throughout a system. Traceability can satisfy that need by providing links that can show the dependencies and implementations of design within a system that may not readily be evident.

Another need that engineers have is to be able to trace source code to test cases and scenarios. This ensures that an engineer's code will be completely tested, for their source code can be linked to a test. As code is implemented, the links to tests can be assembled for regression, performance, or other testing purposes. An engineer can also use this information for status reporting for project management in order to show progress and that their code has been accounted for in the system test plans [Ramesh 95, Ramesh 01].

Testing/ Quality Assurance Value Propositions

Testing/ QA personnel need to be able to link source code to test cases to ensure that all the source code has been tested. Furthermore, QA needs to be able to link the source code back to the system designs and requirements to verify that they are truly testing all aspects of a system. As a result, the tester can show backwards traceability (backwards to requirements) for the tests that they execute to show that a test implemented sets of source code as outlined in a system design that was derived from system requirements.

Another requirement for QA is to be able to manage changes in system implementation as they reflect on test plans. For example, if an engineer changes variables in source code, QA needs to know which test plans are affected by the change. Finally, testers need to report status of the execution of test plans to project management, similar to system engineering and architecture. This method of status reporting using RT can show which parts of a system have been tested and which have not [Ramesh 95, Ramesh 01, Boehm 00a, Boehm 03a].

5.1.1 Conflict Analysis

It would be nice to believe that the various stakeholders mentioned above could work in concert to achieve their goals most of the time. While several synergies do exist for the needs of upper management, project management, customers, engineers, and QA, conflicts can also arise when trying to meet the needs of all the RT stakeholders. If these clashes are understood, they can then be mitigated or avoided altogether which can help reduce the amount of conflicts. Otherwise, the clashes that do develop which are not proactively understood can increase the chances of problems of failure of the initiative at hand [Boehm 00a, Boehm 03a].

In order to understand the needs and potential conflicts of the various stakeholders in the RT process, a Spiderweb Diagram can be used. This type of diagram easily shows the various stakeholders, their needs regarding RT, and the potential synergies and conflicts that can result on a project. Boehm, Port, and Al-Said identify a model-clash as an *"inconsistency among the assumptions of the various process, product, property, and success models"* that are employed on a project [Boehm 00a, Boehm 03a]. Using Boehm's definitions, the models are described below:

- *Process models* are any models used by developers in order to develop a system, such as the spiral model, waterfall, agile methods, or others [Boehm 00a]. These models have distinct requirements and best practices that need to be adhered to in order for the model or process to be useful and therefore successful.
- *Product models* are those that describe operational concepts such as actual requirements, architecture, design, and code [Boehm 00a]. As an example, Ramesh has developed reference models that can be used for requirements traceability that describe how architecture, design, and code fit into traceability activities. [Ramesh 01].
- *Property models* identify factors such as cost, schedule, performance, security, reusability, as well the acceptable levels and tradeoffs therein [Boehm 00a].
- *Success models* encompass system correctness, win-win, business case, or other approaches to identify the success of an activity or project [Boehm 00a]. Success models are used when identifying or validating the ROI of a project, or understanding when a user interface may meet all the needs of the end user [Boehm 00a, Boehm 03a].

These four models are independent of each other, but can be used as groupings for RT stakeholder needs to show how the various stakeholders can or cannot work together. Conflicts can then be addressed and resolutions identified so that the project will not be negatively impacted by these relationships. We have taken the analysis of the needs from each stakeholder identified in the Onion Model (Upper management, Customer/ Direct Beneficiary, Project Manager, System Architecture/ Engineering, Testing/ QA) and used their value propositions as the basis for our conflict analysis. The conflicts and opportunities identified here have been derived in a similar manner to the stakeholders and their needs: from personal experience as well as research [Ramesh 95, Ramesh 01, Boehm 00a]. Based on these needs, opportunities and conflicts, we have derived a Spiderweb Diagram for Requirements Traceability:

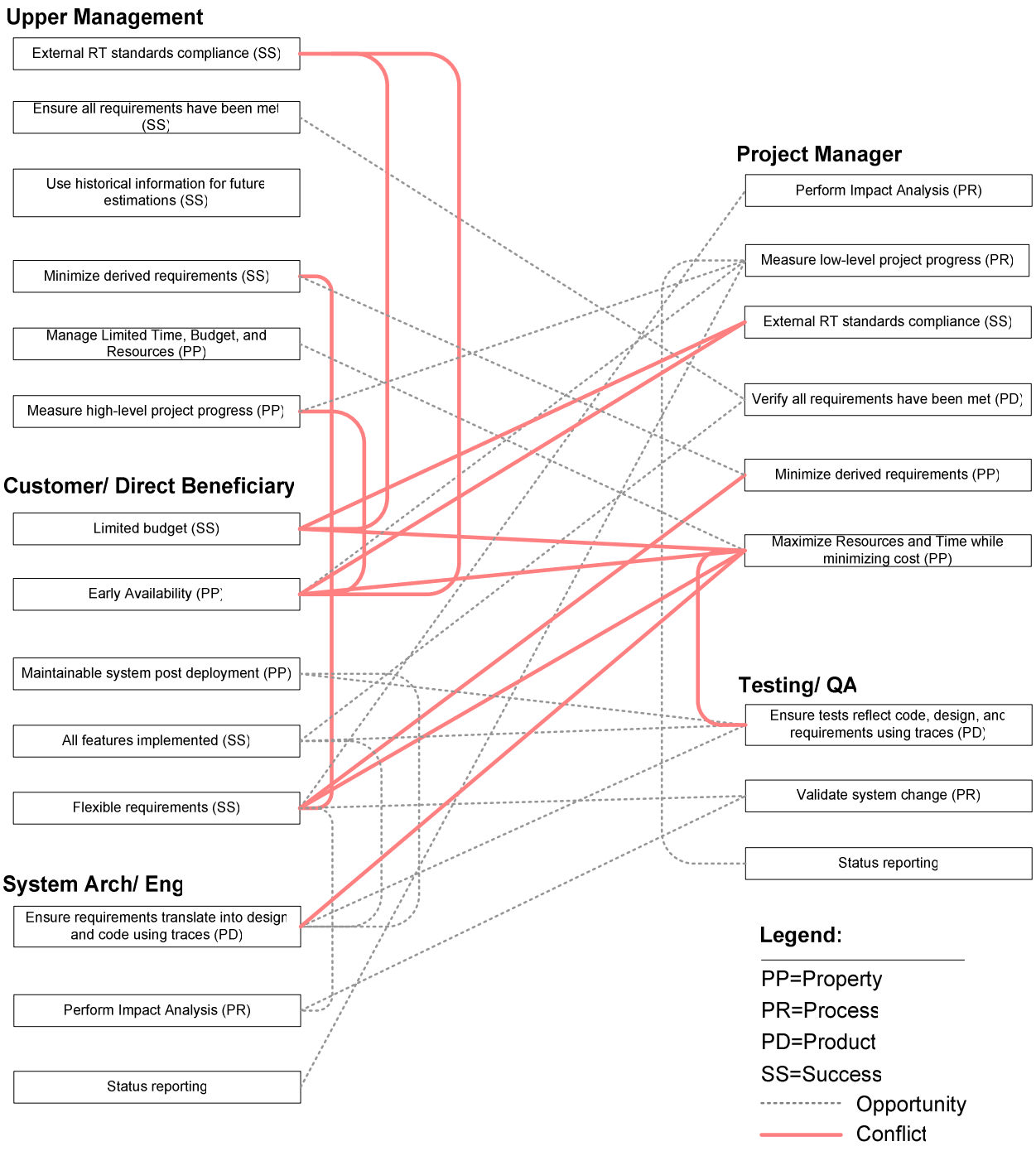

Figure 5-4. Requirements Traceability Spiderweb Diagram

After reviewing the Spiderweb Diagram, it can be seen that the majority of the conflicts that might arise during RT stem from the Customer/ Direct Beneficiary's needs. This is not surprising since RT is a labor intensive process that involves time and money in the form of resources to execute properly… all of which are usually not limitless on projects. While there are methods to save time and potentially money on a project as discussed in the *Business Case Analysis* process

(Section 5.3), the customers needs described here are generalizations of what an average customer wants. More specific needs unique to a project can be plugged into the diagram, and the relationships and problems can be identified from there. If any of the customers' needs end up being critical such as Early Availability (basically, limited time), this can be a signal to the project team to use as much automated RT as possible and forgo the accuracy of using a manual traceability matrix since manual activities are usually more time intensive. Further discussions on situational RT techniques and tradeoffs are explained in the *Business Case Analysis* process.

In order to understand the diagram, the relationships are notated from left to right, top down. The red lines notate a conflict in the needs of the stakeholders, and grey dashed lines show a synergy in their needs. As an example, the Customer Property Model to have "Early Availability" of a system is in conflict with the Project Manager's need for "External RT Standards Compliance". This is because complying with an external standard, such as a certain organizational CMM level that might be in place, could add additional time to the project, and not allow for early delivery of system. As another example, the System Architecture/ Engineering Product Model of "Status Reporting" has a synergy with the Project Management Property Model of "Measuring Low-Level Project Progress" since project progress can be better measured with adequate status reporting from the team.

Below is a summary of the main types of conflict within the diagram:

Property-Property Conflicts

These types of conflicts are inherent in RT. Property models describe needs that relate to time, budget, resources, and the like. These are major drivers for successfully executing RT. It has already been shown that the more automation there is in RT, fewer resources are needed, therefore, less cost and higher accuracy. Conflicts can arise as customers require more flexibility with the requirements for a system. This can increase the chance that derived requirements will be created by the project team [Ramesh 01]. If it is understood that this relationship and conflict can exist, both sides (customers and project team) can address the issue at the onset of an initiative to mitigate the conflict.

Property-Success Conflicts

Customers usually want the best system at the lowest price as fast as possible. Therefore, conflicts that arise in this are stem from cost, schedule, and resource needs that collide with factors that are needed in order for the project to be considered a success. For example, Upper Management and Project Management need to comply with any external laws, policies, or standards imposed by the organization, the customer, or even the government in certain situations. Compliance can add time and money to a project, which can conflict with the customer needs of Early Availability and Limited Budget.

Process-Property Conflicts

The main source of this conflict is between Project Management and the various teams on the project. Project Management maintains many Property Models on a project such as the budget, time to complete activities, as well as the resources. This is in direct conflict with QA and System Engineers need to actually use RT for artifact linking, verification, and the like. These activities take time and resources to execute them. It is up to Project Management to strike the

right balance of using RT while staying within the time and budget constraints for which they are accountable. The financial aspects of using various RT techniques on a project are explained in the *Business Case Analysis* process (Section 5.3). However, developing a process for guiding project-level RT decisions as described in [Domges 98] is beyond the scope of this paper.

As a final note, the relationships within the diagram are not binary. Opportunities and conflicts can both be relative. We realize that some conflicts can be stronger than others, depending on the project and the organization. Opportunities can also create great synergies, or simply be common understandings within certain companies.

5.2 Benefits Realization Analysis

In most organizations, management needs very accurate yet concise information in order to make the best decisions on which projects or initiatives to consider undertaking. No longer can sponsors of ideas simply go to management as if they were a bank and withdraw money for their "pet project". This is especially true since the dot-com bust. Managers have to prove that their initiative is worthy of investment by the organization by proactively understanding the benefits an initiative will bring before money is invested. One way this is done is by a Benefits Realization Analysis (BRA) [Boehm 03a, Thorp 03].

A Benefits Realization Analysis, as its name suggests, outlines the benefits that an initiative will bring to an organization. At this point in the VBRT framework, there is no quantification or qualification of the benefits nor any concept of trade-offs between the benefits. That is outlined in the *Business Case Analysis* process (Section 5.3). A BRA is simply a high level outline of what the potential benefits of an initiative will bring based on its alignment with organizational goals and the value propositions of the stakeholders identified in the *Stakeholder Value Proposition Elicitation and Reconciliation* process (Section 5.1). The results of this are twofold. First, it increases the understanding of all parties as to what an initiative will really bring to fruition. Second, it can help identify any additional stakeholders that may need to be involved in the realization process in order to make the analysis more complete [Thorp 03].

Requirements Traceability itself has several high-level benefits that can be brought to fruition. While the benefits are not quantified at this point, they can at least be identified and their relationships to each other can be understood. Once all of the benefits and their relationships are defined, then the ultimate goals of an RT initiative will become visible as will be shown in the next section.

5.2.1 The Benefits Realization Approach Results Chain

In the book *The Information Paradox*, John Thorp defines a simple framework for developing a benefits realization analysis that is easy to use and understand. It is called the Benefits Realization Approach Results Chain. This type of diagram outlines all the initiatives, sub-initiatives, assumptions, and outcomes of those initiatives in a graphical format. By using this type of map, it is easy to show what benefits can be derived from utilizing RT on a project. The Results Chain itself does not show how the benefits can or will be realized. It just shows what

they actually are [Thorp 03]. However, the following discussion will show, based on several studies, how these benefits can come to fruition.

The figure below illustrates our proposed Requirements Traceability Results Chain. This model can be modified based on specific needs or unique situations. This model shows a high-level overview encompassing the leading benefits that have been realized in several case studies and observations in industry which will be explained next.

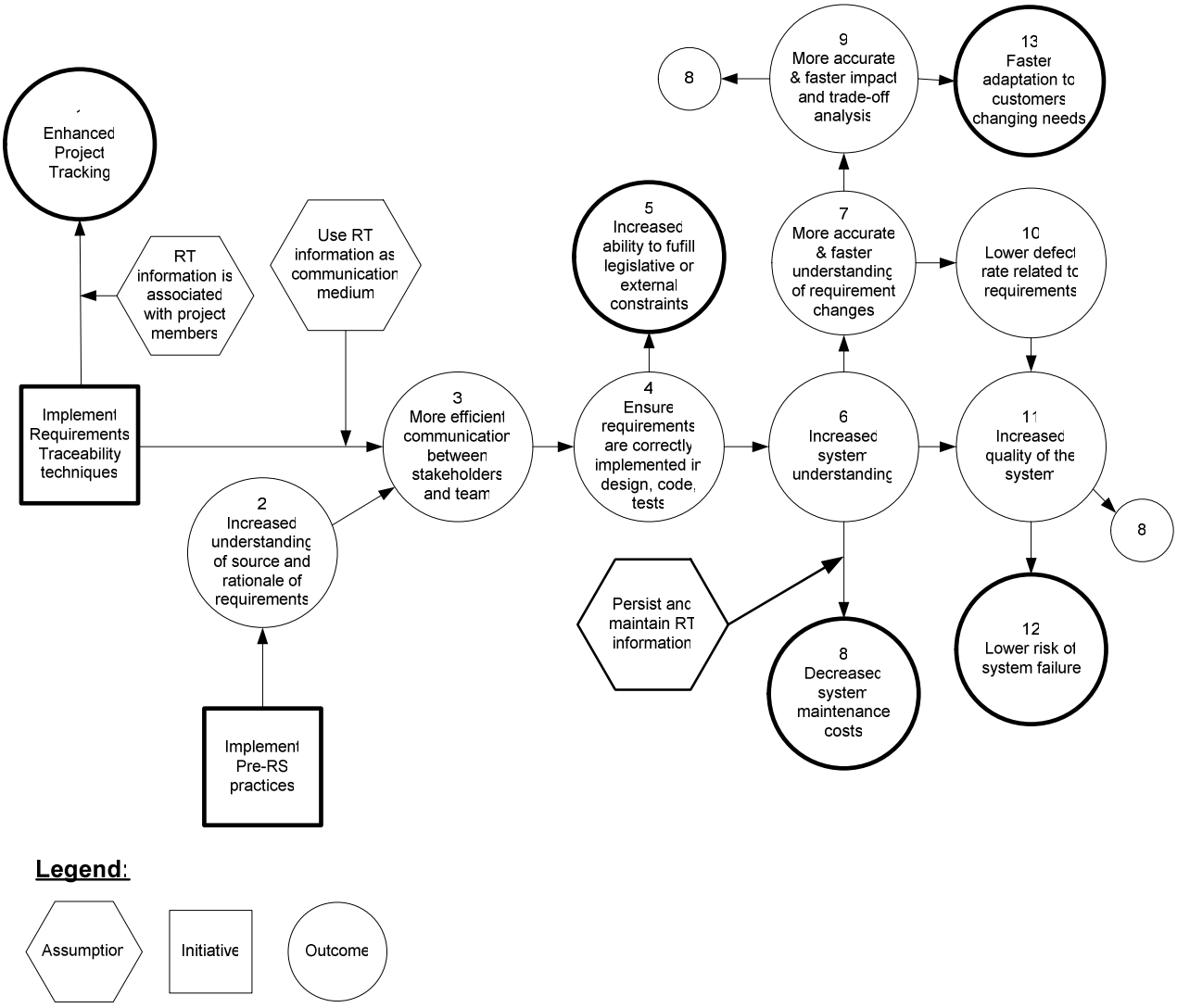

Figure 5-5. Requirements Traceability Results Chain

The Benefits Realization Analysis Results Chain is just that… a chain of results that lead to ultimate outcomes. The chain is read from left to right and begins with the initiative represented by a bold square. Initiatives are the actual actions or events that will be kicked off to ensure the benefits in the diagram are realized [Thorp 03]. Keep in mind the diagram describes the benefits,

or the "what", from a high-level. It does not describe the details of how these benefits can be realized since the main consumer of this model is Upper Management. That being the case, the model should remain high-level, yet detailed enough to give a broad understanding of the topic.

