UNIVERSITY OF
CALGARY

UNIVERSITY OF CALGARY

Requirements Engineering for Projects with Critical Time-To-Market

by Christopher McPhee

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA

OCTOBER, 2001

ABSTRACT

Requirements Engineering (RE), although a relatively new discipline, has come a long way from the original rule-of-thumb: requirements should describe *what* is to be done, rather than *how.* Although this rule has become axiomatic, RE has developed a much broader scope. Several RE methods and techniques have been developed to assist large projects in their RE process, but it is not clear that these frameworks will adequately scale-down to be useful in a small project where time-to-market is critical.

This document identifies the essential components of a RE process and provides alternatives to the Requirements Analyst (RA) when deciding on a particular set of RE methods. Existing methods of choosing RE techniques are discussed, and a new technique is proposed specifically for choosing RE techniques for Time-to-Market (TTM) projects.

The components of the essential RE process as well as many other insights into the practice of RE were identified by examining the current literature and by analyzing the data from 25 completed surveys concerning the area of RE. Although too small to be statistically significant, the survey data allows qualitative insight into the state of the practice in RE.

ACKNOWLEDGEMENTS

I would like to thank Dr. Rob Kremer for his assistance in the early stages of my thesis preparation and especially Dr. Armin Eberlein for his support and guidance in my research from inception through to completion. I also thank the individuals who were kind enough to take the time to fill-out the survey that was administered as part of this research.

A special thanks goes to my wife Vicki for all the support, encouragement, and patience she has provided throughout this endeavor.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 CHAPTER ONE: INTRODUCTION

Objectives

- To explain the importance of Requirements Engineering in software development.
- To explain the problem of Requirements Engineering (RE) in Time-to-Market (TTM) projects.
- To introduce the research presented in this thesis.
- To outline the structure of the remainder of this document.

1.1 Requirements Engineering in Software Development

With the Internet becoming more prevalent in terms of a communications medium, computers and information technology are increasingly more a part of daily living. As software applications become more important in day-to-day life, it is necessary that these applications adequately meet the needs of the users. At the same time, the market conditions are such that it is important for companies to get their products to market as quickly as possible so as to gain critical market share [Olsen 1995].

For a company to develop a software product efficiently and successfully, it is important for the RE phase to provide clear direction [Costello 1995]. Unlike other industries where the majority of capital is spent on tangible materials, the tangible materials in a software product are a very small part of the overall cost of the project. By far, the costliest part of a software product is the development of the application. Reproduction costs are measurable and are normally insignificant [Shapiro 1998]. Increased costs resulting from changes to requirements, however, are often much more difficult to measure quantitatively. As a result, software requirements are often quite volatile [Sommerville 1997]. When requirements change, effort is required to modify the product to meet the new requirements, thus increasing the cost and delaying the schedule of the project. If requirements remain volatile for too long, the likelihood of project success decreases dramatically, as modifying software to incorporate new/modified requirements is often complicated and error-prone [Brooks 1995].

1.2 Requirements Engineering in Time-to-Market Projects

With software delivered via the Internet (i.e. web-enabled applications), the volatility of requirements becomes even more of a problem as the reproduction costs shrink to virtually nothing [Shapiro 1998]. In the relatively immature market place of the Internet, consumers are increasingly looking towards the Internet to meet their information and software needs. As a result, there is a strong push from

Information Technology (IT) companies to meet the needs of the customers as quickly as possible in order that the profitability of their ventures be maximized [Card 1995]. To meet the needs of their customers, however, IT organizations must be adept at learning what the customer needs, and developing products to meet those needs as quickly as possible. In addition to meeting the needs of today's customer, IT organizations must be able to quickly adapt to the changing technology and the changing work environments of their customers [Preece 1994].

Unfortunately, RE is not particularly easy to do well. Even when experts are employed in the RE phase to help determine the set of requirements, they may miss up to half of the necessary end-user requirements [Cumo 1994]. To make matters even more difficult, the application of large-scale organizational methodologies to the RE process of TTM projects has been shown to be largely ineffective [Vitalari 1983, Dagwell 1983]. Although it is difficult to do well, a sound RE process is important if a company wishes to achieve repeatable project success [Paulk 1993].

Do the current industry-standard RE techniques address the needs of E-Commerce and Internet development in a market where TTM is critical for many companies and their projects? If so, how does an organization develop an appropriate RE process and choose the right RE tools and techniques to adequately capture and express the requirements for a proposed system?

1.3 Goals of Research

The purpose of this research is to provide guidance to software developers in determining the appropriate steps to take in the RE process for a TTM project and the appropriate techniques to facilitate a successful RE project phase. Nikula et al have shown there to be a need for education and knowledge transfer from academia to industry in terms of RE processes and best practices [Nikula 2000]. In addition to discussing the fundamentals of RE for use in a TTM project, this research also describes an RE technique evaluation process which can assist

software developers in determining the most appropriate set of RE techniques for use on a TTM project.

The impetus behind these research goals is the author's hypothesis that most requirements engineering performed on TTM projects is ad hoc. Further, it is believed that software engineers do not normally explicitly choose a RE technique to use on a TTM project, and if they do, that choice is based primarily on previous experience or on the company standard. Several Requirements Engineering methods and techniques have been developed to assist large projects in their RE process, but it is not clear that these frameworks will adequately scale-down to be useful in a small project where TTM is critical [Vitalari 1983, Dagwell 1983]. Providing a structured methodology for developing a suitable RE process and appropriate RE techniques should help to make this portion of the software life cycle less ad hoc for the development of TTM projects.

Summarized, the goals of this research are:

1. Determine the gaps in software developer knowledge of RE process and techniques. (see Chapter 4)
    a. Determine the current level of knowledge pertaining to RE techniques.
    b. Determine the methodologies followed by software developers in choosing RE techniques.
    c. Determine the attitude of software developers towards the RE phase of a project.
    d. Determine how software developers view RE when comparing TTM and non-TTM projects.
2. Document a technique for evaluating the applicability of RE techniques for TTM projects.
    a. Research and analyze literature in terms of applicability to TTM projects. (see Chapter 2)

b. Determine deficiencies in literature regarding RE process and techniques for TTM projects. (see Chapter 3)

1.4 Description of Document Sections

Chapter Two: Literature Survey – discusses the areas of literature researched and the key findings in these areas. It outlines the relationships between the different areas of literature studied and provides a basis for a critical analysis of the literature.

Chapter Three: Analysis of Literature – presents a description of how the literature surveyed addresses the problem of RE in TTM projects and where gaps remain. This analysis identifies areas where further research is necessary to enable the construction of a method for determining RE process and techniques for use in a TTM project.

Chapter Four: Empirical Research – outlines the research methodology employed, depicts the results obtained and presents an analysis of the data.

Chapter Five: Conclusion – summarizes the information contained in this document, discusses the key findings of the research, and identifies areas of further research.

2 CHAPTER TWO: LITERATURE SURVEY

Objectives

- To present the literature relating to:
  - The RE process
  - RE techniques
  - TTM projects
  - Existing RE technique selection methodologies
  - The literature relating to RE metrics

2.1 Survey of the RE Process

The RE process consists of 4 main areas, as shown outlined in Figure 1. Although the 4 areas are distinct from one another in their process and purpose, they may occur simultaneously in the requirements stage of a project. For TTM projects, it is desirable that these 4 tasks occur at the same time or at least in very small increments. By being able to perform the tasks together, the requirements engineer saves overall schedule time by requiring less overhead to set-up meetings, contact individuals to clarify issues, etc [Card 1995].

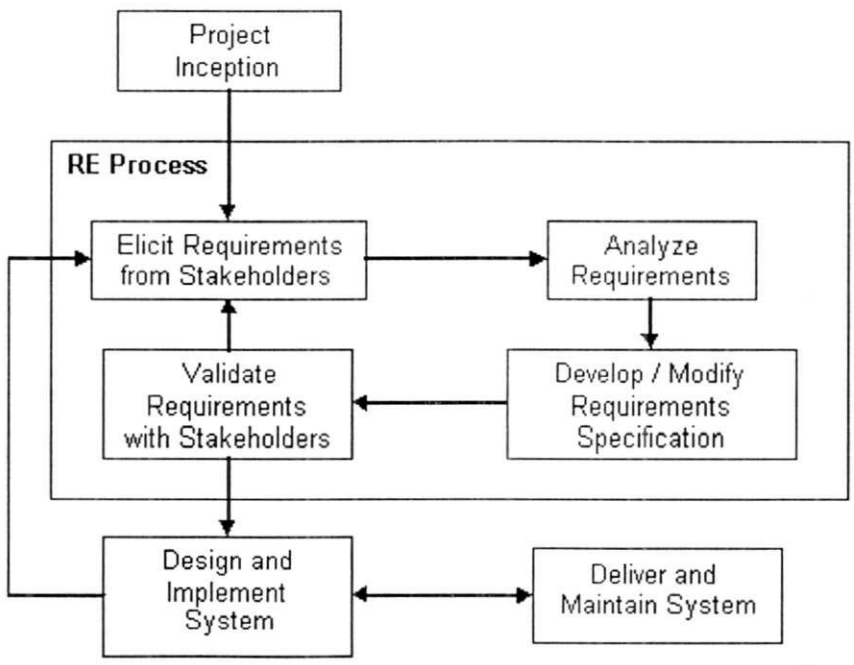

Figure 1 RE Process

It is not necessary for project teams to rush through the RE process for TTM projects. The time spent up-front may have significant time savings during the latter stages of the project. In a survey of 16 Finnish companies, one third reported that even with the ever-increasing need to reduce schedule time, they should be spending more time and effort on RE [Nikula 2000].

2.1.1 Prerequisites of RE

As Requirements Engineering is the first phase of a project, one might assume that there are no prerequisites. However, of the 5 common problems in RE as identified by Sommerville, all are somewhat related to prerequisite activities to the RE process: [Sommerville 1998]

1. Lack of stakeholder involvement.
2. Business needs are not considered.
3. Lack of requirements management.
4. Lack of defined responsibilities.
5. Stakeholder communication problems.

To alleviate the problems relating to stakeholder involvement (points 1, 2, and 5), a solid relationship must be built with the stakeholders before the RE process starts and maintained throughout the project. Points 3 and 4 are not necessarily prerequisites to the RE phase, but they must be addressed very early on in the RE process to improve the chances for a successful project.

2.1.2 Phases of the RE Process

The RE process consists of four main activities: elicitation, analysis, specification, and validation [Sommerville 1998]. Elicitation is the activity of gathering the requirements from stakeholders. After gathering the requirements, they are analyzed to determine areas requiring clarifications, logical groupings, etc. After being analyzed, the requirements are documented and validated with the stakeholder to ensure that the product developed from the requirements will meet the needs of the stakeholder.

Elicitation – The purpose of the elicitation phase is to gather the requirements. Davis suggests that this stage of the RE process is actually more of an exercise in discovery - learning about the problem, understanding who the user is and what the user really wants [Davis 1993]. This phase is absolutely essential, because without it, it would not be possible to build a product at all. Even for software

development projects where no specific customer provides the requirements, the software developer implicitly provides the requirements from which the product is to be developed. However, for TTM projects, and especially for Internet applications, it is not necessary that the entire task of elicitation be performed before subsequent work commences. All that is necessary for TTM projects is that the core requirements be determined. For the first iteration of product development it is necessary that the architecture be built upon a relatively stable set of core requirements. Minor requirements discovered later can then be implemented in subsequent releases [McConnell 1996].

Analysis – The purpose of the analysis phase is to create a complete and consistent set of requirements [Sommerville 1998]. This may include examining the set of requirements to determine if the requirements meet each of the commonly defined attributes of good requirements: unambiguous, complete, verifiable, consistent, modifiable, traceable, and usable [IEEE 1984]. It is quite likely that deficiencies will appear in the set of requirements, requiring that the requirements engineer go back to the customer to clarify ambiguities and possibly elicit further requirements or remove/modify inconsistent or conflicting requirements. Sommerville and Kotonya suggest that the Elicitation and Analysis phases of RE are interleaved in a spiral model, consisting of Elicitation (which produces a draft set of requirements), Analysis (which results in a set of problems with the set of requirements), and Negotiation to resolve the differences [Sommerville 1998]. Intuitively, the smaller the set of requirements, the less time that the analysis phase should take. As was the case in the elicitation phase, if the set of requirements can be kept to the minimum set required to implement the core functionality, the analysis phase should take less time than it would otherwise have taken.

Specification – The purpose of the specification stage is to document the requirements such that they may be validated by the stakeholder and used by the implementers to develop a product that meets the requirements. Without a

specification, there is no way to determine if the final product has met the initial needs of the customer, nor is there any concrete artifact for the implementers to use when constructing the product. The only situation where it would be possible for this phase to be skipped would be where the customer is part of the development team, or the customer has no need to validate the set of requirements or the resulting product. That said, however, it is possible for the specification to consist of a prototype and users' help documentation [McConnell 1998].

Validation – The purpose of the validation phase is to ensure that the set of requirements is necessary and sufficient. The validation phase also allows stakeholders an opportunity to review the requirements to determine if the document, as a whole, consists of a set of well-specified requirements [IEEE 1984, Sommerville 1998]. Not performing this phase greatly increases the risk that the set of requirements will be insufficient for the purpose of creating a successful product. If the project is structured such that only the essential requirements are documented, the validation phase will require minimal effort. The purpose of validation is simply to ensure that the specification accurately reflects the requirements of the customer.

2.1.3 Exit Criteria of the RE Process

The majority of the RE activities should be completed before substantial effort is expended on detailed design and implementation. Concrete exit criteria consist of a documented requirements specification that has been validated by the affected stakeholders. The validated requirements specification might consist of a formal requirements document [Sommerville 1997, Costello 1995], an experimental prototype [Gasson 1995], or a combination of the two. There will likely be stakeholder conflict encountered while developing and validating a requirements specification, but it is much better to start dealing with, and resolving, conflict in the early stages of a project rather than waiting until the end with the hopes that

everything will turn out alright [Macaulay 1996]. A solid RE process requires plans for addressing and resolving conflict [Sommerville 1997].

2.1.4 Common Deficiencies in RE

In addition to the communication-related RE problems identified by Sommerville as listed in section 2.1.1, Macaulay also suggests that many of the problems associated with the RE phase relate to stakeholder communication [Macaulay 1996]. Adair suggests that there are four necessary elements for effective stakeholder communication [Adair 1997]:

1. Social Contact
2. Common Medium
3. Transmission
4. Understanding

In most stakeholder communications, these elements are thought to be implicit in the interaction, but an RE analyst would be well-advised to explicitly ensure that the four elements are always taken into consideration.

Effective communication concerns not only discussion of the system requirements, but also of the underlying domain knowledge. Curtis suggests that domain-level learning by the RE analyst is not generally seen as a legitimate activity, as the system developers are hired for their expertise. However, a fundamental understanding of the domain-specific attributes of a system are required to facilitate a successful determination of the business requirements [Curtis 1988].

Concerning the effectiveness of the RE phase, the Standish Group reported that only 20% of a product's features are commonly used, while 45% are virtually never used [Standish Group 1998]. This suggests that in addition to following best practices for RE, it is also necessary for the RE analyst to work with the stakeholders to determine the relative importance of features. McConnell identifies a technique called Requirement Scrubbing [McConnell 1996] which refers to going

through each documented requirement and removing those that are not essential to project success.

2.2 Survey of TTM Project Attributes

In order for software developers to effectively and efficiently develop software applications that address the needs of their customers, it is prudent that the project team gather and specify the customer needs in a concise and unambiguous manner [Sommerville 1997, Morris 1993]. It is this underlying requirement that has driven the development of many of the RE methods and techniques that currently exist. Many of these techniques, however, are based on the assumption that the organization for which the application is being developed is in a state of stability and that the requirements exist, and they simply need to be gathered and documented [Costello 1995].

In modern society, however, computers and information technology are fundamentally changing the way that people live. Relationships between individuals and organizations change very rapidly, partly because technology allows it and partly to allow for technology [Pressman 1997]. This results in very dynamic and fluid organizations in which there may not exist a definitive set of requirements for a particular software application. Instead, there is often a general and urgent need for capability, but it is almost impossible to concisely and unambiguously specify that need [Truex 1999]. Even if it were possible to specify the exact needs of a group, its needs may have changed by the time that the product is developed.

In this climate, software development companies who identify a need for a particular product often require that the product be developed very quickly in order for the product to be profitable [Olsen 1995].

Card [1995] suggests that market conditions affect the type of software product that will be most profitable. The following figure depicts the relationship between market conditions and the most profitable software product type:

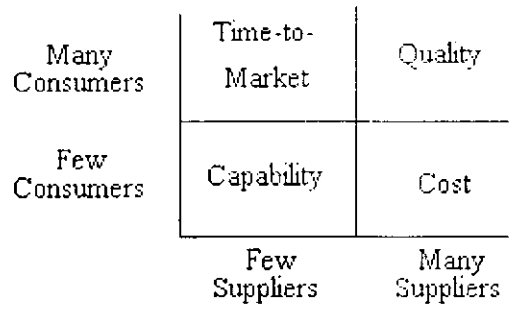

Figure 2 Market Conditions and Related Project Types [Card 1995]

In market conditions where many customers, but few suppliers, exist in a relationship with a particular product, it is often Time-to-Market (TTM) that determines the overall attractiveness of a product (to the customer) or profitability of producing a product (for the supplier), which, in turn, determines the overall success of a product [Card 1995]. For many sectors of the modern IT market, these conditions exist [Pressman 1997]. As such, software developers require a method of determining how to best perform requirements engineering so as to decrease overall time-to-market without compromising product quality [Harlson 1997].

Given that fast time-to-market projects are the main focus of this research, is there any reason to treat this group differently from other software development endeavors? What are the main driving factors behind treating a project as Time-to-Market dependent?

Market Conditions

As software development companies explore economic niches that have not yet matured, the first product is able to garner a critical majority of early market share [Olsen 1995]. As the market place matures, the focus of customers shifts towards value.

Low Distribution Cost

The initial cost may be high when developing information goods (electronic information, web applications, and downloadable software), but the incremental cost of supporting each additional customer is very low when compared to other industries [Shapiro 1998].

Low Switching Costs

Although not always the case, when compared to other areas of business, switching costs are extremely low for E-Commerce products [Shapiro 1998].

Global Marketplace

Given the nature of the Internet, any information good that is connected to the Internet is automatically part of the global environment. This allows for increased visibility, but also means increased competition [Shapiro 1998].

2.3 Survey Rapid Application Development Processes

Rapid Application Development (RAD) is a broad term used in a number of different contexts. The term itself means different things to different people: time-box scheduling, JAD workshops, application generators, rapid prototyping, etc. [Card 1995]. In the context of this research, Rapid Application Development is any development methodology or activity whose overall impetus is to increase speed of application development over that of the traditional development life cycles.

Increased speed of application development can be achieved in three fundamental ways: [Card 1995, p.21]

1. Perform fewer tasks
2. Perform tasks more quickly
3. Perform tasks concurrently

Points 1 and 2 can lead to a decreased amount of development effort, if development effort is defined in terms of person-hours or person-months. Point 3, however, does not achieve decreased development effort since the overall amount

of development effort actually increases as a result of the increased coordination between actors. Point 3 also leads to increased coordination complexity and increases risk in that actors performing concurrent tasks must work from incomplete artifacts and/or must coordinate with other actors performing concurrent activities [Card 1995].

Software development, rapid or otherwise, takes place within the context of a software development model, which defines the general phases of the product development [Costello 1995]. There are several development models including waterfall [Royce 87], incremental [McConnell 1996], and spiral [Boehm 1988].