As can be seen, each initiative can have one or many outcomes that can lead to other outcomes. Outcomes are desired or inherent results of implementing an initiative (outcomes are synonymous with goals) [Thorp 03]. Outcomes should describe an end result, such as "Increased …" or "Reduced…". As various outcomes realize additional ones, other initiatives can start along the way to support them. Ultimately, the chain can end in one or many final outcomes, which usually represent the main goals or desired states of the initiatives that have been undertaken. These final outcomes are represented by bold circles. We derived the final outcomes from "technical" aspects of RT, but cast them in the light of a business benefits that can be measured by upper management who is the ultimate audience of the BRA Results Chain. As an example, a CIO might not understand or care that certain linkages on a project are correctly implemented (Outcome #4) between the design of a message queue and the code that implements it. However, the CIO would understand if this was said to be necessary in order for the project to adapt to customers' changing needs faster (Outcome #13).

Outcomes can link to other outcomes in different parts of the chain, which is shown by direct arrows or smaller circles with only a number in it. In order to be most effective, outcomes should not become too granular since they can be used as benchmarks on executive reporting dashboards or milestone charts that measures the progress of realizing the outcome. Too many granular outcomes can become too unwieldy to measure in some environments. More on this will be discussed in the Value-Based Monitoring and Control section.

Finally, assumptions can come into play along a path to help support the next outcome in the chain. They are necessary when it may not seem apparent why an outcome may link to another outcome. Or, they can help support why a proposed outcome may exist at all [Thorp 03]. As an example, if the application development manager does not assign RT information to her development team and associate the two, e.g. via a project management tool, it will be rather impossible for her to use it as a project tracking facility, and therefore the benefit will not be realized.

As the diagram shows, these outcomes should be realized in the given order. However, this does not suggest that the predecessor benefits must be 100% realized before the next is even started. Work on several outcomes can occur simultaneously to realize the end benefits, and there are many paths to the same final outcome. As an example, the "Decreased System Maintenance Costs" (Outcome #8) final outcome can be realized via the following paths of outcomes:

- 2→3→4→6→7→9→**8**
- 2→3→4→6→11→**8**
- 2→3→4→6→7→10→11→**8**
- 3→4→6→7→9→**8**
- 3→4→6→11→**8**
- 3→4→6→7→10→11→**8**

In order to understand the above diagram, we will look at the initiatives, assumptions, and their resulting outcomes. Remember, final outcomes of implementing RT are denoted by the bold circles while intermediate ones are not.

5.2.1.1 Initiatives

The Requirement Traceability Results Chain is a high level diagram encompassing the main outcomes of RT in general. Based on previous research, we saw RT was composed of two main concepts: Pre-RT and Post-RT, which is how we derived the two main initiatives for the BRA. While both are initiatives in the Results Chain, we feel that the main initiative for RT is "Implement Requirements Traceability techniques" which represents the research described in Section 3.2. In this case, this refers to implementing Post-RT practices in general, whether they are manual in nature, or dynamic such as Information Retrieval or Event-Based Traceability.

The other initiative is "Implement Pre-RS practices" which represents the research described in Section 3.1. In the Results Chain above, this initiative specifically refers to using contribution structure analysis and capturing rationale.

5.2.1.2 Outcomes

There are 13 outcomes or goals that compose the Results Chain. Each outcome can only be fully realized is its predecessors are also realized. In this way, each outcome leads into the next. As an example, in order to realize Outcome #2, "Implement Pre-RS practices" must be applied to the project. The Outcome "More efficient communication between stakeholders and team" can be realized without Outcome #2. However, it wouldn't be as completely or comprehensively realized as it would if Outcome #2 also came to fruition. The end result might be that there may be problems with realizing benefits down the Result Chain if Outcome #2 never happened.

The Requirements Traceability Results Chain outcomes are described below. While all Outcomes are described here, the Final Outcomes are further fleshed out as business benefits in *the Business Case Analysis* section:

Final Outcome #1: Enhanced Project Tracking
This goal describes methods that, used in conjunction with requirements traceability, can be used to better track progress or other project-related areas of an initiative. However, in order for this outcome to be realized properly, there is a noted assumption that RT information must be associated with project members, especially on a granular level. This benefit is further explained in the *Business Case Analysis* Section 5.3.4.1.

Intermediate Outcome #2: Increased understanding of source and rationale of requirements
This is the first outcome of the "Implement Pre-RS practices" initiative. RT can be a mechanism for capturing source and rationale information about requirements, and as such can allow the project team to better understand these types of data. This goal was derived from the research outlined in Section 3.1, and further information about this outcome can be found there.

Intermediate Outcome #3: More efficient communication between stakeholders and team

This is the first outcome of the "Implement Requirements Traceability techniques" initiative. Research has shown that RT links can be an effective communication medium between the project team and customers/end users of the system, as well as between project team members themselves [Domges 98, Jarke 98, Ramesh 98, Ramesh 01, Higgins 03]. If this medium is enhanced by rich information about the source and rationale of requirements by pursuing Outcome #2, then this outcome can be fully realized.

Intermediate Outcome #4: Ensure that requirements are correctly implemented in design, code, and tests

The technical goal of RT is to link the various artifacts of a system to each other. In a forward direction, traceability allows the project team to properly implement requirements into a system's design, the design into code, and the code into tests. In a backwards manner, the systems viability can be verified by ensuring that tests have been completely representative of the code, and the code can then be verified in design [Ramesh 01, Weigers 03, Watkins 94]. It make sense that a project team needs to first have efficient communication between the project stakeholders and the team itself (Outcome #3) in order to ensure that requirements have been correctly implemented. It also stands to reason that if the source and rationale of requirements are also understood (Outcome #2), then this outcome can become even more of a reality.

Final Outcome #5: Increased ability to fulfill legislative or external constraints

This goal very important to those organizations who must comply with certain traceability standards on their projects. Usually, this is mandated by clients or governmental regulations such as the Sarbanes-Oxley Act [SOX] or DoD Standard 2167A, which states *"that the functional requirements which are identified as a part of the functional baseline be traceable directly to specific capabilities within the allocated baseline, which must then be directly traceable to specific capabilities within the product baseline"* [DoD, Ramesh 95]. This outcome can only be realized if the organization can successfully begin to implement Outcome #3 and Outcome #4, and ideally Outcome #2 as well. This benefit is further explained in the *Business Case Analysis* Section 5.3.4.2.

Intermediate Outcome #6: Increased system understanding

Once Outcome #2 and #3 have been realized in some manner, the system can then be better understood by both the project team and the stakeholders in general. Research has shown that using traceability, especially when Outcome #2 has been taken into account, greatly increases the general understanding of how a system works since artifacts are or can be linked to each other. Gaps in implementation, perhaps between a design and the code the implements it, can be found more easily as well as sources of potential improvement [Jarke 98, Ramesh 01, Weigers 03, Bianchi 00].

Intermediate Outcome #7: More accurate and faster understanding of requirement changes

It has been shown that traceability links can greatly assist in managing changing requirements by using them to propagate change throughout a system. While this can be done without using RT by using brute force analysis, it has been shown that RT can increase the speed of analysis and understanding of potential changes to the system [Cleland-Huang 03a, Jarke 98, Ramesh 01, Lam 98, Strens 96]. For example, if there is a change in the design of a system, links can be used to understand which pieces of code need to be modified to reflect the new design. Managing

change in this way can only be realized if the project team has a adequate understanding the system (Outcome #6) by having correct traceability links between design, code, and tests (Outcome #4 and it's predecessor Outcomes).

Final Outcome #8: Decreased system maintenance costs
If a project team has already implemented requirement traces (Outcome #4) and through that effort has gained an increased understanding of the system being built (Outcome #6), RT information is already being captured and used by the project team as a documentation medium. In this context, traceability links can be used as a persistent method to document a system and its relations between its artifacts. These linkages allow the knowledge of system artifact and rationale relationships to persist until the system is no longer needed. Based on our experience in industry [CNA], project teams can have a ready source of system documentation which can be used to troubleshoot future problems the system may have in a faster and more efficient manner. System maintenance costs can also be lowered if a higher quality system is developed (Outcome #11) due to implementing traceability. If the project team uses RT as an impact analysis mechanism to understand future changes as the system evolves (Outcome #9), costs can also be reduced. This benefit is further explained in the *Business Case Analysis* Section 5.3.4.4.

Intermediate Outcome #9: More accurate and faster understanding of impact and trade-off analysis
In the same way the links are used in Outcome #7, they can also be used to understand artifact relationships for impact analysis. This is very beneficial during times of change, or even brainstorming new ideas for a system [Ramesh 01, Cleland-Huang 03a, Ramesh 95, Knethen 02, Bianchi 00, Lam 99]. Once again, managing change in this way can only be realized if RT is being used in a change management manner, and the project team has a adequate understanding the system (Outcome #6) by having correct traceability links between design, code, and tests (Outcome #4 and it's predecessor Outcomes).

Intermediate Outcome #10: Lower defect rate related to requirements
As previously explained, proper requirement management techniques have been shown to reduce requirement-related defects [Boehm 01a, Shull 02]. If a project team better understands the system they are building (Outcome #6) due to implementing RT (Outcome #4), and they can understand changes to the system in a more accurate and faster manner (Outcome #7), then it stands to reason that requirement-related defects may be lowered, especially during times of requirement volatility.

Intermediate Outcome #11: Increased quality of the system
Research has shown that if teams have a better understanding of a system (Outcome #6) and the various artifacts of the system are properly linked (or can be accurately linked) (Outcome #4), then the overall quality of the system can increase [Domges 98, Weigers 03]. This outcome can obviously be further realized if the system has a lower defect rate related to requirements (Outcome #10).

Final Outcome #12: Lower risk of system failure
This final outcome is a direct result of its predecessor, Outcome #11. Based on personal observation and experience, we have seen that if the quality of a system increases the risk that it

can fail decreases. This is not to say that risk is eliminated, just reduced. Therefore, we decided to make this a final outcome after Outcome #11 since many organizations addresses business decisions in terms of risk and we felt that quality could be broken down into this outcome. This benefit is further explained in the *Business Case Analysis* Section 5.3.4.5.

Final Outcome #13: Faster adaptation to customers changing needs
This final outcome is derived from the idea that if a project team can perform requirement change management (Outcome #7) and impact analysis (Outcome #8) faster and more accurately, then the project team can respond to the customer faster. We determined this to be a final outcome for two reasons. First, it ties directly into the Customer Perspective of the Requirements Traceability Balanced Scorecard (Section 5.6.1). Second, we felt an end goal of requirements traceability is to adapt to changing needs, using change management and impact analysis simply as mechanisms to accomplish this final outcome. This benefit is further explained in the *Business Case Analysis* Section 5.3.4.3.

5.2.1.3 Assumptions

The following assumptions support the realization of the benefits in the BRA Results Chain. They must be considered true in order to properly realize the benefits they pertain to. There are three assumptions tied to outcomes in the RT Results Chain. The outcomes and their associated assumptions are described below:

Final Outcome #1: Enhanced Project Tracking
The assumption that *RT information is associated with project members* must be held true in order to for team-level project tracking date to be useful, especially on an individual level. Otherwise, the data could not be associated with project members, and therefore would be useless in this context.

Intermediate Outcome #3: More efficient communication between stakeholders and team
The assumption to *Use RT information as a communication medium* must be held true in order to maintain efficient team to stakeholder communication. The premise of Outcome #3 is that RT can be used as a method of communication, so if this assumption were false, then there would be no communication medium, and therefore inefficient communication.

Final Outcome #8: Decreased system maintenance costs
The assumption to *Persist and maintain RT information* must be held true in order to use RT linkages as a documentation medium, similar to the assumption for Final Outcome #1 above. If RT information was not persisted, then defect investigation and changes made during the maintenance phase of a system would all have to be done via brute force analysis which can be time intensive and error prone. As a result, system maintenance costs might not be decreased.

5.3 Business Case Analysis

Software engineering practices are increasingly being viewed as investments that have benefits, costs, and risks associated with them [Boehm 03a]. A business case can be used to help show or justify the expenditure of money for an effort, such as RT. Business cases are materials prepared

for decision makers to show that a business idea being considered is viable for the company and that the financial data surrounding it makes sense. If it does not make sense or does not provide financial value to the organization, then upper management can justify not engaging in the business idea [Reifer 01].

A business case can take many forms from company to company. The purpose of this section is not to delve into the details of creating a fully fleshed business case, for the qualitative aspects of a business case for RT has already been described in [Cleland-Huang/Zemont 04]. Instead, we will perform a quantitative analysis on the four RT scenarios from that paper and show how they relate to the benefits that have been outlined in the Benefits Realization Analysis Results Chain (Section 5.2.1).

The four scenarios from the business case are as follows:

5.3.1 No traceability coverage

This alternative is often called the "do nothing" scenario [Reifer 01]. If the organization did not use formal RT in any way, this alternative would be the result. In this approach, links between artifacts and requirements are not documented and the associations between them are maintained in people's heads.

5.3.2 Partial matrix coverage

This alternative reflects users only keeping and maintaining links for critical or high visibility requirements using a manual traceability matrix. All other requirements are not linked or maintained.

5.3.3 Complete matrix coverage

This alternative represents total link coverage of all possible requirements, designs, tests, and code using a manual traceability matrix. In essence, this is the "perfect world" scenario where the organization has as much money and resources that they need in order to maintain a complete traceability matrix.

5.3.4 Heterogeneous traceability

As previously described in Section 3.2.5, this alternative represents using the proper blend of manual traceability and dynamic techniques (EBT, IR, Scenario-Based) in order to establish proper levels of traceability based on the requirement's criticality, volatility, and amount of risk to the system.

In summary, a business case must outline the costs and benefits of an idea as quantitative as possible. Qualitative concepts should mostly be used to support quantitative measures of the business case. This will be shown in the next two sections describing the costs and benefits of various approaches to RT within an organization.

5.3.5 Benefits of Implementing RT

By using the Requirements Traceability Results Chain (Section 5.2.1), we have already developed a baseline for the benefits of RT within an organization. To recap, here are the final outcomes from the Results Chain which will be the focus of the qualitative portion of our business case:

1. Enhanced project tracking
2. Increased ability to fulfill legislative or external constraints
3. Faster adaptation to customers changing needs
4. Decreased system maintenance costs
5. Lower risk of system failure

While these are all compelling arguments, they need to be justified in order for upper management to be able to make decisions about their value to the organization. Using the 5 main goals of the Results Chain, we have outlined each goal's relation to the four scenarios from the RT business case.