The waterfall method consists of distinct product development activities in which the subsequent activity does not start before the prerequisite activity has been fully completed [Royce 1987]. These activities include: Requirements Engineering (Software Concept and Requirements Analysis), Design (Architectural and Detail), Implementation (Coding and Debugging), and Testing [Royce 1987]. If changes to a prerequisite activity are not allowed under any circumstances, the waterfall method has very little risk that the project will be delayed as a result of changing requirements. At the same time, it takes a great deal of effort to fully and completely specify all requirements before beginning design, and even then, requirements often change [McConnell 1996, Brooks 1995]. Using the waterfall method, either the product continues and ignores the changing environment, or the product changes to incorporate the changing requirements and thus consumes effort as previously-developed artifacts are reworked.

The incremental life cycle is a commonly-used software development life cycle to increase adaptability and decrease risk of wasted requirements engineering effort. By iterating through the software development activities, the developer can more quickly adapt to changing requirements and thus perform less rework [McConnell 1996]. The spiral life cycle is a specialized iterative life cycle which explicitly takes

risk into consideration when performing software development activities [Boehm 1988].

Extreme Programming (XP) is an example of an iterative life cycle that focuses on customer satisfaction [Wells 2000]. XP uses frequent iterations, usually lasting from 1 to 3 weeks, to keep on top of the customer requirements and the changes that are required as the customer becomes more familiar with their vision of the final product.

2.4 Survey of RE Techniques

RE techniques and tools assist software practitioners in the elicitation, analysis, specification, and/or validation of requirements for a proposed software system [Sommerville 1997]. There are several methods of categorizing RE techniques and tools [Sommerville 1998, Maiden 1996, Bickerton 1992].

RE techniques and tools described in this section have been categorized by the author based on applicable RE phase. That is, the phase of RE in which the RE tool or technique would most commonly be used. For example, although interviews can be used to help validate, or even analyze requirements, they are most commonly used as an elicitation technique. The categorization of the techniques is summarized in Table 1.

Some of the RE tools and techniques are quite similar to one another in either content or purpose, and have been categorized into general-purpose RE technique/tool categories.

Table 1 Applicability of Tools and Techniques to RE Phases

Tool/Technique	Elicitation	Analysis	Specification	Validation
Interviews	Primary			Secondary
Data/Document Mining	Primary			
Cooperative Requirements Capture (CRC)	Primary	Secondary		Secondary
Designer as Apprentice	Primary			Secondary
Observation and Social Analysis	Primary			Secondary
Focus Groups	Primary			Secondary
Future Workshops	Primary			Secondary
Soft Systems Methodology (SSM)	Primary	Secondary		Secondary
Effective Technical and Human Implementation of Computer-based Systems (ETHICS)	Primary	Secondary		Secondary
User Centered Design (UCD)	Primary	Secondary		Secondary
Viewpoint-Oriented Techniques	Primary	Secondary		Secondary
Joint Application Development (JAD)		Primary		Secondary
Quality Function Deployment (QFD)		Primary		Secondary
Informal Modeling	Secondary	Primary	Secondary	Secondary

Tool/Technique	Elicitation	Analysis	Specification	Validation
Semi-formal Modeling	Secondary	Primary	Secondary	Secondary
Scenarios / Use Cases	Secondary	Primary	Secondary	Secondary
Requirements Prioritization	Secondary	Primary		Secondary
Formal Modeling		Secondary	Primary	Secondary
Throw-away Prototyping	Secondary		Secondary	Primary
Evolutionary Prototyping	Secondary		Secondary	Primary
Requirements Testing				Primary
Requirements Reviews		Secondary		Primary
Requirements Tracing		Secondary	Primary	Secondary
Requirements Change Management			Primary	
Requirements Checklists	Secondary			Primary
Requirements Reuse	Secondary		Primary	

2.4.1 Requirements Elicitation Techniques

Interviews

Interviews are discussions between Requirements Engineers and system stakeholders in a question and answer format and are used primarily to elicit or validate requirements [Adair 1997]. Interviews may be structured or unstructured. Structured interviews are used to elicit specific details and require that the interviewer prepare a list of specific questions prior to the interview. Unstructured

interviews are ones where the interviewer prepares a list of general questions, but intends the questions only as a guide for the interview [Macaulay 1996].

Data/Document Mining

Data/Document Mining is the activity of searching through data (documents, forms, computer databases, e-mail notes, etc.) for the purpose of determining context and detailed system information on which requirements are to be based [Morris 1993].

Cooperative Requirements Capture (CRC)

CRC is similar to JAD in that it is a group session approach to system development, but unlike JAD, it explicitly requires the participation of stakeholders who may not otherwise be involved in the development of the system. Participants in a CRC session might include individuals with a financial or regulatory stake in the final product [Macaulay 1996].

Designer as Apprentice

Designer as Apprentice is a technique where the system analyst or developer participates in on-the-job training with end-users of the system. The main purpose of the Designer as Apprentice technique is to give the designer detailed insight into the specific work tasks of the end user who will eventually use the system [Beyer 1995, Reubenstein 1991].

Observation and Social Analysis

Similar to Designer as Apprentice, Observation and Social Analysis includes observing the end-user of the system either working at their current job or working with early releases of the system under development. The purpose of this technique is to gain detailed insight into the work processes followed, and to determine if the end-user has developed any efficiencies in their work that deviate from their task specification [Sommerville 1998].

Focus Groups

Focus Groups are group meetings where open discussion is encouraged. The

discussion provides the analyst with insight into how the individuals think and feel about aspects of the system. Focus groups may be used throughout the requirements engineering process. At the beginning of the project, focus groups may be used to elicit requirements based on what individuals think the system should encompass. During development, focus groups may be used to evaluate prototypes to provide validation and verification of the product. [Templeton 1994]

Future Workshops

Future Workshops are similar to focus groups, but are more structured and are more narrow in scope. Future workshops consist of participants defining the desired future state of the environment. The facilitator(s) ensures that the scope of the environment is defined and is adhered to, and that the participants discuss the desired end-state of the environment. [Macaulay 1996]

Soft Systems Methodology (SSM)

SSM examines the environment within which the system resides. It looks at why the system exists, or should exist, and how it fits into the overall environment. Once a system has been proposed, the differences between the current and proposed system are examined [Checkland 1990]. Like Future Workshops, SSM focuses on a revolutionary change between the current system and the future system, i.e., it determines the end-state and how to achieve the end-state, rather than examining incremental improvements to the current state.

ETHICS

Effective Technical and Human Implementation of Computer-based Systems (ETHICS) is concerned with the impact of a new system on the work processes and job satisfaction of the end-user [Mumford 1986]. Like SSM, it compares the end-state to the current state, but the comparison is based on differences in work process rather than differences in the state of the environment.

User-Centered Design (UCD)

User-Centered Design encompasses a variety of methods and techniques whose

overriding purpose is to design a system to fit the users [Sommerville 1998]. UCD techniques place the user at the center of the design to ensure that the system is built to the needs of the user. Various techniques may be used depending on the overall context of the system [Knight 1989]. Theses techniques include cost-benefit analysis from the user perspective and usability analysis [Macaulay 1996].

Viewpoint-oriented Techniques

Viewpoint-oriented techniques explicitly look at the system from the viewpoints of the various stakeholders. These techniques recognize that one viewpoint is not likely going to be able to specify all requirements of the system, so the various viewpoints are examined in attempts to fully capture the system requirements [Sommerville 1998].

2.4.2 Requirements Analysis Techniques

Joint Application Development (JAD)

JAD is a method whereby system stakeholders work together in facilitated group sessions to specify and perform preliminary development of a system. JAD sessions include representatives in the following roles: session leader (facilitators), user representative, specialist, analyst, information systems representative, executive sponsor [August 1991].

Quality Function Deployment (QFD)

QFD was developed by the Japanese to determine quality requirements for the automobile industry. The primary focus of QFD is the House of Quality which shows the relationship between customer requirements and product features. The roof of the House of Quality shows the interactions between the various product features. In addition to relating features to requirements, QFD also supports market analysis of competitors' products [Akao 1990].

Informal Modeling

Informal Modeling encompasses techniques such as text descriptions and rich pictures. It allows a great degree of flexibility in how the system and its interactions are described. The purpose of informal modeling is to enable the stake-holders to explore and communicate with one another about the system without the requiring that the stake-holders have be familiar with a particular modeling technique. [Macaulay 1996]

Semi-formal Modeling

Semi-formal modeling includes techniques and notations such as Data-Flow Diagrams (DFD), State Charts, and the Unified Modeling Language (UML). Like informal modeling, semi-formal modeling techniques are used to develop a representation of the system for communication and analysis purposes. Unlike informal modeling, however, semi-formal modeling techniques have a predefined set of diagrammatic notations and rules about how systems may be described [Booch 1998, Yourdon 2000]. The pre-defined notations make semi-formal models less ambiguous than informal models, but they are also often less informative to the untrained reader.

Scenarios / Use Cases

Scenarios and use cases allow the system designer to step through a particular sequence of actions or events that describe a portion of the system's behavior. They provide a mechanism for understanding and communicating the sequence of actions (including variants) that take place in a particular portion of the system behavior. [Booch 1998, Jarke 1999, Dano 1997]

Requirements Prioritization

Requirements Prioritization acknowledges the fact that it is not likely possible that every stakeholder will be able to get absolutely everything they want from a proposed system. Requirements prioritization requires that stakeholders list their requirements in such a way as to highlight the most important requirements so that

if trade-offs need to be made because of budget or schedule, there will be some concrete basis for negotiation. [Wiegers 1999]

2.4.3 Requirements Specification Techniques

Formal Modeling

Formal Modeling is used to develop a description of a system that can provide proof of essential (or undesirable) system properties. Unlike informal and semi-formal methods which are mainly concerned with helping the analyst to better understand the system, formal methods are concerned with absolute correctness in a particular aspect of the system's behavior. [Goguen 1997]

2.4.4 Requirements Validation Techniques

Throw-away Prototyping

Throw-away Prototyping, also known as research prototyping, is used either to elicit feedback about a proposed system, or to determine the feasibility of a particular approach to implementation of a system [Brooks 1995]. Throw-away prototypes are created with speed of development in mind, while sacrificing system attributes such as quality and flexibility. Throw-away prototypes are not meant to be built upon, but are meant purely as a means of examining a particular aspect of the system [Sommerville 1998].

Evolutionary Prototyping

Unlike Throw-away Prototyping, Evolutionary Prototyping is a life-cycle model. It is based around the principle that user requirements will change, and so is focused on a flexible design that is delivered to the users in increments with constant adaptation to the system as requirements evolve [Sommerville 1998].

Requirements Testing

Requirements Testing is the activity of defining requirements test cases during the requirements analysis phase. Test cases have to be defined which can be executed in the final system showing that the requirements have been met by the

implementation. This activity often reveals requirements problems before design and implementation occur. [Rosenberg 1998]

Requirements Reviews

Requirements Reviews involve stakeholders reviewing the requirements specification to ensure that it is compliant with their requirements of the system [Rothman 2000]. Requirements reviews might be as informal as stakeholders examining the specification and providing feedback to the requirements engineer, to a formal meeting where minutes are taken and action items are identified. A requirements review is normally complete when the requirements engineer has addressed the feedback from the stakeholders and the stakeholders have agreed to the requirements specification. Even if the requirements are reviewed by the development team alone, reading the specification from various perspectives can help to find defects which might not have otherwise been evident [Schull 2000].

2.4.5 RE Process Tools/Activities

Requirements Tracing

Requirements Tracing involves determining links between system requirements and the source of the requirement, the rationale behind a requirements, or system artifacts (test cases, code modules, design modules, etc.) [Sommerville 1997].

Requirements Change Management

Requirements Change Management is a process in which proposed changes to the system requirements are reviewed and accepted by system stake-holders before the change is incorporated into the requirements specification and, subsequently, the system itself. Requirements Change Management also includes managing changes to the system and system artifacts as the result of a change to the requirements specification. [Lavazza 2000]

Requirements Checklists

Requirements Checklists are used to ensure that important areas of requirements

analysis are not missed [Macaulay 1996]. These areas include: ensuring that the requirements are not built upon premature design, ensuring that several requirements have not been grouped together as one requirement, ensuring requirements clarity, ensuring requirements testability, etc.

Requirements Reuse

Requirements Reuse involves reusing requirements specifications, or portions of requirements specifications, that were developed for previous projects. Requirements reuse can be as simple as reusing a template that identifies common requirements or general requirements areas, to reusing specific requirements when the problem and/or solution domain of two projects is similar. [Lam 1997]

2.5 Survey of RE Technique Selection Methodologies

2.5.1 Consideration of Organizational Factors

Bickerton and Siddiqi [1992] propose a classification of RE techniques to assist individuals in choosing an appropriate method of RE in terms of social assumptions made about society. They suggest that RE techniques implicitly make assumptions about society, and that by examining those assumptions, RE techniques can be chosen to more effectively be used within a given social context.

The following figure depicts the taxonomy of social assumptions that may be used to evaluate RE techniques. [Bickerton 1992, p.182]

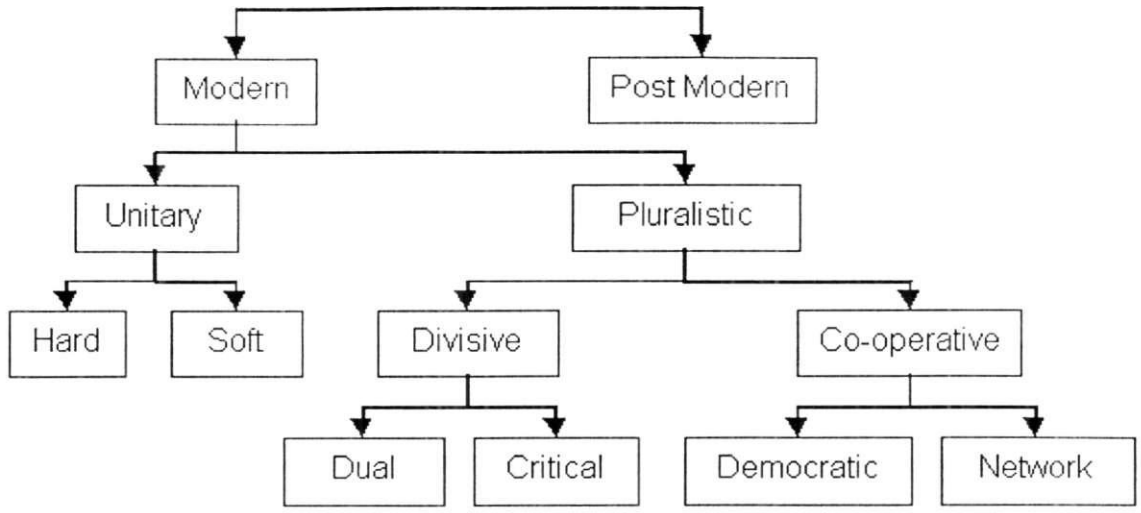

Figure 3 Taxonomy of Social Assumptions

Modern - assumes that objective truth exists, and it is possible to capture that truth at a particular point in time.

Post-Modern - attacks the belief that objective truth exists and that there is no opportunity for universal consensus.

Unitary - views society as an organic self-regulating whole.

Pluralistic - views society as distinct groups.

Hard - assumes organizational structure is of hierarchical nature, stable and formally definable.

Soft - assumes that there are different world views and that the technical system impacts the social system.

Dual - assumes society's fundamental infrastructure consists of conflict between the technology controllers and the technology users.

Critical - society consists of many conflicts between many groups, each seeking to further themselves.

Democratic - power can be exercised through representative forms of management.

Network - views organization as a series of informal networks who work cooperatively to achieve individual gains.

Future work includes gathering empirical evidence to support or refute the taxonomy.

2.5.2 ACRE

A method of evaluating requirements methods has been proposed by Maiden and Rugg [1996] in the framework ACRE (ACquisition REquirements). ACRE provides guidance for acquiring requirements from stakeholders rather than from documentation using the theories and evidence behind cognition and social interaction. ACRE evaluates 12 RE techniques based on 6 evaluation criteria. [Maiden 1996]

Evaluation Criteria

1. Purpose of requirements
2. Knowledge types
3. Internal filtering of requirements
4. Observable phenomena
5. Acquisition context
6. Method Interdependencies

Future work in this area includes gathering empirical evidence to support the effectiveness of using ACRE to evaluate RE techniques as well as strengthening the ACRE framework by interleaving the RE techniques included in ACRE with modeling and validation activities.

2.6 Survey of RE Metrics

Rosenberg, Hammer, and Huffman [1998] identify a set of metrics relating specifically to the requirements specification itself. By collecting and analyzing requirements specification metrics, problems can be found and corrected early in

the development life cycle. Seven measures proposed are as follows: [Rosenberg 1998, p. 3]

1. Lines of text.
2. Imperatives - Phrases that command a specific activity be performed.
3. Continuances - Phrases that introduce lower-level requirements.
4. Directives - References to supplementary information.
5. Weak Phrases - Phrases that are ambiguous and open to interpretation.
6. Incomplete - Phrases that contain references to information that has yet to be determined.
7. Options - Phrases that specify that implementation options are available.

Costello and Liu [1995] propose a different set of metrics for Requirements Engineering processes and artifacts that are meant to provide insight into the effectiveness of the Requirements Engineering phase of a project. The following sections describe the proposed requirements metrics: [Costello 1995, p.53]

2.6.1 Requirements Volatility

Requirements volatility is a measure of the amount and frequency of changes made to the set of requirements. It would normally be expected that requirements would be relatively volatile during requirements engineering, but that the volatility would reduce throughout design, implementation, and testing. If requirements volatility is high in the late stages of a project, it is likely that a great deal of rework will be necessary to implement the modified requirements.

2.6.2 Requirements Traceability

Requirements traceability is a measure of the connection between requirements and information pertaining to, or associated with, the requirements. This might include documenting the connections to source, rationale, higher-level requirements, lower-level requirements, design, code, and testing artifacts. Traceability metrics provide an indication of the complexity of the product as well

as the amount of rework required if and when changes to the requirements specification are required.

2.6.3 Requirements Completeness

Requirements completeness is a measure of the completeness of the requirements specification. This metric identifies how quickly the requirements specification is being developed and how much additional effort is required. A complete requirements specification is obviously desirable because if design and implementation work is done from an incomplete specification, rework will inevitably result. Unfortunately, it is extremely difficult to determine when a requirement specification is complete. This metric is a measure of how completely the requirements have been broken down into low-level requirements, but it does not provide a complete picture of whether or not all of the high-level requirements have been captured. The requirements engineer must be aware that although a requirements specification might be completely analyzed, it still might not be complete.

2.6.4 Requirements Defect Density

Requirements defect density is a measure of the density of defects found during requirements reviews or inspections. Although the time required to fix defects during requirements engineering is likely relatively small, defect density can be an indicator of the number of defects that remain in the specification but were not found during inspection. The validity of this measure is reliant on the experience and competence of the individuals responsible for reviewing the specification.

- The defect density can be a useful indicator of the amount of rework that will be required when undetected defects come to light in the further stages of a project. Requirements engineering techniques which reduce defect density are desirable, as are techniques that assist requirements engineers in identifying defects.

2.6.5 Requirements Fault Density

Requirements fault density is a measure of the problems found with requirements after the requirements engineering phase. The comparison of fault density to defect density provides an indication of the effectiveness of the requirements reviews. Like defect density, fault density also provides an indication of how many undetected faults likely exist within the specification. The determination of this requires the existence of valid historical data on which predictions can be based.

Requirements engineering techniques that reduce fault density are desirable, as are requirements techniques which reduce the impacts of requirements faults. Requirements faults are more complicated to fix than requirements defects because work may have been completed in implementing the faulty requirements. Requirements engineering techniques that allow faults to be fixed with minimal system impact and/or reduced rework are desirable.

2.6.6 Requirements Interface Consistency

Requirements interface consistency is a measure of the consistency between requirements and derived design and implementation artifacts. This metric provides an indication of how well the product artifacts correspond with the requirements specification. Requirements engineering techniques which provide a high level of consistency are desirable as are techniques which allow for the identification of interface consistency.