5.3.5.1 Enhanced Project Tracking

Enhanced Project Tracking refers to the ability to utilize RT information in order for upper management and project managers to understand progress on implementing requirements in a system. Traceability information can be used for tracking and projecting budget information as well as work in-progress during the project lifecycle. If granular information is kept on traceability links, such as which team member owns the implementation of a requirement, then the project manager can use this information to help determine % complete for a given set of features. If there is temporal information associated as well, the project manager can understand where the team is spending their time, and can help derive implementation schedules more accurately this way. For example, in a case study at a weapons lab, Ramesh observed a project manager associating sets of links between requirements, design and source code with a specific developer [Ramesh 95]. The project manager would then generate a weekly status GANTT chart from traceability information that charted the teams progress. In the same way, by using RT information associated with project members, it is possible over time to derive manpower needs and estimate project costs by observing the time and effort it takes to complete work related to certain links or sets of links [Ramesh 01, Ramesh 95, Ramesh 98]. This is of enormous benefit to the customer for better cost control can mean cost savings or avoiding cost overruns, and better schedule control may result in earlier project deliveries, or better estimations of actual project completion.

If the organization has any data or information relating project cost or schedule overruns due to inaccurate or not detailed project tracking data, then a case can be made for quantifying this benefit. This can be done by estimating the increased accuracy of project tracking data or a better ability to forecast project status via RT and translating it into the financial benefits of better cost or schedule control. For example, if an organization knows that 10% of projects exceed their financial or schedule limits due to inaccurate detailed project tracking information, they may be able to quantify that if they were to implement formal RT, the % of projects exceeding limits would be reduced by, say, 5%. In this way, they can avoid the potential cost of financial

penalties by delivering projects late or over budget, or even avoid the cost of losing customers completely. For more on the how the benefit of this aspect can be measured, see the *Value-Based Monitoring and Control* process (Section 5.5).

Obviously, this can be done for most RT scenarios to some degree. In order to model this relationship between techniques and benefits, we developed the "Degree of Realization" concept. For the given benefit, in this case Enhanced Project Tracking, each scenario quantified in the business case has a qualitative measure of a High, Medium, or Low Degree of Realization as it relates to the benefit, as well as a Description to explain the relationship.

Table 5-1. "Enhanced project tracking" Degree of Realization

Scenario	Degree of Realization	Description
No Traceability Coverage	None	If there are no links established at all on the project, this benefit cannot be realized.
Partial Matrix Coverage	Low	Since only critical requirement links are maintained, this benefit can only be realized for critical requirements.
Complete Matrix Coverage	High	Obviously, this scenario has 100% matrix link coverage, so every artifact has links that can be used to track project progress.
Heterogeneous Traceability	Medium	The manual links can have granular information associated with them. However, the dynamic links are usually created on an as-needed basis, so their lifespan may be shorter than the project, and therefore may not be of optimal use for project progress tracking.

5.3.5.2 Increased Ability to Fulfill Legislative or External Constraints

Certain organizations must comply with certain traceability standards on their projects. Usually, this is mandated by clients or governmental regulations such as DoD Standard 2167A [DoD, Ramesh 95]. Compliance due to external forces can also arise from an organization trying to attain the next CMM level during a process improvement initiative. In both scenarios, implementing RT can have huge benefits. For example, by implementing RT, these organizations can do business with the DoD or other government agencies when they may not have been able to previously. If they are already doing business with the government, they can avoid fines or loss of contracts by implementing RT.

As a result, quantifying this benefit is rather straightforward. In the example of doing business with the government, the cost avoidance of potential financial penalties for not using RT as stipulated in a contract can be quantified and weighed against the costs of employing RT. For CMM initiatives, if traceability is not put into practice, then the organization might not be able to move to the next CMM level, which would negatively affect its ability to get contracts where a

certain CMM level may be required. The cost of potential contracts that cannot be bid on as a result of non-RT compliance can then be used in the business case.

Excluding the "No Traceability Coverage" scenario where no RT practices exist at all, all other scenarios should be able to fulfill the realization of this benefit. This is because the degree or method of traceability is not often dictated by the DoD or the CMM, but simply that RT practices are established and followed. How each scenario relates to this benefit is outlined in the table below.

Table 5-2. "Increased Ability to Fulfill Legislative or External Constraints" Degree of Realization

Scenario	Degree of Realization	Description
No Traceability Coverage	None	Not using RT obviously does not fulfill standards or regulations that may be imposed on an organization.
Partial Matrix Coverage	High	Using even partial RT can fulfill standards or regulations.
Complete Matrix Coverage	High	Using complete RT can fulfill standards or regulations.
Heterogeneous Traceability	High	Using heterogeneous RT can fulfill standards or regulations.

5.3.5.3 Faster Adaptation to Customers Changing Needs

During the course of a project, change is usually inevitable. Customers may change their mind on features in the final product. Requirements may get dropped or added throughout the lifecycle of the project at the customer's request. Additionally, the impact of each change must be assessed and taken into account. It is therefore critical that an organization is able to adequately adapt to the evolution of requirements throughout a project in order to satisfy a customers' needs For more information on the role of change in regards to RT, see the *Change as Opportunity* process in Section 5.6.

Quantifying this benefit can mainly focus on two areas: defect rates and impact analysis. An organization can measure the quantity of defects during a project due to changes not being adequately propagated throughout the projects' artifacts (and therefore the system itself) and measure that against the ability to propagate all changes as a system evolves during the project lifecycle. The effort to reflect changes can also be measured by comparing the manual effort of finding the touch points in a system where change affects various artifacts compared to the scenario of using RT techniques such as IR where the information of artifact associations is readily available or can be generated dynamically, and therefore the effort and cost of reflecting changes can be much lower. This idea can even be extended by measuring the manual effort of modifying test scenarios due to system changes, and comparing that to using a technique such as Event-Based Traceability or Scenario-Based Traceability where tests can dynamically pick up changed values for test cases in order to re-execute them. Furthermore, if Pre-RS techniques are used, such as Contribution Structure analysis, the rationale and source of changes can be

captured which can avoid the cost of system defects due to misunderstood requirements. If the source and rationale are known, project members know where to go in order to clarify information and requirements. For more on the how the benefit of this aspect can be measured, see the *Value-Based Monitoring and Control* process (Section 5.5).

Performing impact analysis is a key intermediate outcome for this benefit as per the Results Chain. By using techniques such as EBT, the impact of changes can be readily analyzed and decisions can be made in a timely manner as whether or not to implement the change. This can be quantified against the effort of performing impact analysis by manually analyzing artifacts in order to understand the impact of modifications, which can be much more costly and error prone.

In summation, here is how each RT scenario correlates to this benefit:

Table 5-3. "Faster Adaptation to Customers Changing Needs" Degree of Realization

Scenario	Degree of Realization	Description
No Traceability Coverage	None	Changes to customer needs and the relating impact of those changes are manually derived, which is more time consuming and error prone.
Partial Matrix Coverage	Low	Changes to customer needs and the relating impact of those changes are manually derived except for critical requirements, which is more time consuming and error prone.
Complete Matrix Coverage	Medium	Links can be traversed to understand impacts and propagate changes to the system. There is still manual intervention in the cases of having to re-execute tests based on changed data.
Heterogeneous Traceability	High	Links can be traversed to understand impacts and propagate changes to the system. By using techniques such as EBT, tests can be dynamically re-executed with little manual intervention.

5.3.5.4 Decreased System Maintenance Costs

After the project lifecycle is complete, the product that is built will continue to exist for some time as a tool for its end users and direct beneficiaries. During the maintenance phase of the product lifecycle, defects may be discovered or additional functionality may be added to fit the changing needs of the systems' users [PMI 00]. Since RT linkages and associated information are a readily available mechanism of system documentation, quantifying this benefit can center on the cost of time in order to update and manage the linkages and understand their relationships in times of change. If system artifacts need to be updated, the costs of manually searching for artifact associations can be weighed against the cost of using dynamic RT techniques such as

Scenario-Based Traceability or IR to quickly identify artifacts and the associated relationships that may relate to the changes at hand. Furthermore, the potential for avoiding errors by manual analysis can be factored in as well since documented linkages can show relationships right away.

Using this same thought process, another aspect that can be quantified for this benefit is the time to resolve and identify defects. An organization can measure the average amount of time to adequately and correctly identify system defects manually, and compare that to the potential time and cost of using dynamic RT in order to identify artifact relationships or defect correction. As a result, organizations can understand how to avoid the cost of time-consuming manual defect identification and the resultant brute force analysis of identifying the touch points in order to change the system to correct the defect. While manual links accomplish this, if there are changes associated with the defect, the relationships must be manually changed as well, which is more labor intensive than dynamic techniques and therefore more costly.

How each scenario is associated with this benefit is described below:

Table 5-4. "Decreased System Maintenance Costs" Degree of Realization

Scenario	Degree of Realization	Description
No Traceability Coverage	None	If RT is not used, maintenance and defect identifications/ system changes must be done by brute force analysis.
Partial Matrix Coverage	Low	Maintenance and defect identifications/ system changes that are not for critical requirements must be done by brute force analysis.
Complete Matrix Coverage	Medium	While the cost of maintenance and identification is low since there are linkages, changes due to defects are manually made.
Heterogeneous Traceability	High	Links can be traversed to identify defects and perform link maintenance. Re-establishing associations due to changes by defects can be accomplished by using dynamic RT techniques.

5.3.5.5 Lower Risk of System Failure

The concept of system failure can mean different things to different organizations. For an e-commerce website, it can simply mean a server crashes and needs to be rebooted. However, for safety-critical systems such as software that controls an airplane in flight or a heart monitor in a hospital, failure can result in severe personal injury. As previously shown through the other benefits in this section (*Decreased System Maintenance Costs*, etc.), RT allows project team members better insight into artifact associations for development and maintenance as well as provides better control of system changes and assessing impact based on customers' changing

needs. By considering all these benefits and outcomes, requirement traceability can be thought of as a risk mitigation tool against system failure.

Risk is usually described as the result of "Impact x Probability". "Impact" is the result on the system if the risk comes to fruition and "Probability" is the likelihood of the risk occurring in the first place. Within the context of lowering the risk for system failure, this can be thought of as reducing the probability that a costly and avoidable mistake will occur due to artifacts not being supported by traceability linkages. For example, changes to the system that has not been adequately propagated increases the probability of failure, and potentially the impact. Or, by not using Pre-RS techniques that allow understanding of requirement sources or rationale for their existence, the probability of system failure increases since project members have to interpret and extrapolate meaning from requirements themselves without being able to verify them against their source. However, if the risk does come to fruition, data associated with traceability links can be used to discover the source of the failure rather than simply using brute force analysis of the development artifacts in order to identify and correct the cause of the system failure. Therefore, this benefit can be quantified by assessing the cost of previous system failures due to misunderstanding relationships and data between artifacts of the system, or as a result of changes not properly propagated throughout the system artifacts. This can then be compared against the potential risk mitigation and cost avoidance of future system failures by using RT. Since the concept pf risk composes one of the seven processes of VBRT, further discussion of risk and RT can be found in the *Continuous Risk and Opportunity Management* process discussion (Section 5.5).

Table 5-5. "Lower Risk of System Failure" Degree of Realization

Scenario	Degree of Realization	Description
No Traceability Coverage	Low	There is almost no level of risk mitigation since there is no traceability between artifacts.
Partial Matrix Coverage	Medium	There is a medium level of risk mitigation since only critical requirements are traced. However, the level of risk mitigation is low for those requirements not deemed critical.
Complete Matrix Coverage	High	Since all requirements are traced throughout the system, there is a high level of risk mitigation against system failures. However, the level of risk mitigation is medium during system changes.
Heterogeneous Traceability	High	Since most requirements are traced on an as-needed basis, there is a high level of risk mitigation against system failures.

5.4 Concurrent System and Software Engineering

Over the past few years, businesses are aligning themselves with more agile system development lifecycle processes [SoftwareMag, Ware 04], even though there are others who are skeptical of the value of such methods and believe they need to be further proven in industry before they should be considered viable [Neill 03, Stephens 03]. This is due to several reasons including requiring faster time-to-market, more flexibility in times of change, increased reduction of development costs, and more productivity from a limited staff [Reifer 02, Ware 04]. As suggested by the VBSE framework, by using concurrent/agile methodologies, an organization can maximize the value of software engineering efforts [Boehm 03a]. Several process models have been developed over the past decade in order to help accomplish such concurrency, such as Extreme Programming [XP], the Crystal Methodologies [Crystal], Adaptive Software Development [Adaptive], the Rational Unified Process [RUP], and Scrum [Scrum]. These concurrent development processes usually result in faster deployment times, since portions of the system are released iteratively, which can provide value to the customer sooner than traditional sequential or waterfall methods [Boehm 03].

While these agile methodologies address many aspects of software engineering within their core descriptions and practices, they do not directly focus on the idea of using RT within an agile environment. Actually, the idea of tracing requirements seems to be anathema to most agile development processes since RT requires a static medium such as a requirements management tool or requirements specification that will contain the requirements of the system throughout the lifecycle of the project. Requirement management tools and traceability are often seen as not economically viable in an agile environment, and therefore discouraged since such methodologies promote people orientation, not process or tool orientation [XP, Paetsch 03, Manifesto]. As an example, Extreme Programming utilizes the concept of User Stories on note cards as requirements. Once the User Story has been implemented, the card is usually discarded since the knowledge of the requirement has been transferred to the software code, test cases, and programmer's memory [XP, Nawrocki 02]. Furthermore, requirements are not the focus of XP, the test cases are. As long as the test cases pass, the requirements that derived them are considered irrelevant at that point [XP].

At this point in time, there is a lot of discussion in the agile community about whether or not RT is a valuable activity for agile processes. Alaistair Cockburn, author and co-creator of the Agile Manifesto and creator of the Crystal Methodologies, believes that traceability is a waste of time and has offers no real value to an organization, based on his experience at clients [Cockburn 04]. Others, such as Brad Appleton, another author and co-creator of the Agile Manifesto, suggested that agile processes might be able to utilize requirements traceability by implementing the following ideas:

- Having tests cases directly traceable to user stories.
- Minimizing the number of artifacts produced (fewer items means fewer items and fewer inter-relationships to track and manage).
- Knowing which user stories each "commit" is for and being able to associate them in the version-control tool if necessary in order to automate code-level traceability (e.g., a "checkin" comment).
- Aggressively managing dependencies (encapsulation/refactoring)

- Using concepts of domain-driven design to ensure the easy correlation between stories and objects.
- Managing traceability at the level of user stories (feature or use-case) and objects instead of individual line-item requirements and lines of code. [Appleton 04]

This idea was enhanced by Mike Beedle (a co-author of the Agile Manifesto), when he suggested that agile methods do not adhere to requirements-driven traceability, but rather test-driven traceability through Aspects [Beedle 04a]. The response to this was an in-depth discussion that generated over 100 response postings [APM 04a]. It was even suggested by Mike Beedle to modify the Manifesto to include the concept of traceability [Beedle 04b]. This idea was promptly rejected by other practitioners and other co-authors of the Manifesto [APM 04b].

While the ideas posited by Brad Appleton are interesting, they are untested in the area of agile development. The idea of how RT can successfully fit into concurrent software development practices is an evolving topic. There has been some initial research into tools such as Echo, which is a tool that capture conversations, translating them into User Stories, which can then be traceable since the audio is translated to an XML-based repository [Lee 03]. Other research is trying to successfully introduce RT processes into agile methods [Nawrocki 02, Paetsch 03]. However, proving RT can add value in agile/ concurrent environments is outside the scope of this paper due to the fact that the methodologies themselves are still being proven in industry, there is a general lack of solid research on the topic, and the nature of the processes themselves currently rejects RT as a practice. The creators of the various agile processes are just now starting to consider the topic as the UseNet postings suggest. The idea may eventually get rejected altogether, but if not, this would be an interesting topic for future research.

5.5 Continuous Risk and Opportunity Management

Risk can be defined as *"an uncertain event that, if it occurs, has a positive or negative effect on a project's objectives"* [PMI 00]. This is also thought of in terms of probability of occurrence and its resultant impact upon the system. The result is usually a ranking or score to understand the scope of the risk compared to other risks. It has been extensively shown that risk identification, mitigation, and management are not only techniques that should be used up front during a project but throughout the project lifecycle [Boehm 03a, Conrow 97, Murthi 02, Boehm 91, Leishman 02]. As a result, risk has important consequences as to how a project will be executed. Some project members may be more risk adverse than others and will make different decisions based on those feelings, which will result in different outcomes. Since these decisions directly impact the project's success, VBSE risk and opportunity management can be thought of as a set of people-oriented practices that balance negative risk with positive opportunity considerations in order to maximize value for the project and organization as a whole.[Boehm 03a]

Risk can be thought of in the formula: Risk = Probability x Impact. Therefore, as stated by Cleland-Huang, Zemont et. al. in [Cleland-Huang/Zemont 04], the goal of VBRT risk management is to lower the probability *"that a requirement may be changed or impacted by a change in order to avoid the impact of potential damage caused by the invalidation of a requirement"*. It has been previously mentioned in the *Faster Adaptation to Customers*

Changing Needs section (Section 5.3.4.3) that change and volatility are usually inevitable in a project. Customers may change their minds on features, their might be a lack of stakeholder agreement on requirements, or their might be unstable market conditions that can cause requirement instability [Lesihman 02]. In terms of risk, every requirement has a probability of changing and has an impact upon the system if it does change. If the change is not properly supported by traceability links, it is known as a Failure to Trace Risk (FTR) [Cleland-Huang/Zemont 04]. This section summarizes the ideas of the FTR concept as put forth by Cleland-Huang, Zemont, et. al. in response to the question of how much traceability is enough to produce an adequate level of risk mitigation for changes on a project. These concepts serve as a basis to determine the level of risk exposure based on decisions on whether to not to implement traceability for a requirement or set of requirements [Cleland-Huang/Zemont 04].

Table 5-6. Impact and PC Score Ranges

Impact (I)	
4	Limited
6	Moderate
8	Substantial
10	Critical
Probability of Change (PC)	
0.3	Stable
0.6	Changeable
0.9	Volatile

A Failure to Trace Risk is composed of two variables: Impact (I) and Probability of Change (PC). By using these variables, a Failure to Trace Risk (FTR) score is derived as the following:

FTR = ((PC*2) +1)*I

In order to effectively use this concept, thresholds need to be established for the project or organization based on FTR scores. This way, appropriate strategies could be derived from FTR results. For example, consider the following requirements based on the Impact and PC Score Ranges table:

Table 5-7. Sample Requirements for FTR Example

Requirement	**Change**	**Impact**	**FTR**
Requirement A	0.9	10	28
Requirement B	0.3	10	16
Requirement C	0.9	4	11.2
Requirement D	0.3	4	6.4

Without having thresholds or parameters for FTRs, the numbers mean nothing by themselves. Therefore, there needs to be project-level strategies in place to understand that when an FTR falls

within a certain range, a specific set of traceability techniques need to be used. Such a strategy needs to take into consideration the fact that each requirement has a different FTR, and in order to mitigate that risk, all requirements cannot be traced with the same technique. This would bring back the trap of value-neutrality where all requirements are treated the same, when based on their volatility, criticality, and impact to the system, they are not.

As shown in [Cleland-Huang/Zemont 04], the best scenario and technique in which to address the proper treatment of developing strategies for FTRs is Heterogeneous Traceability. Based on certain FTR ranges, scores can determine how those requirements should be treated. For example, a traceability strategy might be set for a project as follows:

Table 5-8. Example FTR Trace Strategies <revise from published paper>

Risk Level	FTR Range	Traceability Strategy
Nominal	0-10	Trace these requirements using IR methods
Significant	11-20	Trace ~5% of the most critical requirements explicitly. Trace < 2% of Non-Functional Requirements (NFR) using EBT. Trace the remaining using IR methods and Scenario-Based Traceability.
Substantial	21-28	Trace ~50% of critical requirements explicitly. Trace ~5% of NFRs using EBT. All other requirements can be traced using IR techniques and Scenario-Based Traceability.

Therefore, Requirement D has a “Nominal” risk level. This requirement should be traced by using IR techniques. Requirements B and C have a “Significant” risk level. Therefore, if they are very critical, they should be traced explicitly via a manual traceability matrix or by using a requirements management tool. Or if they are NFRs, the project team can decide whether to use EBT to establish and maintain their traces or not. Finally, Requirement A is a “Substantial” risk level requirement. Most likely, this requirement should be traced explicitly. Otherwise, if it not an NFR, IR techniques or Scenario-Based Traceability can be used.

By using a table such as this, low-level traceability decisions can be made by project team members in order to mitigate the most traceability risk on a project. The FTR ranges can be modified as well based on a project team’s level of risk tolerance. The ranges can be extended for risk adverse organizations and teams, or narrowed for those who are risk tolerant. Even if the effort is too great to manage risk on a requirement level, this same concept can be used for sets of requirements as well. By using this method, most of the subjective nature of determining the proper technique for tracing requirements properly is reduced.

Over time, certain requirements may become less volatile and set in stone, the impact of certain requirements may change, or new requirements may be added or dropped from the system. As these changes occur, FTR scores must be re-evaluated and acted upon. Therefore, analyzing FTRs for requirements or sets of requirements is not just an up front activity of VBRT risk management, but rather an ongoing effort throughout the project lifecycle.

5.6 Change as Opportunity

In Boehm's model, C*hange As Opportunity* permeates throughout all of VBSE. There can be potential changes to any one of the facets of the model. New stakeholders might be introduced into the VBRT model, and their needs and potential conflicts need to be taken into account. Business goals may change and alter the strategy map outlined to implement the benefits of RT for an organization. Therefore, he recommends that organizations adopt practices that allow for more adaptability to change and view change as an opportunity for improvement, not as an activity to be avoided [Boehm 03]. RT is a great example of such a practice.

In today's system development environments, requirements change seems inevitable [Lam 98, Weigers 03, Boehm 03]. Changing requirements, and their mis-management, have lead to rework and project failure [Sawyer 99, Weigers 03] since requirement changes have a price associated with them. Doug Leffingwell from Rational Software (now IBM) showed the cost of system changes and errors throughout the course of projects based on studies at GTE, TRW, and IBM. In essence, the cost of fixing errors increases throughout the course of a project. In his example, he assigned an arbitrary cost of *one* to detect an fix an error in the coding stage of a project. Based on this assumption, the cost to detect and fix an error in the requirements stage of development was 5 to 10 times less [Leffingwell 97]. Others have observed that this estimation may be too low for larger projects where the cost of finding and fixing defects during maintenance as opposed to the requirements phase can be as high as 100 times more [Boehm 01a, Shull 02]!

Changes can originate from many directions. They can come from changing needs or desires of the customer of the system, from the environment surrounding the system such as policies or laws, from the design or technology of the system, or from the potential impact of the system to the end-users methods of doing business and interacting with the system itself [Lam 98, Strens 96]. Unfortunately, change is often seen as a negative activity that should be avoided. However, in order to be successful in software engineering change needs to be managed and accounted for. Adverse changes to a system need to be accounted for, and potential opportunities (such as an increase in profitability) from change need to be exploited [Boehm 03, Strens 96].

The increased ability to properly manage a system during times of change is an inherent property of RT. Change Management is a major aspect of the value that RT can bring to a project. That is why it is a final outcome of the Requirements Traceability Results Chain as described in Section 5.1. Links between artifacts, especially dynamic links, can assist in change management since associated artifacts with a change can be found as needed [Haumer 99]. For example, if code needs to be changed due to a design change as a result of a business requirement change, traceability practices can link all the appropriate artifacts together so the change can be accurately permeated throughout the system [Watkins 94]. While manual RT can accomplish this, there might need to be modifications in the linkages between artifacts due to the change, and that is more effort overhead for the team to manage. Therefore, as described in Section 5.3, a heterogeneous approach is preferred. Since linkages can associate the proper artifacts in times of change, RT is also an asset during impact analysis activities [Weigers 03, Knethen 02]. Since associated artifacts can be linked to each other, the manner and scope of potential changes can be understood. If Pre-RS practices are used as well, RT can help with change verification since the source and rationale of requirements would be understood and could be accessed as outlined in

Section 3.1 and [Haumer 99]. RT also ties change to risk management. As previously mentioned in Section 5.4, failure to establish a proper trace between artifacts can lead to artifact relationships not being understood in times of change in creasing the risk exposure of a project [Weigers 03, Cleland-Huang/Zemont 04].

The focus of this paper, in regards to requirements change management, is that managing change is a very integral benefit to the value model for requirements traceability. There has been extensive research on the topic of requirements change management and various models and processes have already been developed to manage requirements volatility, evolution, and cost of change [Lam 99, Strens 96, Haumer 99, Egyed 02, Lavazza 00, Lam 98, Knethen 02]. While these sources agree that a main value of RT is its ability to help with change management, some feel there still is much more to be done on this front to guide users through low-level requirements change management decision-making [Haumer 99, Lam 99]. However, it is outside the scope of this paper to present yet another model to manage requirements change in an opportunistic manner. Furthermore, the extent of the value of change management and RT has already been outlined in several aforementioned sections of this thesis (See Sections 3.2, 5.2.1, and 5.3.4.3).

5.7 Value-Based Monitoring and Control

It is necessary for project managers and upper management to be able to examine the progress of initiatives so they can better manage them. Ideas should be constantly validated against their goals in order to understand if they are performing to plan. In terms of VBSE, this means that aspects outlined in the *Business Case Analysis* (Section 5.3)and the results of a *Benefits Realization Analysis* (Section 5.2) can be used as milestones for measuring progress of the initiatives delivering business value to a project. These milestones can even be used to map out an implementation strategy for the initiatives. However, this can really only be successful if there are proper feedback mechanisms in place so that information can be collected, summarized, and corrective action can take place if the proper value is not being recognized [Boehm 03a].

In the Benefits Realization Analysis Results Chain for RT, 5 final outcomes were identified as well as a host of supporting intermediate outcomes. These benefits help support the overall initiatives to “Implement Requirements Traceability Techniques” and “Implement Pre-RS Practices”. While it is important to identify the outcomes that support the initiatives, it is also important to know when successful implementation has been achieved. If an outcome can be thought of as an organizational or project-level strategic initiative, how will the organization or project manager know they have achieved the outcome? How will they know that they are on track or not in implementing the outcome? And, how will they know the steps needed in order to realize the outcome in the first place? It is imperative that projects and organizations have mechanisms that allow them to map out and monitor the progress of strategic initiatives in order to make corrective actions so the initiatives are successful and provide the value that was intended. One of the most common methods to map, measure, and control strategic initiatives is the Balanced Scorecard (BSC) [Kaplan 92, Kaplan 93].

The BSC was developed at Harvard University by Robert Kaplan and David Norton in 1992. It has been used by small companies and large international organizations [Kaplan 92, Kaplan 93,

Grembergen 02, Mair 02]. Its development was in response to inadequate processes to identify and monitor performance measures within an organization. Up until that time, they found many mangers relied on only one set of measurements (such as financial measures) in order to gauge the health of the organization. Their idea was that no manager should have to rely on one set of measures at the expense of others. They thought upper management should be able to have the visibility and a balanced presentation of both financial and operational measurements in order to run their organizations or departments [Kaplan 92].

After researching 12 initial companies, they developed a "balanced scorecard", which is *a set of measures that give top managers a fast but comprehensive view of the business*. [Kaplan 92] There are four top down perspectives that allow managers to understand their business from an internal and external point of view. By using only these four perspectives, the BSC can avoid information overload by streamlining the monitoring and control process. Furthermore, by having only four generic categories, organizations can have more flexibility in defining their own detailed scorecard. The four perspectives and the questions they answer are as follows [Kaplan 92]:

1. Customer Perspective: How do customers see us?
2. Internal Perspective: What must we excel at?
3. Innovation and Learning Perspective: Can we continue to improve and create value?
4. Financial Perspective: How do we look to shareholders?

These four perspectives are applicable in the development of a balanced scorecard which can be used to both understand the goals of an RT initiative within a project or organization as well as measure the value of RT. As previously mentioned, within the four perspectives are goals and measures. By leveraging the outcomes put forth in the *Benefits Realization Analysis* process (Section 5.2), we derived the goals of a balanced scorecard. Furthermore, by understanding the quantification of benefits from the *Business Case Analysis* (Section 5.3), we developed the measures for those goals. We then encapsulated the four perspectives and their measures in the Requirements Traceability Balanced Scorecard (RTBSC). Just like the scorecard that Kaplan and Norton envisioned, the RT scorecard is not an absolute. Different markets, industries, and products require different scorecards [Kaplan 93]. As a result, organizations can take our RTBSC as a template to modify for their projects.

5.6.1 The Requirements Traceability Balanced Scorecard

5.6.1.1 The Customer Perspective

The goals that can be associated with the Customer Perspective focus on the top concerns of the customer (and Direct Beneficiaries), which usually revolve around time, quality (defect level), performance and service, or cost [Kaplan 92]. In the case of RT, the scorecard goals center around what the customer directly gains from implementing traceability. The outcomes that model this part of the scorecard are those that pertain to communication, change, quality, and level of risk as shown in the table below.

Table 5-9. RTBSC Customer Perspective

Customer Perspective	
Goals	**Measures**
More efficient communication between stakeholders and team	- Management and consultation time spent per project - Average time to verify requirement and rationale with source
Faster adaptation to customers changing needs	**- % accepted changes successfully implemented (defined by customer)** **- Schedule impact due to changes**
Lower defect rate related to requirements	- Requirement-related defect rate
Increased quality of the system	- % of system uptime - % of system failures due to traceability errors
Lower risk of system failure	**- RT Risk Index score: % of requirements properly traced based on FTR score**

Bold = Final Outcome

The goals and measures that are in bold indicate that they represent a final outcome from the Benefits Realization Analysis Results Chain, which is the source of the goals for the scorecard we developed. We derived the measures for the goals from the business case, as well as metrics that might pertain to the goal at hand. For example, *More Efficient Communication Between Stakeholders and Team* can be generally measured in terms of time spent on consultation efforts with customers and stakeholders, including management oversight and meeting time. Generally speaking, if communication is more efficient, then the time to communicate should be reduced. The other measurement looks at the time spent to verify or refine requirements. This goal directly ties back to the "Implement Pre-RS practices" initiative in the Results Chain. If a project team uses Pre-RS techniques, the communication can be made more efficient since the rationale and source of requirements is known to all team members. While the metrics for this goal are internally measured within the organization developing the system, we feel there are other measures that can only be defined by the customer, which makes sense for the Customer Perspective. One of the final outcomes in this part of the scorecard, *Faster Adaptation to Customers Changing Needs*, has a measure that looks at the number of accepted changes successfully implemented. The word "accepted" is critical here since during the course of most projects, many changes are requested of the project team, and due to mutual agreement or lack of time, they never get implemented. In this case, we do not care about the unapproved changes, but rather the ones that have been approved and which the customer is relying on to make the system more valuable to them. The other metric that this goal is concerned about is the impact on the schedule due to changes. If there is faster adaptation changes by implementing RT, then the cycle time for change request and change management in general should go down. Therefore, this is a worthwhile measure to consider.

Lower Risk of System Failure is the other final outcome in this perspective. As described in the *Continuous Risk and Opportunity Management* process (Section 5.5), this goal can be measured

by looking at the decisions around FTR scores, and making sure that the traceability decisions based on the project-level strategy for handling a certain score have been correctly implemented. If they have, then it can be said that traceability risk is properly being mitigated. If not, the corrective action might need to take place in order to ensure the proper technique is being used in order to help mitigate the risk of system failure.

Finally, the last two goals that we identified, *Lower Defect Rate Related to Requirements* and *Increased Quality of the System,* are rather straightforward. The defect rate goal can be measured by analyzing the defect rate related to requirements. The goal for increased quality can be measured by the percent of system uptime/downtime and analyzing how much system downtimes or defects are related to traceability errors.