2.6.7 Requirements Problem Report / Action Item / Issue

Requirements problem report / action item / issue metrics can provide an indication of the consequences of a requirements defect or fault. In practice, this metric would likely only be collected after the requirements engineering phase and would thus refer specifically to requirements faults. In addition to providing the requirement faults metric, this metric can also provide an indication of the impact of requirements changes. Requirements engineering techniques that reduce the

number of problem reports, action items, and issues are desirable because they reduce the rework involved in bringing the product to market.

2.6.8 Requirements Integrated Progress

Requirements integrated progress provides an indication as to the progress of the development of the requirements specification throughout the development progress. This metric is the summation of the other requirements metrics and is meant to provide an overall indication in terms of how much of the requirements specification is completed and how much remains to be done.

2.7 Summary of Literature Survey

It is very important that the RE phase be properly carried-out if a software engineering project is to achieve a reasonable level of success [Rosenberg 1998]. Unfortunately, many projects have tight schedule constraints meaning that a project team has to make decisions about where to focus their time and energy. It is tempting for a project group to perform minimal requirements and jump into design and implementation [McConnell 1996], however this can lead to costly fixes and rework later in the project [Brooks 1995, Boehm 1988].

There has been a great deal of literature and debate on how to best achieve an effective and efficient RAD environment [Card 1995], and methodologies have been developed to assist practitioners in choosing appropriate RE tools and techniques [Maiden 1996, Bickerton 1992], but little research has been done on how to choose appropriate RE tools and techniques in a RAD environment.

The question of how much RE is enough is very important to a requirements engineer in a TTM project, because time spent in RE is time not spent in design and implementation, but requirements defects found in design, implementation, or testing, can be very expensive to fix [Boehm 1988]. Providing guidance to requirements engineers in structuring an RE process for TTM projects would be beneficial as it would allow them to focus their efforts on making informed

decisions about an RE process rather than having to determine an appropriate process (including tools and techniques) on a trial and error basis.

3 CHAPTER THREE: ANALYSIS OF LITERATURE

Objectives

- To discuss the literature survey information in terms of the thesis problem
- To explain the correspondence of RE process and techniques to TTM projects
- To identify the deficiencies in the literature survey information in terms of the thesis problem

3.1 Analysis of RE Process Descriptions

Developing a minimal yet sufficient RE process is not an easy task. If too little RE is performed at the beginning of a project, the upfront effort saved may be dwarfed by the rework effort expended [Brooks 1995, Stark 1998]. If too much time is spent in the RE phase of a project, the overall project schedule may be jeopardized. In fact, Truex et al suggest that there is no such thing as a stable set of requirements and that projects should plan for maximum flexibility and maintenance if a product is to be successful [Truex 1999].

3.1.1 Applicability of RE Process Phases to TTM projects

Based on the discussion in section 2.1.2, it is in a project's best interest to devote some amount of effort to each of the four main phases of RE (elicitation, analysis, specification, and validation). To assist the software developer in the RE phase of a project, the developer may choose to employ an RE technique. However, focusing effort on implementing one RE technique is not enough to ensure adequate result for minimal effort. In addition to using a main RE technique, a set of core activities is suggested by McPhee [2000] to help develop a minimal yet sufficient set of requirements.

Core RE tools/techniques include:

- Requirements Prioritization
- Requirements Checklists
- Requirements Reuse
- Requirements Tracing
- Requirements Reviews
- Requirements Change Management

By including the core RE activities in conjunction with the chosen RE technique(s), the project group improves the overall quality of the RE specification without requiring that these factors be explicitly covered in the main RE technique(s).

3.1.2 RE Process Tailoring for TTM projects

Structuring an RE process around the construction of a minimal set of high-priority requirements still does not completely define the RE process for a TTM project. Software project teams may choose to implement a RE technique to assist them in eliciting, analyzing, specifying, and validating the requirements.

3.2 Analysis of RE Techniques

There are three fundamental methods of reducing the time required to complete a task or set of tasks [Card 1995]:

1. Perform tasks more quickly.
2. Perform tasks concurrently.
3. Perform fewer tasks.

In this section, each of the requirements engineering techniques examined in the literature survey are analyzed in terms of the three ways of reducing time required mentioned above (summarized in Table 2). The purpose behind this analysis is to determine if particular RE techniques allow for reduction of time required either for requirements engineering, or for the overall project. It is possible to do very little requirements engineering, thus reducing the time required for requirements engineering to essentially nil. This, however, puts the project at great risk as time may be wasted reworking the product as requirements are discovered [Stark 1998]. The analysis of the RE techniques assumes that the technique, either by itself or in combination with other techniques, will be used to construct a reasonably complete requirements specification within the initial requirements engineering phase of the project.

To provide for a quantitative analysis, each technique is rated against the three aforementioned ways of reducing schedule time. The rating is based on the following scale:

1 = The technique makes no allowance for the criterion.

2 = The technique may be tailored to make allowance for criterion.

3 = The technique explicitly makes allowance for criterion.

Table 2 Applicability of RE Tools and Techniques to TTM Projects

Tool/Technique	Faster Tasks	Concurrent Tasks	Fewer Tasks	Total
JAD	3	3	2	8
Informal Modeling	2	3	2	7
Semi-formal Modeling	2	3	2	7
Scenarios / Use Cases	2	3	2	7
Evolutionary Prototyping	2	3	2	7
Requirements Testing	2	3	2	7
Requirements Reuse	3	2	2	7
Requirements Checklists	3	1	2	6
Requirements Reviews	3	1	2	6
Requirements Prioritization	2	1	3	6
Formal Modeling	2	3	1	6
Throw-away Prototyping	2	2	2	6

Tool/Technique	Faster Tasks	Concurrent Tasks	Fewer Tasks	Total
Interviews	2	2	2	6
QFD	2	3	1	6
CRC	2	2	2	6
Designer as Apprentice	1	2	2	5
Observation and Social Analysis	1	2	2	5
Focus Groups	2	2	1	5
Future Workshops	2	2	1	5
UCD	1	2	2	5
Viewpoint-oriented Techniques	1	2	2	5
Requirements Tracing	2	1	2	5
Requirements Change Management	1	1	2	4
SSM	1	1	2	4
ETHICS	1	1	1	3
Data/Document Mining	1	1	1	3

3.2.1 Interviews

Allowance for Performing Tasks more Quickly

The only way that Interviews could allow tasks to be performed more quickly would be to ensure that the interviewer is experienced and is prepared for the interview. Depending on the number of stakeholders, interviews can take a substantial amount of time. Interviewers must be adept at keeping the interview on track and efficiently transferring knowledge from the interviewee.

Rating: 2

Allowance for Performing Tasks Concurrently

Little design or implementation work may be performed before the initial interviews are complete, but if individuals taking part in the interview have sufficient knowledge of both the problem and the solution domain, it is possible that a good deal of requirements analysis could be performed during the interview. With a competent scribe taking notes, it is also possible that specification, and even validation, could be partially performed during the interview.

Rating: 2

Allowance for Performing Fewer Tasks

If care is taken during the interview to accurately capture the discussed requirements and models, it is possible that these artifacts could be used in design, rather than having to recreate the information from memory.

Rating: 2

3.2.2 Data/Document Mining

Allowance for Performing Tasks more Quickly

If documentation is reviewed during the elicitation phase, and is summarized by the requirements engineer, the product developers could use the summarized information when making their decisions rather than having to go through the original information. This would require more up-front effort by the requirements engineer, but could allow the developers to more quickly complete their tasks than would otherwise have been the case. The amount of documentation for a particular problem domain can be substantial, so it might be appropriate for the requirements engineer to meet with the stakeholders to determine which documentation is most important to review. Mining less documentation should take less time, but this must be balanced off against the risk that the requirements engineer might miss reviewing important information.

Rating: 1

Allowance for Performing Tasks Concurrently

Data mining does not provide allowance for performing tasks concurrently.

Rating: 1

Allowance for Performing Fewer Tasks

Data mining does not provide allowance for performing fewer tasks.

Rating: 1

3.2.3 Joint Application Design (JAD)

Allowance for Performing Tasks more Quickly

JAD allows tasks to be performed more quickly in that it focuses the system designers and developers on constructing system artifacts quickly. The degree to which this is effective depends, to a large degree, on the ability of the facilitator to ensure that responsibilities are delegated and that assigned

tasks are carried out.

Rating: 3

Allowance for Performing Tasks Concurrently

JAD allows tasks to be performed concurrently in RE because, by having stakeholders working together in an organized fashion, elicitation, analysis, specification, and validation may be performed concurrently.

Rating: 3

Allowance for Performing Fewer Tasks

JAD sessions may decrease the number of tasks performed by decreasing the number of meetings required between the stakeholders and the system developers. By having the stakeholders included in the JAD sessions, issues can be resolved immediately rather than requiring the overhead of meetings and further communication to resolve issues discovered by the system designers during requirements analysis.

Rating: 2

3.2.4 Quality Function Deployment (QFD)

Allowance for Performing Tasks more Quickly

The only allowance for performing tasks more quickly that is provided by QFD is if House of Quality diagrams are used throughout the product development. If this is the case, system implementers can easily determine the implications that changing product features will have on customer requirements. Without a House of Quality diagram to refer to, it might be more difficult to track this information down.

Rating: 2

Allowance for Performing Tasks Concurrently

QFD provides little allowance for performing tasks concurrently, although the requirements specification can be updated in terms of traceability during the development of a House of Quality diagram.

Rating: 3

Allowance for Performing Fewer Tasks

QFD does not allow the reduction of the number of tasks.

Rating: 1

3.2.5 Cooperative Requirements Capture (CRC)

Allowance for Performing Tasks more Quickly

CRC does not specifically allow for performing tasks more quickly, but if CRC sessions are moderated by a skilled facilitator, it is possible that conflicts between stakeholder requirements could be resolved more quickly than might otherwise be the case.

Rating: 2

Allowance for Performing Tasks Concurrently

CRC does not make allowance for performing tasks concurrently. Depending on the make-up of the CRC team, however, it might be possible to perform the various phases of requirements engineering concurrently (elicitation, analysis, specification, and validation) to the same extent as was applicable with JAD.