5.6.1.2 The Internal Perspective

Internal Perspective goals center on measuring the internal business processes of the organization that have the greatest impact on customer satisfaction, such as aspects that affect cycle time or employee skills [Kaplan 92]. Therefore, outcomes that relate to internal business processes and logistics for using RT are the best candidates for the goals in this Perspective. The proposed RT scorecard for the Internal Perspective is as follows:

Table 5-10. RTBSC Internal Perspective

Internal Business Perspective	
Goals	**Measures**
Ensure requirements are correctly implemented in design, code, tests	- RT Index Score
Increased ability to fulfill legislative or external constraints	**- Customer acquisition and retention (due to meeting DoD/ gov't/ private standards)** **- CMMi level attained**
Enhanced project tracking	**- Project schedule variance accuracy** **- Project cost variance accuracy**
More accurate and faster impact and trade-off analysis	- Average time to analyze impact of change - Defect rate due to impact analysis efforts
More accurate and faster understanding of requirement changes	- Average time to propagate change throughout artifacts - Defect rate due to changes not adequately propagated

Bold = Final Outcome

As can be seen, these goals focus on aspects that impact the organization's internal business processes. For the goal *Ensure Requirements are Correctly Implemented in Design, Code, Tests*, we developed the *RT Index Score* measurement. This score represents the percent of requirements that can be traced for a given context. If an organization is using heterogeneous traceability, all requirements will not be traced at any given time since some techniques are dynamic and used on an as-needed basis. However, if there are project level standards that need to be followed for a requirement or artifact, this Index can measure that. For example, if there is

a project-level strategy that dictates all requirements must have a source documented in some fashion (perhaps as a attribute within an RT tool), an audit can verify if all requirements have the attribute assigned. If the score is 75%, then the project manager and upper management know that 25% of the requirements for the project cannot be traced properly if the requirement's source is to be used as a search parameter for an IR query or even a simple manual search.

The goal *Increased Ability to Fulfill Legislative or External Constraints* can be measured by the number of customer contracts won, or customers retained due to compliance with a government or private industry standard. If the organization is in the process of a CMMi process improvement initiative, this goals can be measured against the CMMi level attained in relation to using RT within the organization. This goal can be fulfilled if Level 2 Requirements Management or Level 3 Requirements Development is attained for the organization [SEI 02].

The measures for other final outcome, *Enhanced Project Tracking*, can center on variances of time and cost on a project. If the level of project tracking is actually enhanced, then it can be assumed that project managers will be able to have more control over their projects due to having more accurate data by using traces as tracking mechanisms. Therefore, cost and time measurements might be more accurate and therefore have less variation as the project progresses.

The last two goals pertain to the internal processes of change management and impact analysis. Both can be measured by the similar metrics of time to perform change or impact analysis and the defect rate associated with improper change or impact analysis efforts.

5.6.1.3 The Innovation and Learning Perspective

This perspective is used to measure the organizations ability to innovate, improve, and learn since such activities tie directly to the company's value. It pertains to launching new products, improving efficiency, or even increasing revenue [Kaplan 92]. The traceability scorecard for this Perspective is described below:

Table 5-11. RTBSC Innovation and Learning Perspective

Innovation and Learning Perspective	
Goals	**Measures**
Increased system understanding	- System engineering efficiency (design, code, test)
Increased understanding of source and rationale of requirements	- System engineering efficiency (design, code, test)

In an effort for continual improvement and learning, it is important that the project team understands as much as possible about the system being developed. RT is the mechanism that can facilitate this learning. The two goals in this Perspective center on the knowledge gained and maintained by both Pre-RS and Post-RS techniques. "Increased understanding of source and rationale of requirements" describes how the project team can always be aware about the stakeholders involved in generating requirements and system artifacts in general as well as the rationale behind why the requirement exists in the first place. The other goal, "Increased system understanding", can pertain to both Pre-RS and Post-RS traceability. Since both goals describe

knowledge and learning about the system being developed, the can both be measured by the efficiency of the project team. They type of measurement would relate to the role of the project team member who's efficiency is being measured. Therefore, the measures for these goals can be refined if deemed appropriate by the organization. If the team has an increased level of system understanding, it might be assumed that they can be more efficient in their jobs, whether it be more lines of code (LOC) written per hour by developers, or manual requirements linked to system artifacts within an RT tool by business analysts.

5.6.1.4 Financial Perspective

The Financial Perspective concentrates on profitability, cost savings, growth, and shareholder value [Kaplan 92]. While an RT scorecard can rollup into an organizational-level scorecard that measures shareholder value, cash flow, and other such finances, there is still one goal from the Results Chain that can be directly related to the financial aspect of traceability.

Table 5-12. RTBSC Financial Perspective

Financial Perspective	
Goals	**Measures**
Decreased system maintenance costs	**- Average time and cost to retrieve artifact information and associations.**

Bold = Final Outcome

This goal refers to using traceability links as a part of the system documentation for maintenance efforts. This can be measured by analyzing the time (or translating time into cost) to retrieve artifact information and associations during system maintenance efforts. This measurement should see a decrease in time/cost as RT is implemented throughout an organization. It can be assumed that if RT links are persisted, they can be used as a tool to quickly understand a system and access related artifacts to a part of the system being maintained.

5.6.2 Strategy Map

An ancillary aspect to the BSC as Kaplan and Norton envisioned was to use it as an implementation strategy map [Kaplan 93]. In this way, the goals can map through a logical progression of implementation in order to successfully achieve them. In this paper, the *Benefits Realization Analysis Results Chain* (Section 5.2.1) accomplishes this objective, for it is a map of the goals or outcomes that need to occur in order to "Implement Requirements Traceability Techniques" initiative. Such a map can serve two purposes. First, it can act as a guide for implementing a final outcome. In the *Benefits Realization Analysis* process (Section 5.2), this is referred to as an "implementation map". The other purpose of such a map is to plot the progress against the scorecard goals/Results Chain benefits. For example, it makes no sense to expect a successful outcome of "More accurate and faster impact and trade-off analysis" using RT, if the benefit *Ensure Requirements are Correctly Iimplemented in Design, Code, Tests*" has not been already realized to some extent.

In this way, we implemented the Results Chain as a graphical scorecard. Each initiative can be shaded to illustrate the extent to which it has been implemented. The outcomes can be shaded to show the extent to which they have been achieved [Thorp 03]. Furthermore, colors can be added (green, yellow, red) to show the rate of progression to meet the outcome or initiative within a metric as set by the organization. For example, this metric could be time (on-time, potentially late, late) or cost (on budget, slightly over budget, very over budget).

In order to illustrate this concept, we used the same implementation map example in the *Benefits Realization Analysis* process (Section 5.2). The graphical scorecard for the final outcome *Decreased System Maintenance Costs* might look like this (for simplicity sake, just the relevant benefits are shown):

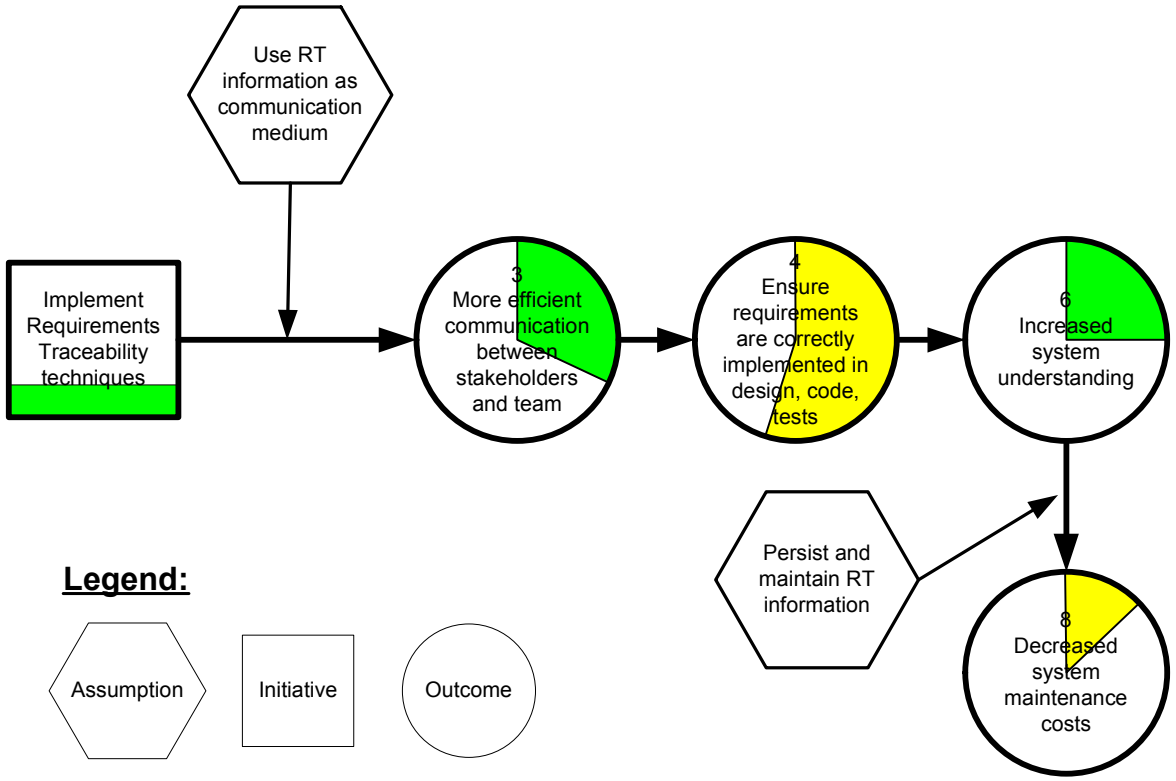

Figure 5-6. RT Balanced Scorecard Implementation Map

As can be seen by this example, the initiative is less that ¼ complete, but still on track for successful implementation. The *Ensure Requirements are Correctly Implemented in Design, Code, Tests* is over ½ complete, but is behind schedule or over budget. Even though the other two outcomes are green, the final outcome of *Decreased System Maintenance Costs* seems to be in jeopardy of missing its cost or timeline goals. Normally, such a representation will have objective data in support of the % completes as well as the ranges for the colors, and an explanation of their current status.

Summary

In this section, we presented the Value-Based Requirements Traceability (VBRT) framework as an integrated process model. This model can be used to manage the implementation or use of Requirements Traceability as a software engineering *and* a business process that must offer value to the organization in order to succeed. Otherwise, the organization may be at risk of losing money by thinking of RT in a value-neutral setting, when in fact, software engineering process are value creation activities. While the model has a definite flow, it has been built with enough flexibility that it can be modified based on an organizations unique context.

We first showed that an organization must first perform a *Benefits Realization Analysis* in order to understand the goals of RT for the company, and how they relate to each other. We outlined a set of base benefits that are realized by implementing RT in the form of the Benefits Realization Analysis Results Chain. However, these benefits can be augmented by additional goals that a company feels it can realize. Afterwards, the *Stakeholder Value Proposition Elicitation and Reconciliation* process needs to be executed in order to outline the needs and conflicts of the five main stakeholders involved in the RT process. This is accomplished by performing a conflict analysis and modeling the results in a Spiderweb Diagram. One this is done, a *Business Case Analysis* can be constructed to show the quantitative and qualitative viability of implementing RT. Through the Business Case, we have shown that Heterogeneous Traceability is the best technique that allows an organization to gain the most value from RT. After these three processes are complete, RT can be used on the *Software and System Engineering* process. As outlined in Section 5.4, we feel that this framework can be effective in concurrent and non-concurrent development environments, so no distinction has been made in VBRT. Next, the *Change as Opportunity* process allows the management of changes to requirements during *Software and System Engineering* using various models as described in Section 5.6. Also, the *Continuous Risk and Opportunity Management* process feeds into the *Software and System Engineering* process using Failure-To-Trace Risk scores to manage risk by deciding to trace or not to trace individual or certain sets of requirements. Finally, the *Value-Based Monitoring and Control* process acts as the feedback mechanism for the entire framework. By using the Requirements Traceability Balanced Scorecard, this process can report back progress of implementing RT to each process. This reporting can be done via a graphical Strategy Map for ease of use an understanding.

While the VBRT framework has not been field tested at this time, we have developed three case studies in the next section that will describe how the VBRT process works in real-world situations.

CHAPTER SIX: Case Studies

US Conglomerate (USC) is a trucking company that owns 7 other transportation companies across the U.S. As the company moved from ad-hoc IT project processes and management toward portfolio management of projects, several managers were tasked with finding ways for the IT organization to derive more value for the money spent from its operating budget. One method to accomplish this was through analyzing and potentially redefining some of their core software engineering practices.

Due to the new direction of portfolio management of projects, the IT Managers knew they could not follow the old procedures of putting together quick and simple spreadsheets with price points for various processes or tools they wanted to implement or purchase. This already had led to their current situation of having a toolbox of development tools, processes, and methodologies, most of which were not even used. Furthermore, the drive for portfolio management of projects meant they had to develop an adequate business case and supporting data for their ideas. If there was no benefit or value shown for an idea, it would most likely be rejected. As a result, they decided to take a value-based approach to understanding and determining their strategy for organizational efficiency and improvement. This meant that they had to prove their existing and any proposed processes worked by showing that they provided value to the project team, as well as the organization as a whole.

They reviewed all the processes that might allow them to make the biggest improvement inroads the fastest. Most of the processes only required minor enhancements. As an example, their teams used source code control, but needed minor enhancements to tighten the security processes on who could check-in and check-out modules and under what circumstances. They determined that source code control was inherent in the development process, so improvements did not need any defending or validation for upper management. It was the same situation with system design processes, bug tracking, and the software build process. This was not the case with the practice of requirements traceability. Many of the managers did not understand the value of this practice and thought it was a waste of time. However, a few managers thought that there might be some merit to utilizing requirements traceability on projects, for they had heard some success stories in the past of how it was immensely useful to a project. After further discussion, the managers decided to investigate the idea of using requirements traceability throughout USC, and tasked two of the managers to lead the effort.

Case Study #1

One of the IT managers was an Application Development Manager (ADM) who had led some troubled projects at the company. Most of his projects were late and over budget, and his system designers and developers were often blamed. In order to find the root cause of the problems of the past projects, he read several of the Post Mortem and After-Action Review documents from the projects in his department. He noticed a glaring trend that his team sometimes missed a final feature or two that was required by the customer. Other times the feature was implemented, but not to the specification outlined in the requirements. He also saw that requirement errors related to traceability seemed to be the cause of excessive rework and missed features in the final product of several projects. Since there was no traceability between requirements, design, and

code, features were often missed or their implementation misinterpreted. This was a cause for concern since even though his department had requirement management tools with trace capability, they didn't often use them or abandoned them mid-project since his team never really understood their value.

Another problem that the manager identified in the After-Action Reviews was that many teams were distributed across the seven companies and had many communication-related and coordination conflicts regarding application functionality. This led to the problem where changes in the requirement documents often did not get updated in code or design. Also, design changes many times did not make it back into requirements documents either, especially if they did not directly affect the customer. This lack of traceability also led to the customer's inability to maintain the systems that were built since the requirements did not accurately reflect was in the design, code, and tests. To show that requirements traceability might be a tool that can help solve these issues, he first identified the stakeholders that would interact with any proposed RT practices, understood their conflicts, and identified any opportunities that could be expanded for enhanced collaboration within the RT arena. As a result of this effort, he would know where teams would gain the most benefit from RT and USC could proactively mitigate and potential conflict from using traceability. This would then increase the chances of successfully implementing and using RT on projects within the company.

He started by identifying the stakeholders in most projects as the following 3 high-level groups:

- Customer Users: Those who will use the system and are the source for requirements
- Customer Maintainers: Those who maintain the system on the Customer's end
- Project Team: The team at USC corporate that developed the system

Next, he outlined their potential needs from RT. This was an aggregation of what they could get out of traceability on a project.

Table 6.1. Stakeholder needs from RT

Group	Need
Customer Users	• Maximize the use of a limited budget • Get a working system delivered to them as soon as possible or earlier • All requested features must be implemented in the final system • Ability to change requirements as need be due to a changing business environment
Customer Maintainers	• A system they could easily maintain after it was deployed "live" • An easy transition of the system artifacts over to them
Project Team	• Due to utilizing a waterfall development process, they preferred stable requirements after the Requirements Phase • Maximize the output of team members for the lowest cost and in the shortest timeframe • Ensure requirements are implemented the design, code, and test cases • Requirements verification during testing phase

Based on the identified stakeholders and their needs, he developed the USC RT Spiderweb Diagram.

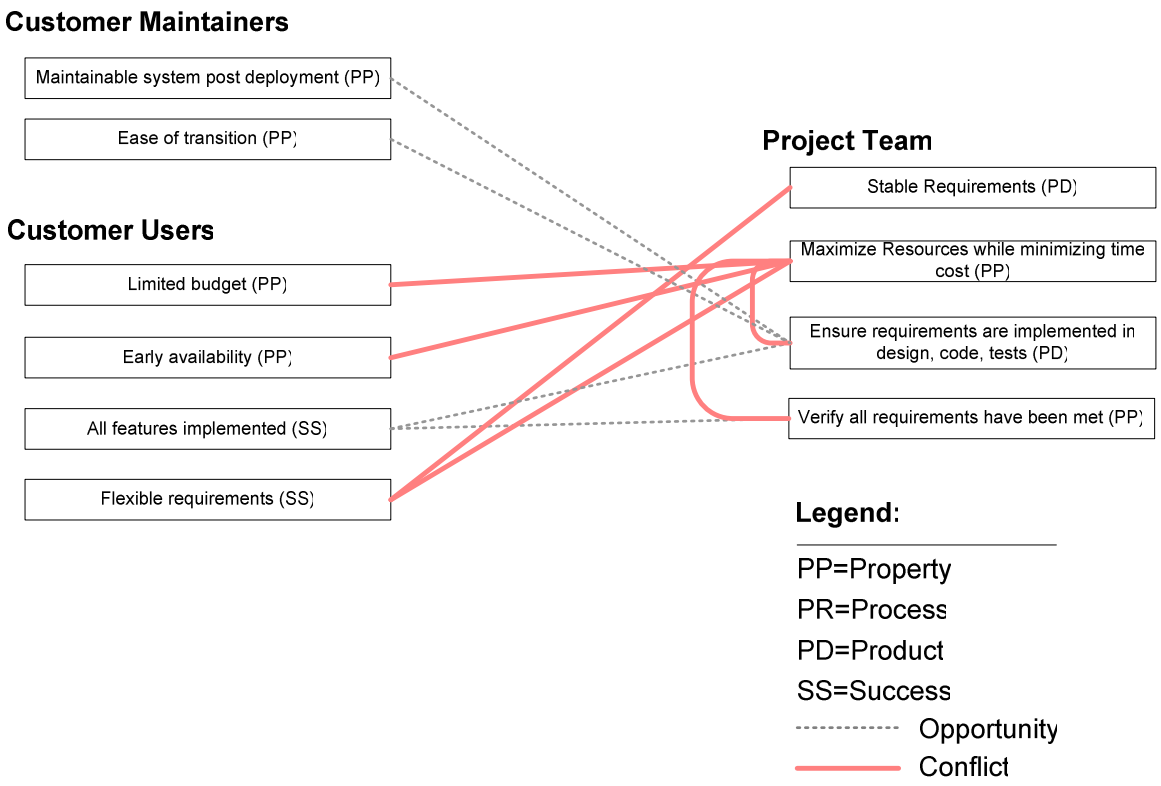

Figure 6-1. USC Spiderweb Diagram

After reviewing his model, the ADM identified several model-clash relationships pertaining to RT (the red lines):

- Product Team: The Project Team had a few conflicts within itself. The Property model of managing the project constraints of time, cost, and schedule were in conflict with the Property model of requirements verification (e.g. Pre-RS) and the Product model of ensuring that requirements have been implemented (e.g. Post-RS) throughout all artifacts in the system. As a result of this, he felt an adequate business case and benefit analysis would have to be developed in order to show these conflicts could be resolved by implementing RT. If these needs could be met easier and faster than before, the organization could save time and money compared to the way the teams operated today.
- Customer Users: The Customer Users had several conflicts as well. One pertained to their Success model of flexible requirements, which conflicted with the Project Team's Process model of stable requirements. Stable requirements are a Process model since it reflects the concept that the project team operates in the waterfall development environment. Since the team uses a waterfall model, they desire stable requirements while the customer wants to be able to change the system as need be. Being able to reduce this conflict would also support the implementation of RT since the organization could better manage the changes derived from flexible requirements. Flexible requirements, limited budget, and early availability also conflict with the Property model of managing project constraints. Changes to requirements often added time to the project and cost to implement as well. Limited budget limited the project teams' options for

delivering the system, and early availability for the system often led to increased cost to deliver on a shorter timeframe. By understanding that these model-clashes existed in the first place, the managers were able to focus on showing how RT can help mitigate the conflicts.

- Customer Maintainers: While there were no conflicts represented in the model, their goals of a maintainable "System Post-Deployment" and having an "Ease of transition" must be kept in mind during the course of a project.

After understanding the model-clashes inherent within their organizations, the Development Manager submitted this information to the other manager investigating RT for the organization.

Case Study #2

The QA Manager for the organization also had problems from her aspect of projects that she felt RT might be able to help solve. For most projects, traceability was often abandoned (or never used) as the project progressed due to tight schedules, missed deadlines, and shifting resources. She knew based on previous experience that by the time projects got to the testing phase, her testers wanted to trace tests back to code and requirements, but couldn't since the RT efforts were incomplete or abandoned. This in turn increased the risk of system failures since her testers were unable to adequately match test cases to code, design, and the requirements that defined the system.

In order to help drive the initiative for proving the value of requirements traceability, she developed a business case for upper management. She took time to observe the requirements-related processes at her company, noted the failure points, and then outlined the potential benefits to her firm. She noted that if RT was implemented, there would potentially be a reduction in the testing cycle since test cases could be derived and verified from requirements and code via traceability links, which would meet the needs of the Project Team in the Spiderweb Diagram. A reduction in the testing cycle could save money for the project. This would also help project teams in times of change and reduce the conflicts between the Project Team and the Customer User as supported by the Spiderweb Diagram. When requirements were modified, added, or deleted, the artifacts affected by the change, including test cases, could be understood more thoroughly if traceability was used. Furthermore, from her point of view, she could monitor the progress of her testing team based on the linkages from test cases to use cases. When test cases were successfully executed, the links could have an attribute signifying completion in a requirements management tool. This concept could be replicated to the other ADM's as well. Developer's and system designer's progress could be better monitored based on the links between requirements and their implementation in design and code. Finally, she noted that many requirements-related errors and project delays might be avoided if there was traceability information captured about the source and rationale of why a requirement exists. If that could be done, then in situations of requirements change, validation, and verification, the project team could better understand the system being built and know who to contact if questions arose.

The QA Manager took all of these observations along with the RT Spiderweb Diagram and modeled them into a modified USC-version of the Requirements Traceability Results Chain. By

doing this, she was able to map out the benefits that RT could bring to USC and understand their relation to each other.

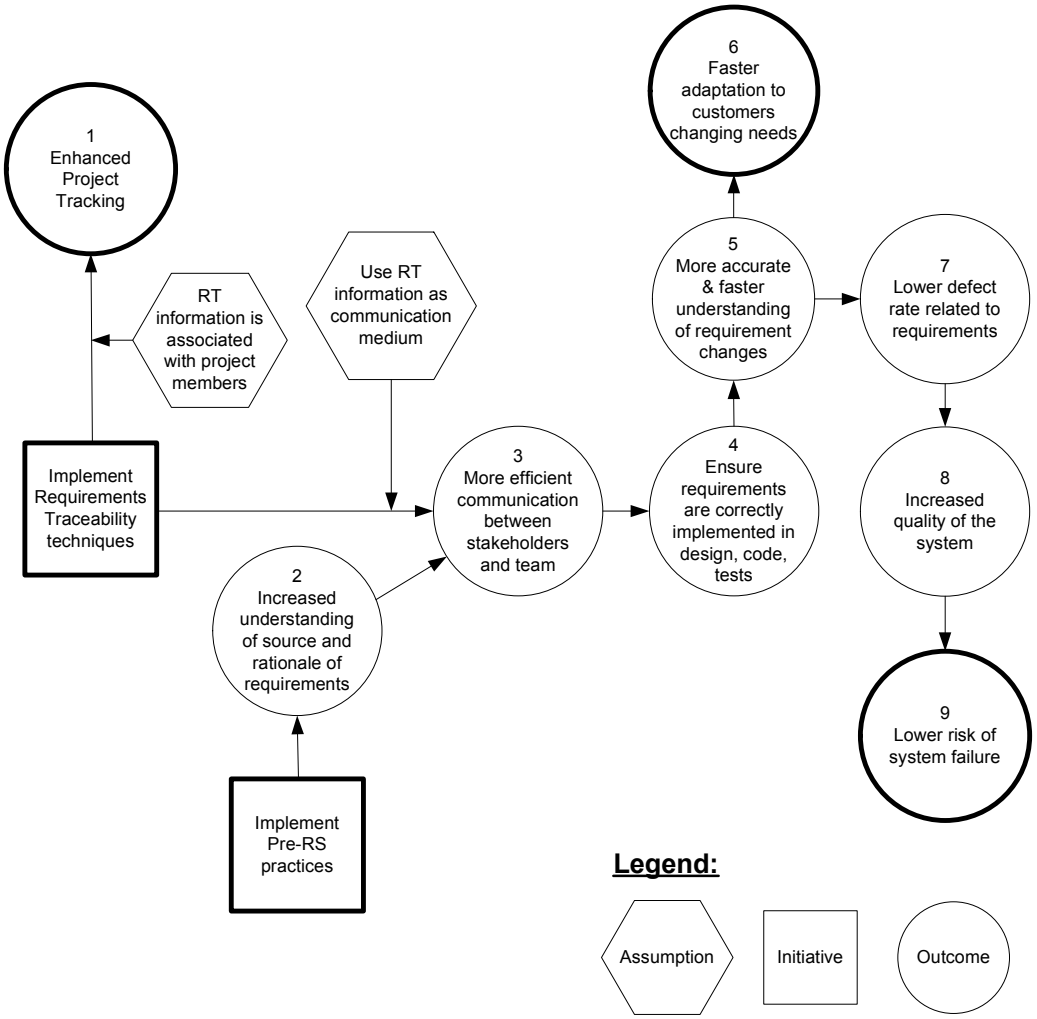

Figure 6-2. USC Benefits Realization Analysis Results Chain

After developing the Results Chain, she noted all of the final outcomes were beneficial to the firm, but also realized that the intermediate outcome #3 was of special importance to USC since project teams were distributed throughout the country. This would have to be called out in the business case.

Once she understood the benefits that her firm could realize from RT, she had to develop the financial support for her claims. While the Results Chain mapped out the benefits, they needed to be adequately quantified. She knew that using no traceability techniques at all led to many project problems and requirement-related errors. She also knew that using completely manual traceability techniques were possible, but not feasible. USC did not have the manpower to have full-time staff manage requirements in that manner. Also as mentioned earlier, previous projects at the company showed that most of the time a completely manual approach to traceability failed due to the effort eventually being abandoned due to shifting priorities or resources. Therefore,

she proposed using Heterogeneous Traceability in order to maximize the value of RT. All of these concepts were validated and incorporated into a business case for upper management. The QA Manager outlined the costs of using no RT approach (as most projects currently were), a manual approach to traceability, as well as a blend of manual and dynamic techniques. She felt that dynamic traceability would make RT much easier to use, and could be used by many different stakeholders on the team such as business analysts, developers, system architects, and designers. Since the effort to maintain dynamic traceability links would be less than a manual effort, she felt the rate of adoption would be very acceptable.

Based on historical data from projects that utilized some form of RT, she developed a baseline of an average project at the company for developing a financial case for RT. These generalities could then be used to generate the financial outline for traceability.

Table 6-2. USC General Costs of RT

	No Traceability	Manual Traceability	Heterogeneous Traceability
No. of requirements	1500	1500	1500
No. of changed requirements	500	500	500
Average number of links per requirement	5	5	5
Total links (average)	7500	7500	7500
Explicit links	0	6750	1875
Dynamic links	0	0	3375
No links	6750	0	1725
Non-traceable links	750	750	525

Once she outlined the general costs of traceability, she then applied those numbers to the cost of the actual effort of RT. She based these numbers on salary data provided by the USC Finance department and historical project effort hours for performing RT-type activities (similar to Chapter 5, Section 5.3.2). All dollar amounts were corrected over 1 year, which was the average project lifecycle for USC. The average maintenance duration for systems within the companies was 3 years. The IRR for the firm was set to 6% by the Finance department. Finally, the ADM and QA Manager determined that $200,000 of benefits per project could be realized based on the Results Chain.

Table 6-3. USC Effort Costs of RT

	No Traceability	Manual Traceability	Heterogeneous Traceability
Setup cost per link	$0	$13	$13
Total Setup Cost	$0	$82,783	$22,995
Maintenance cost per link	$0	$4	$4
Total Maintenance Cost	$0	$25,471	$7,075
Cost of change per link	$26	$6	$6
Total Change Cost	$61,320	$14,150	$14,150
Total Cost of Traceability	**$61,320**	**$122,404**	**$44,220**
Risk of System Failure	High	Medium	Low

The financials of the business case supported her claim that Heterogeneous Traceability was the best way to approach RT for the company. Furthermore, the level of the risk of system failure for Heterogeneous Traceability could be managed by implementing Failure-To-Trace Risk concepts within the organization's project-level risk management practices. After describing the benefits and supporting financial information, the QA Manager felt she built an adequate case for implementing RT for USC. However, it required culture changes to use RT full-lifecycle, and needed the support of upper management for financial and implementation support. Therefore, she and the Development Manager presented their ideas to upper management to enlist their assistance and approval.

Case Study #3

Once submitted, upper management reviewed the Spiderweb Diagram and analysis, the Benefits Realization Analysis Results Chain, as well as the business case. They concluded that the financials from the business case were sound. Even though they believed that the first few projects would be considered loss leaders since it would take some time for the project teams to ramp up on the new processes and concepts, they felt that RT should be implemented at their firm as the long term benefits would shadow the short term losses. While upper management agreed with most of the outcomes already in the Results Chain, they felt that they needed to modify the model, as well as add two more outcomes: "Shorter Development Cycles" and "Project Cost Reduction", which led to a new final outcome of "Increased IT Revenue". They felt that the time saved by their project teams using dynamic techniques, and the fact that requirement-related questions could be clarified easier if system rationale and sources for requirements were captured, would potentially offset the use of RT and allow systems to be developed faster for the seven transportation companies they owned. This in turn would potentially lead to more work that the seven companies would request of the IT department, which would increase revenue since the IT department charged back all work to the company requesting the project. The resulting USC RT Results Chain looked like this:

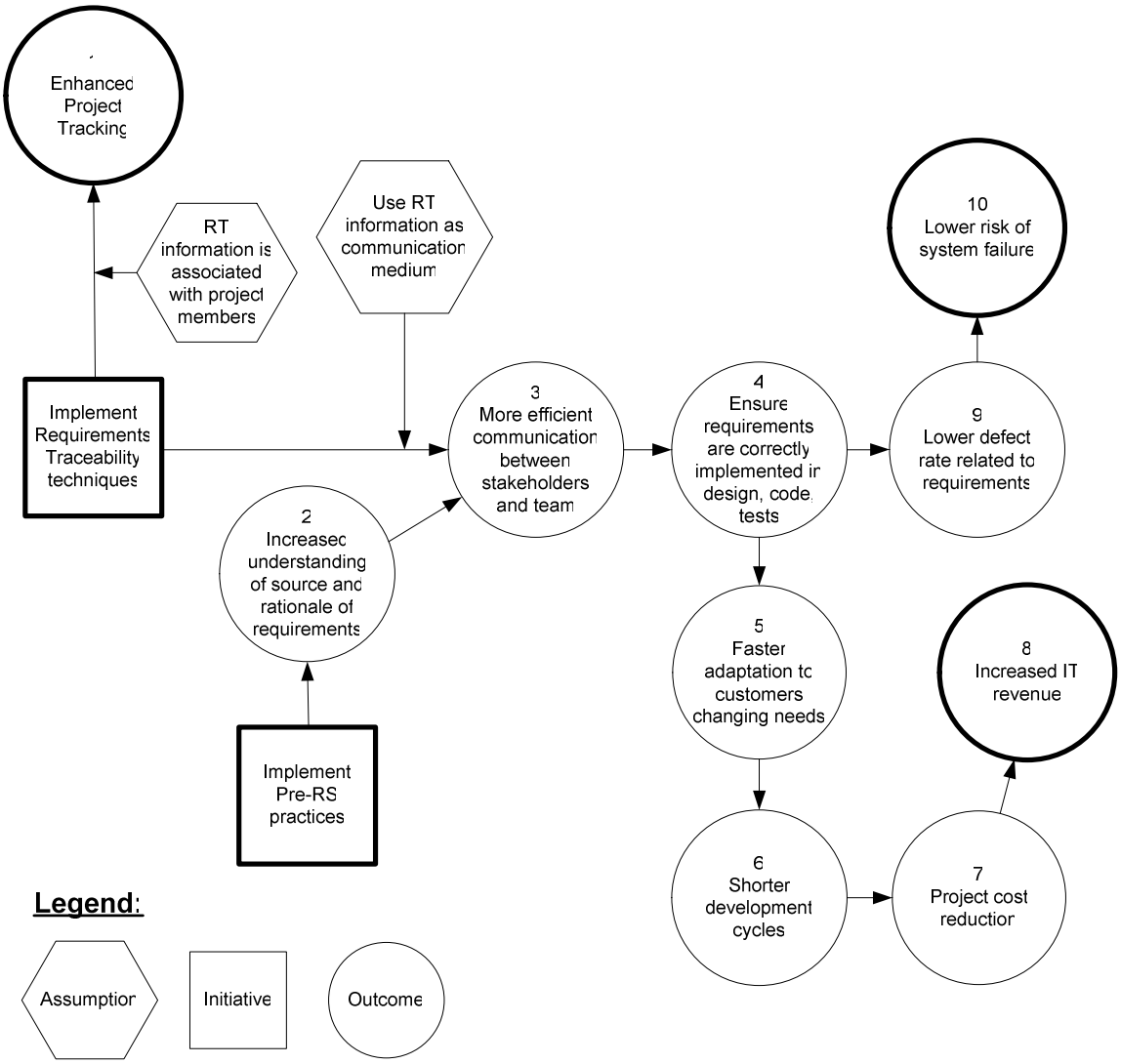

Figure 6-3. USC Modified Results Chain

Once upper management was comfortable with the benefits, the qualitative analysis, and the conflicts and opportunities presented by all the potential users, they decided to build an implementation roadmap for RT. This could help them understand the impact to USC as a whole. They were able to do this using the Requirements Traceability Balanced Scorecard as a baseline and modifying it for their organization. They took their benefits and mapped them to the four perspectives of the scorecard. For the additional outcomes they developed, they felt that a good measure of "Shorter Development Cycles" and "Project Cost Reduction" was to measure the budget and duration of the projects using dynamic RT techniques. They could then be compared to previous projects that did not use RT. The time that was used for managing the linkages and performing the RT effort needed to be monitored separately to make sure that the benefits were truly being realized, and the new process was not getting in the way of project success. The "Increased IT Revenue" could be easily measured by looking at the historical revenue from projects already completed for the seven subsidiary companies, and then looking at the revenue from the companies after RT was implemented enterprise-wide.

The resultant USC RT Balanced Scorecard they developed is as follows:

Table 6-4. USC RT Balanced Scorecard

Customer Perspective	
Goals	**Measures**
More efficient communication between stakeholders and team	- Management and consultation time spent per project - Average time to verify requirement and rationale with source
Faster adaptation to customers changing needs	- % accepted changes successfully implemented (defined by customer) - Schedule impact due to changes
Lower defect rate related to requirements	- Requirement-related defect rate
Shorter development cycles	- Duration of project (in relation to its complexity)
Lower risk of system failure	**- RT Risk Index score (Section 5.4)**

Internal Business Perspective	
Goals	**Measures**
Ensure requirements are correctly implemented in design, code, tests	- RT Index Score (Section 5.4)
Enhanced project tracking	**- Project schedule status/forecasting accuracy** **- Project cost status/forecasting accuracy**

Innovation and Learning Perspective	
Goals	**Measures**
Increased understanding of source and rationale of requirements	- System engineering efficiency (design, code, test)

Financial Perspective	
Goals	**Measures**
Project Cost Reduction	- Project budget/ manpower required for project (in relation to its complexity)
Increased IT revenue	**- Gross quarterly revenue based on projects from seven external companies**

As a final measure to ensure that the goals were being realized in a timely manner, upper management developed a strategy map to monitor the progress of rolling out RT within the organization. This graphical model allowed them to understand, at a glance, how the implementation of the benefits were progressing. The two factors that they cared about the most was the % completion of realizing the benefits, and the timeliness of their realization. The dial

represented the progress to date, and the color represented the status of the schedule for implementing the outcome. Green meant the initiative was on-schedule. Yellow meant that the effort was behind schedule, but there was a plan to get the initiative back on track. Red signified that the effort was behind schedule and there was no current plan to get back on track at that time.

After almost a year, the USC RT Strategy Map looked like this:

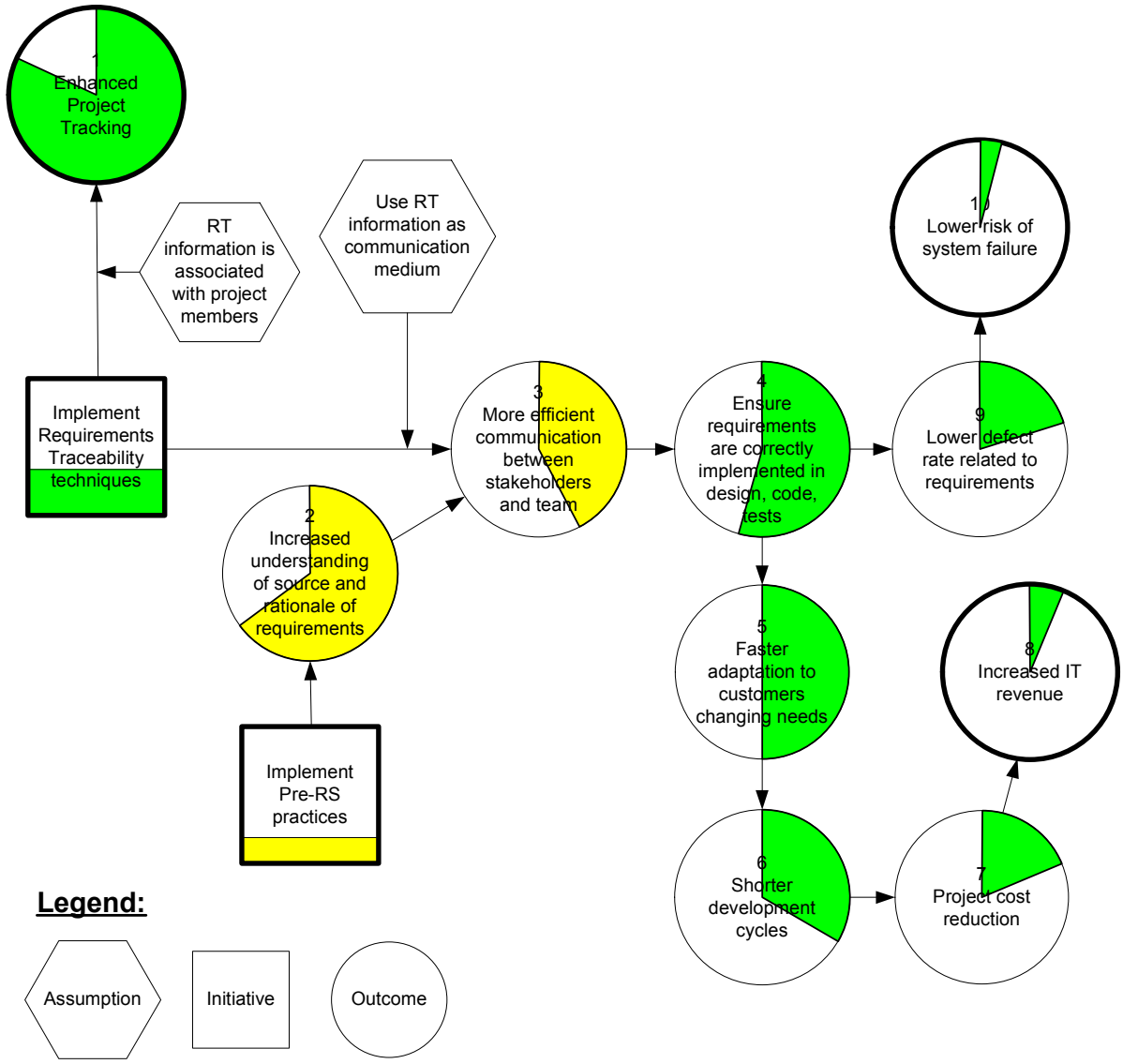

Figure 6-4. USC Strategy Map

Summary

These case studies outlined an approach where by understanding *Stakeholder Value Proposition Elicitation and Reconciliation*, performing a *Benefits Realization Analysis* and *Business Case Analysis*, the value that RT could bring to USC was understood. The aspect of *Change as Opportunity* was introduced since the organization no longer had to view change as a problematic idea, but could now manage it better on their projects by implementing traceability. *Continuous Risk and Opportunity Management* was also realized by the fact that the QA Manager understood that the risk of system failure by inadequate test cases could be mitigated by implementing RT to ensure that test cases had adequate coverage of the system. Finally, these five aspects of VBSE then fed into a *Value-Based Monitoring and Control* mechanism in order to create a feedback loop that showed the progress and success of implementing the benefits of RT.

CHAPTER SEVEN: Future Work

The Value-Based Requirements Traceability (VBRT) framework we presented in this paper has many opportunities for further expansion and research. While most of our ideas have strictly followed Value-Based Software Engineering by Boehm, we felt that the *Concurrent Software and Systems Engineering* aspect of VBSE did not adequately pertain to VBRT. While our framework can be process agnostic, we set an underlying assumption that the use of VBRT is geared toward traditional software and systems engineering environments. It would be interesting to investigate if VBRT could or should be modified in order to work within an agile environment. In order for this to happen, especially in methodologies that promote test-driven development, the framework might have to only consider *backward toward requirements* traceability since tests launch the development process. This would have to be investigated.

Another area of future research would be to verify and validate VBRT within industry. This can be an extensive course of action since the processes that can be associated with VBRT: process improvement, cultural differences, tool buy-vs.-build decisions, and the associated acceptance by upper management (since they most likely would be the end consumers of the output of the framework), can take quite some time. It is suggested that in order to be the most successful, not only would the sample size need to be the largest possible, but the framework should be introduced within myriad situations. For example, it would be interesting to see the outcome with companies who have no RT processes in place, and others that do. It would also be interesting to see the effects the framework can have on large as well as small organizations.

CHAPTER EIGHT: Conclusion

In this paper, we have presented the comprehensive Value-Based Requirements Traceability (VBRT) framework. The VBRT framework is composed of seven processes that can walk a company through the process of understanding, realizing, and measuring the value of RT. The seven aspects of the framework are outlined below:

Stakeholder Value Proposition Elicitation and Reconciliation
We showed that by performing a comprehensive analysis using the stakeholders in the RT process that we identified, an organization can map out users that take part in the RT process and identify their relationships in order to proactively mitigate conflicts and take advantages of opportunities for collaboration.

Benefits Realization Analysis
By using an effective tool such as the Business Realization Analysis Results Chain, we modeled the main benefits for RT that should be able to be used within most organizations. This model can also be used as an implementation map for rolling out RT. While the benefits can be self-contained, the model is flexible enough to incorporate custom benefits as they might pertain to unique situations.

Business Case Analysis
Most research has focused on using a single technique for using RT. We have shown that the most cost effective manner in which to implement RT is to use Heterogeneous Traceability, which is a proper blend of manual and dynamic techniques based on a unique context. Heterogeneous Traceability is the technique that aligns the closest to the benefits outlined in the BRA Results Chain as well.

Concurrent System and Software Engineering
We presented the VBRT framework as a model that is process agnostic. However, we concluded that there would have to be additional thought and research into how it can be adopted into agile methodologies since by their very nature (and the Agile Manifesto), they reject the concept of a process intensive requirements framework within the development lifecycle.

Continuous Risk and Opportunity Management
The concept of the Failure-to-Trace Risk (FTR) as proposed by Cleland-Huang, Zemont, et. al. introduces risk management and monitoring into the framework and provides a way to quantitatively measure risk situations within sets of requirement links.

Change as Opportunity
Using traces as a mechanism to propagate and understand change throughout a system is an inherent property of RT. Therefore, we did not develop another change management model since this process has already been covered by a vast amount of research and we felt we could offer no more value to this topic.

Value-Based Monitoring and Control

We presented the Requirements Traceability Balanced Scorecard (RTBSC) as an effective way to monitor the implementation and value gained from VBRT. This process was also shown as a feedback mechanism into the framework if changes need to be made as a result of monitoring and controlling the processes in the framework. Finally, the RTBSC was presented as a strategy map/dashboard as well in order to visually understand the progress of RT value realization.

In conclusion, the VBRT framework was developed in response to the current state of research in RT focusing on the technological aspects for the process and not the benefits it can provide and how organizations can realize those benefits. Showing the value of RT is very important since the main reason for project failure, as explained by the Standish Group, is due to requirements-related problems during the software development lifecycle. We feel that taking RT out of a value-neutral setting and understanding how the process can benefit an organization may increase its adoption rate as a software engineering process which can benefit the IT industry as a whole.

References

[Adaptive] Adaptive Software Development: http://www.adaptivesd.com.

[Alexander 04] I. Alexander, S. Robertson, "Understanding Project Sociology by Modelling Stakeholders", *IEEE Software*, Vol. 21, No. 1, pp. 23-27, January/February 2004.

[Anderson 02] S. Anderson, M. Felici, "Quantitative Aspects of Requirements Evolution", *26th Annual International Computer Software and Applications Conference*, August 2002, p. 27.

[Antoniol 00] G. Anoniol, G. Canfora, G. Casazza, A. De Lucia, "Information Retrieval Models for Recovering Traceability Links between Code and Documentation", *Proceedings of the Internaltional Conference on Software Maintenance,*2000, pp. 40-49.

[Antoniol 02] G. Anoniol, G. Canfora, G. Casazza, A. De Lucia, E. Merlo "Recovering Traceability Links between Code and Documentation", *IEEE Transactiona on Software Engineering*, Vol. 28, No. 10, pp. 970-983, October, 2002

[APM 04a] Agileprojectmanagement Yahoo! Group Discussion, Digests 252-264, March 10 - March 20, 2004.

[APM 04b] Agileprojectmanagement Yahoo! Group Discussion, Digests 255, March 13, 2004.

[Appleton 04] B. Appleton (brad@bradapp.net), "Why Traceability? Can it be Agile?", Post to agileprojectmanagement Yahoo! Group, March 10, 2004.

[Beedle 04a] M. Beedle (beedlem@e-architects.com), "Re: Refactoring Requirements over Tracing them", Post to agileprojectmanagement Yahoo! Group, March 20, 2004.

[Beedle 04b] M. Beedle (beedlem@e-architects.com), "Proposed Amendment to the Agile Manifesto", Post to agileprojectmanagement Yahoo! Group, March 13, 2004.

[Bianchi 00] Bianchi, A., Fasolino, A., and Visaggio, G., “An Exploratory Case Study of the Maintenance Effectiveness of Traceability Models”, *Proceedings of the Eighth International Workshop on Program Comprehension*, 2000, pp 149-159.

[Boehm 00a] B. Boehm, D. Port, M. Al-Said, "Avoiding the Software Model-Clash Spiderweb", *Computer,* Vol. 33, No. 11, pp. 120-122, November 2000.

[Boehm 00b] B. Boehm and K.J. Sullivan, "Software economics: a roadmap," in The Future of Software Engineering, *22nd International Conference on Software Engineering*, June, 2000, pp. 319--344

[Boehm 01a] B. Boehm, V. Basili, "Software Defect Reduction Top 10 List", *Computer,* Vol. 34, No. 1, pp. 135-137, January 2001.