Rating: 2

Allowance for Performing Fewer Tasks

CRC could allow fewer tasks to be performed if several stakeholders are consulted at the same time. This decreases the number of meetings required and has the added benefit of the different stakeholder groups

hearing first-hand the concerns of other stakeholder groups. The role of the facilitator in such a setting is extremely important, as it is up to the facilitator to ensure that the meeting progresses and does not get off track.

Rating: 2

3.2.6 Designer-As-Apprentice

Allowance for Performing Tasks more Quickly

Designer-as-Apprentice does not make allowance for performing tasks more quickly. It is a time consuming technique but, as was the case in JAD and CRC, the focus on Designer-as-Apprentice is not to decrease schedule time but to increase the chance of project success.

Rating: 1

Allowance for Performing Tasks Concurrently

Depending on the level of customer cooperation, it may be possible for the analyst to perform some requirements analysis and specification during the apprenticeship.

Rating: 2

Allowance for Performing Fewer Tasks

Designer-as-Apprentice makes little allowance for performing fewer tasks, although the better the analyst understands the problem domain at the beginning of the project, the less rework that will need to be performed as a result of unforeseen requirements.

Rating: 2

3.2.7 Observation and Social Analysis

Allowance for Performing Tasks more Quickly

The Observation and Social Analysis method does not make allowance performing tasks more quickly. The length of the observation sessions may have impact on the overall schedule length, but in general, Observation and Social Analysis is a very time consuming requirements engineering technique.

Rating: 1

Allowance for Performing Tasks Concurrently

Depending on the specific process followed by the requirements engineer, some degree of requirements elicitation, analysis, and specification could be performed during the observation sessions.

Rating: 2

Allowance for Performing Fewer Tasks

The Observation and Social Analysis method does not make allowance for performing fewer tasks, except that it may reduce rework if the requirements engineer gains a sufficient level of problem domain knowledge during the process of observing the customer at work.

Rating: 2

3.2.8 Informal Modeling

Allowance for Performing Tasks more Quickly

Informal modeling could allow the requirements engineering process to progress quite quickly if all stakeholders involved are comfortable with models. Informal modeling can also quite easily be used in conjunction with other requirements engineering techniques (Interviews or JAD) to facilitate the performance of these tasks more quickly.

Rating: 2

Allowance for Performing Tasks Concurrently

Informal modeling makes allowance for the performance of tasks within the requirements engineering phase of a project to be performed concurrently. While creating informal models, some requirements analysis and specification (by way of the models created) is also performed.

Rating: 3

Allowance for Performing Fewer Tasks

If the informal models created during the requirements engineering stage are sufficiently self-documented (the symbols are explained), the models could be used by the system developers to gain problem domain knowledge or background for why particular requirements exist rather then having to go back to the customer or requirements engineer for the information. This is not to say that informal models necessarily always speak to the requirements of the system, but they normally provide some background about the problem and/or solution domain.

Rating: 2

3.2.9 Semi-formal Modeling

Allowance for Performing Tasks more Quickly

Semi-formal methods allow tasks to be performed more quickly by allowing for faster and better knowledge transfer. The purpose of a semi-formal technique or notation is to guide the analyst through the development of a model to explore the system in a way that increases understanding of the system and allows that understanding to be transferred to other individuals. The artifacts created using semi-formal techniques also allow tasks in latter phases of a project (design and implementation) to be completed more quickly than might otherwise be possible. If an artifact is created during requirements analysis that would otherwise have been created during design, the designer could use that artifact as is, or extend it for his specific

purposes.

Rating: 2

Allowance for Performing Tasks Concurrently

To some degree, semi-formal methods allow requirements analysis and design to be performed concurrently. It is dangerous to jump to a solution too early in requirements analysis because it is prudent to thoroughly examine options before deciding on a solution. Nevertheless, semi-formal methods encourage the analyst to think through the technical feasibility of a solution which might have an impact on the overall requirements of a system.

Rating: 3

Allowance for Performing Fewer Tasks

Semi-formal methods produce artifacts which may be used in latter stages of a project. Although it might be necessary to modify these models, it might also be possible to use the models as they were developed during the requirements phase. If so, this would eliminate a task that would otherwise have been performed during design.

Rating: 2

3.2.10 Formal Modeling

Allowance for Performing Tasks more Quickly

Formal methods do not make allowance for performing tasks more quickly unless the formal specification allows for automatic code generation. If this is the case, the use of formal methods would speed-up the code generation portion of implementation.

Rating: 2

Allowance for Performing Tasks Concurrently

Formal methods make allowance for performing tasks concurrently in that they allow for validation of completeness of the requirements specification during the specification process itself.

Rating: 3

Allowance for Performing Fewer Tasks

Formal methods do not make allowance for performing fewer tasks.

Rating: 1

3.2.11 Scenarios / Use Cases

Allowance for Performing Tasks more Quickly

Scenarios and use cases allow for performing tasks more quickly in that they provide a structured method for exploring the operation of the proposed system. By going through the system in a structured manner, the requirements engineer can more quickly understand where the customer is unsure of what the requirements really are. The sooner the holes in the proposed system requirements are detected, the sooner they can be filled in. By establishing a good understanding of the system during the requirements stage, the implementation will go more quickly as less rework will be required as a result of finding missing requirements.

Rating: 2

Allowance for Performing Tasks Concurrently

To some degree, scenarios and use cases allow elicitation, analysis, specification, and validation to occur concurrently. By going through operational scenarios with the customer during elicitation, the requirements engineer may be able to perform some analysis and specification during the elicitation session. Once a set of requirements has been determined, it can immediately be validated both with the customer and against the developed

scenarios and use cases.

Rating: 3

Allowance for Performing Fewer Tasks

Scenarios and use cases allow the performance of fewer tasks in that the artifacts that come out of the requirements engineering phase can be used in the design, implementation, and testing stages to both help the individuals doing these jobs to better understand and construct the system. The designers, implementers, and testers need to develop operation scenarios to perform their tasks anyway, so if the scenarios already exist, theses individuals can simply use the scenarios that have already been developed.

Rating: 2

3.2.12 Focus Groups

Allowance for Performing Tasks more Quickly

Focus groups do not make allowance for performing tasks more quickly, although they may be quicker than interviewing individuals one at a time.

Rating: 2

Allowance for Performing Tasks Concurrently

Focus groups do not make allowance for performing tasks concurrently, other than the fact that they may allow the requirements engineer to do some analysis, specification, and validation during the focus group session along with requirements elicitation.

Rating: 2

Allowance for Performing Fewer Tasks

Focus groups do not directly allow for performing fewer tasks, but if the use of focus group sessions improves the understanding of the needs of the

target consumer group, less work will need to be done in reworking the product after it has been released.

Rating: 1

3.2.13 Future Workshops

Allowance for Performing Tasks more Quickly

Future workshops do not directly allow for performing tasks more quickly, although they may be more efficient than speaking to individuals one at a time.

Rating: 2

Allowance for Performing Tasks Concurrently

Future workshops allow for performing tasks concurrently in that they allow the requirements engineer to draft an initial set of requirements during the future workshop session itself.

Rating: 2

Allowance for Performing Fewer Tasks

Future workshops do not make allowance for performing fewer tasks.

Rating: 1

3.2.14 Soft Systems Methodology (SSM)

Allowance for Performing Tasks more Quickly

Soft systems methodology does not allow for performing tasks more quickly. Instead, it is concerned with ensuring that the business is properly modeled from an organizational perspective rather than a purely business or technical perspective.

Rating: 1

Allowance for Performing Tasks Concurrently

SSM does not allow for performing tasks concurrently.

Rating: 1

Allowance for Performing Fewer Tasks

As with other requirements engineering techniques concerned with the effectiveness of the requirements engineering process, SSM does not directly lead to performing fewer tasks, but it does indirectly influence the number of tasks required by helping to ensure that a complete set of requirements is developed early in the requirements engineering process.

Rating: 2

3.2.15 Effective Technical and Human Implementation of Computer-based Systems (ETHICS)

Allowance for Performing Tasks more Quickly

ETHICS does not allow for performing tasks more quickly.

Rating: 1

Allowance for Performing Tasks Concurrently

ETHICS does not allow for performing tasks concurrently. The methodology of ETHICS is based on a 12-step program that does not make allowance for performing steps concurrently.

Rating: 1

Allowance for Performing Fewer Tasks

ETHICS does not allow for performing fewer tasks. The focus of ETHICS is not on creating a solution quickly, but on creating a solution that will provide a smooth and efficient transition for the users.

Rating: 1

3.2.16 User-Centered Design (UCD)

Allowance for Performing Tasks more Quickly

UCD does not allow for performing tasks more quickly.

Rating: 1

Allowance for Performing Tasks Concurrently

UCD allows for performing tasks concurrently in that the some portion of design may be completed during the requirements engineering phase. Practically, very little design should be completed before a majority of the requirements engineering phase has been completed, but UCD helps to focus the requirements engineer's perspective.

Rating: 2

Allowance for Performing Fewer Tasks

UCD allows for performing fewer tasks only to the extent that if the requirements are fully captured at the beginning of a project, less rework will need to be performed in the latter project stages.

Rating: 2

3.2.17 Throw-Away Prototyping

Allowance for Performing Tasks more Quickly

Throw-away prototyping may allow for performance of tasks more quickly because it facilitates the transfer of knowledge from requirements engineer to designer to implementers. If the user interface has been explored during the requirements engineering phases, the designers and implementers can more quickly develop the system based on their knowledge of the user's interface needs.

Rating: 2

Allowance for Performing Tasks Concurrently

To some extent, developing a throw-away prototype allows for both requirements engineering, design, and implementation to occur concurrently. The major risk here is that, during the development of the prototype, the software engineer will make premature design and implementation decisions.

Rating: 2

Allowance for Performing Fewer Tasks

Since the throw-away prototype should not be reused, throw-away prototyping does not directly reduce the number of tasks performed. If the stakeholders are comfortable with the prototype developed, however, it is less likely that the user interface will need to be reworked because of stakeholder dissatisfaction.

Rating: 2

3.2.18 Evolutionary Prototyping

Allowance for Performing Tasks more Quickly

Evolutionary prototyping does not directly allow for performing tasks more quickly in the requirements engineering phase. It might indirectly allow for performing the requirements engineering phase more quickly if the presence of a prototype assists the requirements engineer in more quickly ascertaining and documenting the requirements. More likely, however, the creating of an evolutionary prototype will extend the length of the requirements engineering process as the prototype will need to be developed according to company design and coding standards.

Rating: 2

Allowance for Performing Tasks Concurrently

The creation of an evolutionary prototype allows for concurrent requirements engineering, design, and implementation. Care must be taken, however, not to jump too far ahead on design and implementation as the prototype must remain quite malleable until the end of the requirements engineering process.

Rating: 3

Allowance for Performing Fewer Tasks

Evolutionary prototyping allows for the performance of fewer tasks, especially when compared to throw-away prototyping, as the prototype will continue to be developed in the design and implementation stages thus eliminating the development of an entirely new artifact. The time and effort required to develop the evolutionary prototype, however, might offset this advantage.

Rating: 2

3.2.19 Requirements Reuse

Allowance for Performing Tasks more Quickly

Requirements specification allows the requirements engineer to more quickly perform the requirements engineering phase of a project. The more similar the new project is to previous projects, the more of the requirements specification that can be produced simply by reusing previous requirements specifications. Even if the requirements engineer only uses a requirements specification template to develop the requirements specification, the task of producing the specification can still be performed more quickly than might otherwise be possible.

Rating: 3

Allowance for Performing Tasks Concurrently

To some extent, requirements reuse allows for elicitation, analysis, specification, and validation to be performed concurrently. This is only the case if the new project is relatively similar to previous projects, allowing for direct usage of a portion of the requirements specification of the previous project.

Rating: 2

Allowance for Performing Fewer Tasks

Requirements reuse may allow for performing of fewer tasks if, when the requirements are reused, the reused requirements are marked as such so that the software developers can tell which requirements have been reused from a previous project. With this knowledge, the developers may be able to use artifacts developed by the previous project in the design and implementation stages.

Rating: 2

3.2.20 Requirements Traceability

Allowance for Performing Tasks more Quickly

Although documenting requirements traceability may slow the requirements engineering phase of a project, it may allow design and implementation to go more quickly than would otherwise be the case. If the individuals charged with implementing the system have a better understanding of the relationships between requirements and stakeholders, they will be able to more quickly determine the impact of certain implementation decisions.

Rating: 2

Allowance for Performing Tasks Concurrently

Traceability does not make allowance for performing tasks concurrently.

Rating: 1

Allowance for Performing Fewer Tasks

Traceability information may allow the performance of fewer tasks by reducing the amount of rework needed when decisions are made without the full understanding of their impact. As well, by documenting sources and rationale behind requirements, the system implementer has less work to do if a requirement needs to be changed. If the source and rationale of a requirement is not documented, the implementer will need to spend time tracking this information down.

Rating: 2

3.2.21 Viewpoint-Oriented Techniques

Allowance for Performing Tasks more Quickly

Viewpoint-oriented techniques do not allow for performing tasks more quickly.

Rating: 1

Allowance for Performing Tasks Concurrently

Although viewpoint-oriented techniques do not allow for performing requirements engineering concurrently with design or implementation, it is possible to elicit requirements from various stakeholder groups concurrently, thus reducing the time required for requirements elicitation.

Rating: 2

Allowance for Performing Fewer Tasks

Viewpoint-oriented techniques do not explicitly allow for performing fewer tasks, but they help reduce the risk that design or implementation rework will need to be performed as a result of certain viewpoints not being adequately addressed during the requirements engineering phase of the project.

Rating: 2

3.2.22 Checklists

Allowance for Performing Tasks more Quickly

Checklists allow tasks to be performed more quickly by providing the requirements engineer with appropriate areas to cover. Using checklists not only reduces the time required for the requirements engineer to decide what must be done, but also ensures that those tasks have been performed without important areas being omitted.

Rating: 3

Allowance for Performing Tasks Concurrently

Checklists do not make allowance for performing tasks concurrently.

Rating: 1

Allowance for Performing Fewer Tasks

Checklists allow for performing fewer tasks in that if a task is done right the first time, less total effort is required because the task does not need to be reworked when the deficiencies are inevitably found. For example, if checklists are used to ensure that the requirements specification is rewritten, the designers or implementers do not need to waste time clarifying ambiguous requirements or tracking down information when incomplete requirements are encountered.

Rating: 2

3.2.23 Requirements Prioritization

Allowance for Performing Tasks more Quickly

Although requirements prioritization does not make the requirements engineering stage of a project go more quickly, it definitely speeds-up the design and implementation phases. If the requirements are prioritized, the

designers and implementers can make better judgments concerning which areas of the system to focus their time and effort on. When technical issues arise, the prioritization exercise can also be used for negotiation purposes to determine, along with the customer, what course of action to take.

Rating: 2

Allowance for Performing Tasks Concurrently

Requirements prioritization does not make allowance for performing tasks concurrently.

Rating: 1

Allowance for Performing Fewer Tasks

Requirements prioritization allows for performing fewer tasks if it means that certain requirements ultimately can be left unfulfilled in the final project, while still achieving system success. If budget and/or schedule are of high priority to the customer, reducing requirements will allow the project to fulfill the requirements without using more budget or schedule than the customer is willing to spend.

Rating: 3

3.2.24 Requirements Testing

Allowance for Performing Tasks more Quickly

Requirements testing lengthens the time required to create the requirements specification, but it may allow the designers, implementers, and testers to perform their tasks more quickly. By having a set of requirements that are specific and testable, the individuals in charge of constructing the system do not need to spend time clarifying the requirements.

Rating: 2

Allowance for Performing Tasks Concurrently

Requirements testing allows for performing tasks concurrently in that the requirements engineer creates artifacts that would normally be created later on in the project construction by the designers, implementers, and testers of the system. Including designers, implementers, and testers early in the project allows them to have technical input during a phase where it has the greatest potential impact.

Rating: 3

Allowance for Performing Fewer Tasks

Requirements testing may allow for performing fewer tasks if, because the final requirements specification is of high-quality, time does not need to be spent reworking the specification during the latter phases of the project when the deficiencies in the requirements specification are found.

Rating: 2

3.2.25 Requirements Reviews

Allowance for Performing Tasks more Quickly

Requirements reviews allow for tasks to be performed more quickly because, if moderated by a trained facilitator, requirements reviews can cover the validation of the requirements specification much more quickly and thoroughly than would be the case in an unstructured, non-facilitated meeting. Having a facilitator keep the meeting on track ensures that isolated conflicts or discussions do not waste the time of non-interested stakeholders that might be present at the review.

Rating: 3

Allowance for Performing Tasks Concurrently

Requirements reviews do not make allowance for performing tasks concurrently.

Rating: 1

Allowance for Performing Fewer Tasks

Requirements reviews make allowance for performing fewer tasks because they allow the requirements engineer to validate the specification with several stakeholders at once, rather than having to review it with each stakeholder separately. In addition, any conflicts that arise can be dealt with immediately rather than having to setup a separate meeting to resolve the conflict.

Rating: 2

3.2.26 Requirements Change Management

Allowance for Performing Tasks more Quickly

Requirements Change Management does not allow the performing of tasks more quickly. In fact, the process initially slows the implementation process. This slow down in implementation must, however, be balanced against the increased risk of project failure if unapproved requirements changes cause the stakeholders to find the delivered product unacceptable.

Rating: 1

Allowance for Performing Tasks Concurrently

Requirements Change Management does not allow the performance of concurrent tasks. Developers can continue to develop the system while waiting for a particular change request to be approved, but should not begin work on implementing the change until the change request has been approved.

Rating: 1

Allowance for Performing Fewer Tasks

Requirements Change Management may allow the performance of fewer tasks if unapproved requirements changes would have otherwise required rework to back the change out if the change were eventually found to be unacceptable to one or more stakeholders.

Rating: 2

3.3 Analysis of TTM Project Development Methodology (RAD)

The essence of the research focus is to assist software practitioners in performing Requirements Engineering when constraints on the project might tempt the practitioner to perform less than sufficient Requirements Engineering. This poses the question, 'How much Requirements Engineering is necessary?'. The answer to that question is very much project dependent. To narrow the scope of the research to keep it manageable, the focus of this research is on TTM, E-Commerce projects.

Before looking at a necessary level of Requirements Engineering, the relationships between the TTM and E-Commerce must be examined.

3.3.1 Time-to-Market

TTM projects are those where the schedule is the over-riding factor in achieving predicted profitability. As suggested by Card [1995], the maturity of the market affects the profitability of a particular product. Time-to-Market has the greatest effect on profitability when there are many consumers and few suppliers. This is not to say that TTM is not a factor for other stages of market maturity, but it is especially important when many consumers are chasing few products. By getting to market earlier than the competitors, it is more likely that a product will gain a large proportion of the initial market share [Card 1995].

TTM projects are not necessarily short-term projects. A TTM project might be 2 years long in schedule length, while a non-TTM project might be 2 months in

schedule length. The focus of this research is in providing assistance when the software practitioner might be tempted to perform insufficient RE in favor of reducing schedule. In this situation, even individuals working on a 2-year TTM project might not spend adequate time and effort on Requirements Engineering.

3.3.2 E-Commerce

E-Commerce projects are defined as projects in which electronic communication is used as the principle means of conducting a business transaction [Zhong 1998]. In preliminary interviews conducted during this research, it was found that RE in this area is relatively hit and miss. A company comes up with a good idea, and attempts to create a product that the public will probably like [Drury 1999].

Given the market conditions in today's society, a large portion of E-Commerce projects are TTM [Shapiro 1998]. Several new economic niches are being explored, and the first product to adequately serve that niche will gain a large proportion of initial market share [Card 1995]. However, switching costs are extremely low, so products are early to market need to be of adequate quality so as to retain the early market share gained [Shapiro 1998]. This suggests that although TTM is extremely important for many E-Commerce products, quality should not be sacrificed or the gains made by being early to market may be lost if the product is of poor quality.