[Boehm 01b] B. Boehm, D. Port, K. Sullivan, "Value-Based Software Engineering", presented at Vanderbilt Workshop New Visions for Software Design & Productivity: Research & Applications,

Vanderbilt University, Nashville, TN, December 13-14, 2001.

[Boehm 03a] B. Boehm, “Value-Based Software Engineering”, *ACM Software Engineering Notes*, Vol. 28, No. 2, pp. 1-12, March 2003.

[Boehm 03b] B. Boehm, L. Huang, "Value-Based Software Engineering: A Case Study", *Computer,* Vol.36, No. 3, pp. 3-41, March 2003.

[Boehm 91] B. Boehm, "Software Risk Management: Principles and Practices", *IEEE Software,* Vol. 8, No. 1, pp. 32-41, January 1991.

[Chung 95] L. Chung, B. Nixon, "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach", *Proceedings of the 17th international conference on Software Engineering,* 1995, pp. 25-37.

[Cleland-Huang 02a] J. Cleland-Huang, et. al., "Automating Speculative Queries through Event-Based Requirements Traceability", *Proceedings of the IEEE Joint Conference on Requirements Engineering*, September 2002, pp. 289-297.

[Cleland-Huang 02b] J. Cleland-Huang, C. Chang, Y. Ge, "Supporting Event Based Traceability Through High-Level Recognition of Change Events", *Proceedings of the 26th Annual International Computer Software and Applications Conference*, 2002.

[Cleland-Huang 03a] J.Cleland-Huang, C. Change, M. Christensen, "Event-Based Traceability for Managing Evolutionary Change", *IEEE Transactions on Software Engineering*, Vol. 29, No. 9, pp. 796-810, September, 2003.

[Cleland-Huang 03b] J. Cleland-Huang, D. Schmelzer, "Dynamically Tracing Non-Functional Requirements through Design Pattern Invariants", *Workshop on Traceability in Emerging Forms of Software Engineering, in conjunction with IEEE International Conference on Automated Software Engineering,* October, 2003.

[Cleland-Huang/Zemont 04] J. Cleland-Huang, G. Zemont, W. Lukasik, "A Heterogeneous Solution for Improving the Return on Investment of Requirements Traceability", *12th IEEE International Requirements Engineering Conference*, September 2004, pp. 230-239.

[CNA] G. Zemont, Claim Center project, CNA Insurance, 2004.

[Cockburn 04] A. Cockburn (acockburn@aol.com), "Re: [scrumdevelopment] Why Traceability? Can it be Agile?", Post to agileprojectmanagement Yahoo! Group, March 10, 2004.

[Collins 94] W. Collins et. al, "How Good is Good Enough: An Ethical Analysis of Software Construction and Use", *Communications of the ACM*, Vol. 37, No. 1, January 1994, pp. 81-91.

[Conrow 97] E. Conrow, P. Shishido, "Implementing Risk Management on Software Intensive Projects", *IEEE Software*, Vol. 14, No. 3, pp. 83-89, May 1997.

[Crystal] Crystal Website, http://alistair.cockburn.us/crystal.

[DoD] U.S. Department of Defense, “Military standard: Defense systems software development, DODSTD- 2167A”, February 1988.

[Domges 98] R. Domges and K. Pohl, “Adapting Traceability Environments to Project-Specific Needs”, *Communications of the ACM*, Vol. 41, No. 12, pp. 54-62, December 1998.

[Egyed 00] A. Egyed, "A Scenario-Driven Aproach to Traceability", *23rd International Conference on Software Engineering,* 2000, pp. 123-132.

[Egyed 02] A. Egyed, P. Grunbacher, “Automating Requirements Traceability: Beyond the Record and Replay Paradigm”, *Proceedings of the 13th IEEE International Conference on Automated Software Engineering*, 2002, pp. 163-171.

[Faulk 00] Faulk, S.; Harmon, R.; & Raffo, D. "Value-Based Software Engineering (VBSE): A Value-Driven Approach to Product-Line Engineering," 205-224. *Software Product Lines: Proceedings of the First Software Product Line Conference (SPLC1)*. Denver, Colorado, August 28-31, 2000. Boston, MA: Kluwer Academic Publishers, 2000.

[Finkelstein 91] A. Finkelstein, "Tracing Back from Requirements", *IEEE Colloquium, Computing and Control Division, Professional Group C1*, 1991, pp. 7/1-7/2.

[Giesen 02] J. Giesen, A. Volker, "Requirements Interdependencies and Stakeholders Preferences ", *IEEE Joint International Conference on Requirements Engineering*, September 2002, p. 206.

[Goldenson 03] D. Goldenson, D Gibson, "Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results", Software Engineering Institute, Pittsburgh, PA, CMU/SEI-2003-SR-009, October 2003.

[Gotel 94] O. Gotel and A. Finkelstein, “An Analysis of the Requirements Traceability Problem”, *Proceedings of the 1st International Conference on Requirements Engineering*, 1994, pp. 94-101.

[Gotel 95] O. Gotel and A. Finkelstein, “Contribution Structures”, *Proc. Second Int’l Conf. Requirements Eng.*, 1995, pp. 100-107.

[Gotel 97] O. Gotel and A. Finkelstein, “Extended Requirements Traceability: Results of an Industrial Case Study”, *3rd IEEE International Symposium on Requirements Engineering*, 1997, pp. 169-178.

[Grembergen 02] W. Grembergen, I. Amelinckx, "Measuring and Managing E-Business Projects through the Balanced Scorecard", *Proceedings of the 35th Hawaii International Conference on System Sciences*, January 2002, pp. 258-266

[Haumer 99] P. Haumer, et. al., "Bridging the Gap Between Past and Future in RE: A Scenario-Based Approach", *IEEE International Symposium on Requirements Engineering*, June 1999, p. 66.

[Hayes 03] J. Hayes, A. Dekhtyar, J. Osborne, "Improving Requirments Tracing via Information Retrieval", *Proceedings of the 11th IEEE International Requirements Engineering Conference,* 2003, pp.138-147.

[Heinz 04] L. Heinz, "CMMI for Small Businesses: Initial Results of the Pilot Study", http://www.sei.cmu.edu/news-at-sei/features/2004/3/feature-1-2004-3.htm, 2004.

[Higgins 03] S. Higgins, et. al, "Managing Requirements for Medical IT Products", *IEEE Software*, Vol. 20, No. 1, January/ February 2003, pp. 26-33.

[IEEE 04] IEEE Computer Society SWEBOK Team, *Guide to the Software Engineering Body of Knowledge (SWEBOK),* IEEE, 2004.

[INCOSE 03] The International Council on systems Engineering (INCOSE), "Tools Survey: Requirements Management Tools", http://www.paper-review.com/tools/rms/read.php, 2003.

[Jarke 93] M. Jarke and K. Pohl, "Establishing Visions in Context Towards a Model of Requirements Processes", *Proceedings of the 14th International Conference on Information Systems*, 1993, pp. 23-34.

[Jarke 98] M. Jarke, “Requirements Tracing”, *Communications of the ACM*, Vol. 41, No. 12, pp. 32-36, December 1998.

[Jodhi 03] J. Sodhi and P. Sodhi, *Managing IT System Requirements,* Vienna: Management Concepts, 2003.

[Kaplan 92] R. Kaplan, D. Norton, "The Balanced Scorecard - Measures That Drive Performance", *Harvard Business Review - HBR OnPoint,* January-February 1992.

[Kaplan 93] R. Kaplan, D. Norton, "Putting the Balanced Scorecard to Work", *Harvard Business Review - HBR OnPoint*, pp.1-16, September-October 1993.

[Knethen 02] A. Von Knethen, "Change-Oriented Requirments Traceability. Support for Evolution of Embedded Systems", *International Conference on Software Maintenance*, Montreal, Canada, October 3-6, 2002

[Lam 98] W. Lam, M. Loomes, "Requirements Evolution in the Midst of Environmental Change: A Managed Approach", *2nd Euromicro Conference on Software Maintenance and Reengineering,* March 1998, p.121-127.

[Lam 98] W. Lam, V. Shankararaman, "Managing Change During Software Development: An Incremental, Knowledge-Based Approach", *10th Annual Conference on Software Engineering and Knowledge Engineering*, 1998,

[Lam 99] W. Lam, V. Shankararaman, "Requirements Change: A Dissection of Management Issues", *25th Euromicro Conference-Volume 2*, 1999, pp. 2244-2251.

[Lamsweerde 01] A. Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour", *Proceedings of the 5th International Symposium*

on Requirements Engineering, August 2001, pp. 249-262.

[Lavazza 00] L. Lavazza, G. Valetto, "Enhancing Requirements and Change Management through Process Modeling and Measurement", *4th International Conference on Requirements Engineering*, 2000, pp. 106-115.

[Lee 03] C. Lee, L. Guadagno, X. Jia, "An Agile Approach to Capturing Requirements and Traceability", *Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering,* 2003, pp.17-23.

[Leffingwell 96] D. Leffingwell, "A Field Guide to Effective Requirements Management Under SEI's Capability Maturity Model", Rational Software Corporation, 1996.

[Leffingwell 97] D. Leffingwell, “Calculating Your Return on Investment from More Effective Requirements Management”, Rational Software Corporation, 1997.

[Leishman 02] T. Leishman, D. Cook, "Requirement Risks Can Drown Software Projects", *Crosstalk,* April 2002, http://www.stsc.hill.af.mil/crosstalk/2002/04/leishman.html

[Macaulay 96] L. Macaulay, "Requirements for Requirements Engineering Techniques", *Proceedings of the 2nd International Conference on Requirements Engineering*, April 1996, pp. 157-164.

[Mair 02] S. Mair, "A Balanced Scorecard for a Small Software Group", *IEEE Software*, Vol. 19, No. 6, pp. 21-27, November/ December 2002.

[Manifesto] Agile Manifesto: http://www.agilemanifesto.org/

[Microsoft] Microsoft: http://www.microsoft.com

[Murthi 02] S. Murthi, "Preventive Risk Management for Software Projects", *IT Professional,* Vol. 4, No. 5, pp. 9-15, September 2002.

[Mylopolous 92] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and Using Non-Functional Requirements: A Process-Oriented Approach", *IEEE Transactions on Software Engineering*, Vol. 18, No. 6, pp. 483-497, June 1992.

[Nawrocki 02] J. Nawrocki et. al., "Extreme Programming Modified: Embrace Requirements Engineering Practices", *Proceedings of the IEEE Joint International Conference on Requirements Engineering*, September 2002, pp.303-310.

[Neill 03] C. Neill, "The Extreme Programming Bandwagon: Revolution or Just Revolting?", *IT Pro*, Vol. 5, No. 5, September/October 2003, pp. 64, 62-63

[Nixon 00] B. Nixon, "Management of Performance Requirements for Information Systems", *IEEE Transactions on Software Engineering*, Vol. 26, No. 12, pp.1122-1146, December 2000.

[O'Neal 01] J. O’Neal, D. Carver, “Analyzing the Impact of Changing Requirements”, *Proceedings of the IEEE International Conference on Software Maintenance*, November 2001, pp. 190-

195.

[Paetsch 03] F. Paetsch, A. Eberlein, F. Maurer, "Requirements Engineering and Agile Software Development", *Proceedings of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2003, pp. 308-314.

[Pinhiero 96] F. Pinhiero, J. Goguen, "An Object-Oriented Tool for Tracing Requirements", *IEEE Software*, Vol. 13, No. 2, March 1996, pp. 52-64.

[PMI 00] Project Management Institute PMBOK Team, *A Guide to the Project Management Body of Knowledge (PMBOK)*, PMI, 2000.

[Pohl 93] K. Pohl, "The Three Dimensions of Requirements Engineering", *Proceedings of Advanced Information Systems Engineering,* 1993, pp. 275-292.

[Pohl 96] K. Pohl, "PRO-ART: Enabling Requirements Pre-Traceability", *2nd International Conference on Requirements Engineering*, 1996, pp. 76-85.

[Pohl 97] K. Pohl, R. Domges, M. Jarke, "Towards Method-Driven Trace Capture", *Proceedings of the 9th International Conference on Advanced Information System Engineering*, 1997, pp.103-116.

[Pohl 99] K. Pohl, K. Weidenhaupt, R. Domges, P. Haumer, M. Jarke, and R. Klamma, “PRIME–Toward Process-Integrated Modeling Environments,” *ACM Transactions on Software Engineering and Methodology*, vol. 8, no. 4, pp. 343-410, Oct. 1999.

[Preiss 01] O. Preiss, A. Wegmann, "Stakeholder Discovery and Classification Based on Systems Science Principles ", *Second Asia-Pacific Conference on Quality Software*, December 2001, p. 194.

[Ramesh 01] B. Ramesh and M. Jarke, “Toward Reference Models for Requirements Traceability”, *IEEE Transactions on Software Engineering*, Vol. 27, No. 1, pp. 58-93, January, 2001.

[Ramesh 95] B. Ramesh, T. Powers, C. Stubbs, M. Edwards, “Implementing Requirements Traceability: A Case Study”, *Proceedings of the Second Int’l Conference on Requirements Engineering*, 1995, pp. 89-95.

[Ramesh 98] B. Ramesh, “Factors Influencing Requirements Traceability Practice”, *Communications of the ACM*, Vol. 41, No. 12, pp. 37-44, December 1998.

[Rational] Rational Software (IBM): http://www.rational.com

[Reifer 01] D. Reifer, *Making the Software Business Case: Improvement by the Numbers,* Addison-Wesley, 2001.

[Reifer 02] D. Reifer, "How good are Agile Methods?", *IEEE Software*, July 2002, pp. 16-18.

[RUP] RUP Website: http://www-306.ibm.com/software/awdtools/rup/

[Scrum] Scrum Website, http://www.controlchaos.com.

[SEI 02] CMMI Product Team, Capability Maturity Model Integration (CMMI), SEI, 2002.

[Sharp 99] H. Sharp, A. Finkelstein, "Stakeholder Identification in the Requirements Engineering Process", *10th International Workshop on Database & Expert Systems Applications,* September 1999, p. 387.

[Shull 02] F. Shull, et. al, "What We Have Learned About Fighting Defects", *Eighth IEEE Symposium on Software Metrics,* June 2002, p. 249.

[SoftwareMag] SoftwareMag.com, "Standish: Project Success Rates Improved Over 10 Years", January 2004, http://www.softwaremag.com/L.cfm?Doc=newsletter/2004-01-15/Standish.

[SOX] Sarbanes-Oxley Act: http://www.sarbanes-oxley.com/

[Standish] The Standish Group, "CHAOS Report", 1994 & 2004, www.standishgroup.com.

[Stephens 03] M. Stephens and D. Rosenberg, "Extreme Programming Refactored: the Case Against XP", Apress, 2003.

[Strens 96] M. Strens, R. Sugden, "Change Analysis: A Step Towards Meeting the Challenge of Changing Requirements", *IEEE Symposium and Workshop on Engineering of Computer Based Systems*, 1996, pp.278-283.

[TeleLogic] TeleLogic: http://www.telelogic.com

[Thorp 03] J. Thorp, *The Information Paradox,* Canada: McGraw-Hill Ryerson, 2003.

[Tryggeseth 97] E. Tryggeseth and O. Nytro, “Dynamic Traceability Links Supported by a System Architecture Description”, *Proceedings of the 1997 International Conference on Software Maintenance*, October 1997, pp. 180-187.

[Tvete 99] B. Tvete, "Introducing Efficient Requirements Management", *10th International Workshop on Database & Expert Systems Applications,* 1999, pp.370-375.

[Ware 04] L. Ware, "The Benefits of Agile IT", *CIO,* August 2004, http://www2.cio.com/research/surveyreport.cfm?id=74.

[Watkins 94] R. Watkins, M. Neal, "Why and How of Requirements Tracing", *IEEE Software*, Vol. 11, No. 4, July/August 1994, pp. 104-106.

[Webster] Marriam-Webster Dictionary online: http://www.m-w.com.

[Weigers 03] K. Weigers, *Software Requirements,* Redmond: Microsoft Press, 2003.

[XP] XP Website, http://www.extremeprogramming.org.