Given that little or no physical good need accompany an E-Commerce product, it is possible to make modifications to the product very quickly [Shapiro 1998]. This can be an advantage in terms of getting to market quickly. An initial version may be released very quickly, and defects can be fixed with relatively little cost once the product is released. However, to adequately meet the needs of a particular niche, the product must still address the key requirements even if it is relatively inexpensive to make updates to the product [Buren 1998, Stark 1998].

Finally, the Internet provides for global competition with few constraints based on geographic location [Shapiro 1998]. This means that for a product to be successful, RE techniques used in constructing the product must allow for comparison against other products [Akao 1990]. This is no different than most other types of products, with the exception that new E-Commerce products can come to market more quickly than physical products which require a distribution medium [Shapiro 1998]. Since competing products may come to market very quickly, RE techniques for E-commerce must be flexible enough to allow for comparison against new and unanticipated products.

The following diagram shows the relationship between E-Commerce and Requirements Engineering:

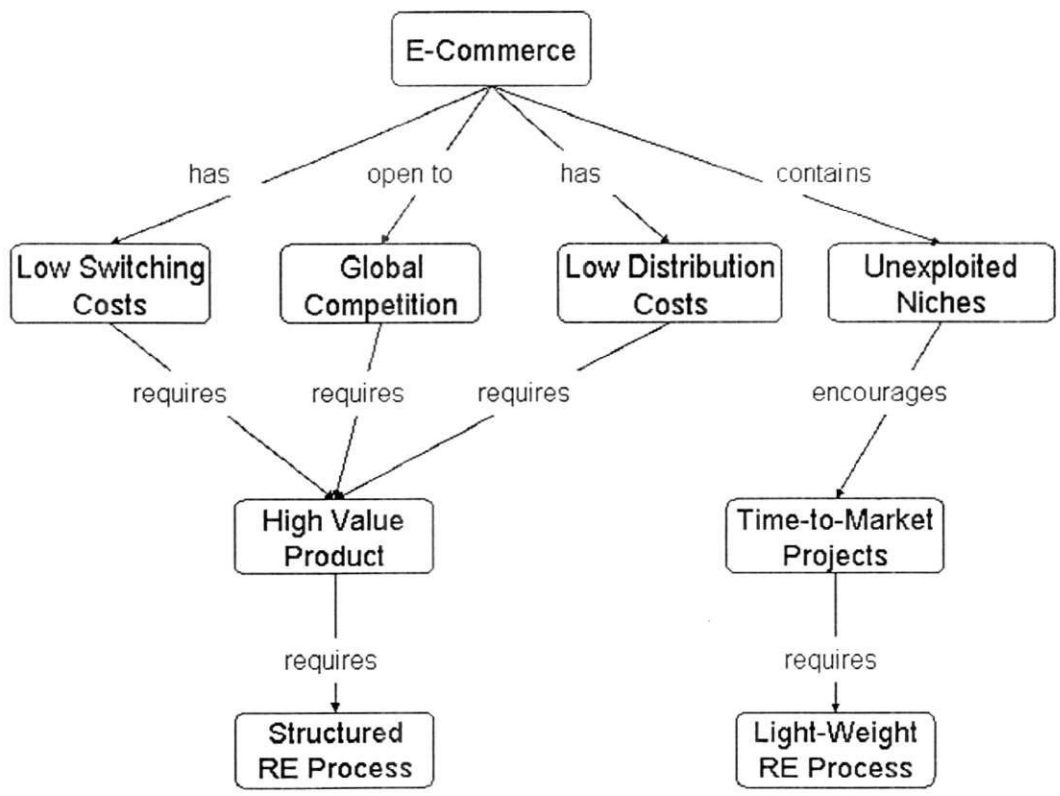

Figure 4 Relationship between E-Commerce and Requirements Engineering

3.4 Analysis of RE Technique Selection Methodology

In the literature survey, two methodologies for evaluating requirements engineering techniques were discussed: Taxonomy of Social Assumptions [Bickerton 1992] and ACRE (ACquisition REquirements) [Maiden 1996]. Both methodologies attempt to assist the requirements engineer in choosing an appropriate method of requirements engineering based on certain qualities of the project or environment.

ACRE [Maiden 1996] assists the requirements engineer by guiding the requirements engineer through the process of evaluating the project against 6 proposed criteria, and then choosing the requirements method(s) that achieve the best fit. The requirements engineering techniques utilized in the ACRE methodology are derived from theories of cognition and social interaction. Similarly, the project attributes used in determining an appropriate methodology are related to the notions of cognition and social interaction to determine which RE technique will provide the best knowledge transfer from stakeholder to requirements engineer.

ACRE does not attempt to examine all, or even a large subset, of available RE techniques. Therefore, using this method to determine an RE technique might help to determine the best of the 12 possible choices, but it does not ensure that the requirements engineer is using the best technique available. ACRE is also fully concerned with achieving a high-quality knowledge transfer with no reference to effort required in implementing the technique. Although the method might provide the requirements engineer with an indication of which technique will best facilitate knowledge transfer, it does not provide the requirements engineer with an indication of cost versus benefit of the chosen technique. The concept of cost versus benefit is very important for TTM projects, as a high quality requirements specification is of little use if the product fails because it missed its schedule deadline.

The taxonomy of social assumptions [Bickerton 1992] provides the requirements engineer with assistance in choosing a requirements engineering technique by having the requirements engineer observe the project in the context of social assumptions. The hypothesis is that by choosing a technique which best fits within the social assumptions of a given environment, the requirements engineer will ultimately be more successful in determining the requirements than would be the case if the social assumptions were ignored. This methodology provides a mapping between the determined location of the organization in the taxonomy and appropriate requirements engineering techniques.

Although the taxonomy provides a relatively extensive listing of requirements engineering techniques, the majority of the techniques fall into the category of Unitary Hard. If it is determined that a given organization falls into that category, this methodology does not provide any assistance in further determining the most appropriate RE technique. As with ACRE, the taxonomy is not specifically concerned with TTM projects, and thus does not factor the effort required to implement a technique.

Although there are current methods for determining a set of requirements engineering techniques for use on a project, it is evident that the current methods are insufficient for practitioners requiring guidance in TTM projects. The current methods for determining RE technique appropriateness are concerned with the effectiveness of a particular technique, but ignore the efficiency of the technique. Although effectiveness and efficiency often go hand-in-hand, the relationship cannot be taken for granted. For requirements engineers working on a TTM project, the current methodologies for choosing a set of requirements engineering techniques are insufficient to fully meet the needs of the requirements engineer.

3.5 Analysis of RE Metrics

In order to objectively verify that a certain requirements engineering technique is better than another in a given circumstance, it is necessary to collect and analyze

metrics data. Without metrics data, only qualitative analysis can be performed in the comparison of two requirements engineering techniques. Although qualitative analysis can provide important and useful information, metrics data is required for a quantitative analysis of a comparison of requirements engineering techniques. Although the usefulness of the metrics is self-evident in most cases, little mention of the cost versus benefit is discussed. For a TTM project, the overhead of any metric collection initiative must be carefully scrutinized. Any schedule time used in collecting and analyzing metrics takes away from schedule time available for working on implementing and releasing the product. For TTM projects, it is desirable to reduce the amount of overhead required to collect a metric, or ensure that the effort expended on collecting the metric is justified by the information provided by the metric.

The act of collecting some of the metrics data implicitly requires the use of certain requirements engineering techniques. For instance, in order to determine the defect density of a requirements specification, the technique of requirements reviews must be implemented. In order to measure requirements volatility, a requirements change management process must be in place. If the metrics collection is a by-product of the requirements engineering process in use, it is an obvious candidate as the only additional overhead is that required to analyze the data and act upon it if necessary. If the collection of the metric, however, requires additional changes to the process either by requiring the implementation of an additional RE technique or overhead required simply to collect the metric, the requirements engineer must ensure that the additional effort is justified.

3.6 Summary of Analysis of Literature

Requirements engineering is an important phase of any software development process. If the requirements are not carefully examined, a great deal of time and effort may be spent in modifying the software product to conform to the new or modified requirements. Although there are several techniques available to the

requirements engineer, there is little guidance available in determining which requirements engineering techniques to use when time-to-market is an important factor in product development.

Two methods of determining an appropriate RE method were discussed in this document: a taxonomy of social assumptions and ACRE. Although these methods provide some assistance in deciding upon an appropriate requirements engineering technique, neither method examines the efficiency. That is, the benefit that the requirements engineering technique provides versus the cost of implementing the technique.

In order for individuals working on a TTM project to make an educated decision on what requirements engineering techniques to use, they need to have access to information concerning the advantages, disadvantages, efficiency, and effectiveness of certain techniques. Since schedule is of the utmost importance in a TTM project, even the process of choosing a requirements engineering technique must be efficient.

4 CHAPTER FOUR: EMPIRICAL RESEARCH

Objectives

- To outline the goals and of the empirical research
- To identify the specific research performed and results obtained
- To discuss the results in terms of the thesis problem

4.1 Goals of the Empirical Research

As discussed in section 1.3, the main goals of the thesis research are to:

1. Determine the gaps in software developer knowledge of RE process and techniques; and
2. Document a technique for evaluating the applicability of RE techniques for TTM projects.

The focus of the empirical research is to gather and analyze data relating to Goal 1, as goal 2 was largely addressed by way of the literature survey and critical analysis of the literature survey.

4.2 Methodology of the Research

An on-line survey was constructed to gather information from individuals involved in software development. The survey covered several different aspects of RE, with the goal of obtaining greater insight into how software developers view the RE phase and what tools or techniques they see as being most useful. Where appropriate, survey respondents were asked to compare TTM and non-TTM projects to determine where the two types of projects were different in terms of approach to RE.

A web-based survey form was constructed and notices were posted to several RE and software development news groups and list servers. In addition to the posted notices, several individuals were contacted by phone and provided answers to the survey questions during a phone or in-person interview.

The following sections discuss the areas of information presented in the survey and the rationale supporting the inclusion of questions in those areas. Refer to Appendix A for the complete survey as it was presented to respondents.

4.2.1 Work Experience

The survey was not targeted at any particular group of software developers. Rather, it was anticipated that responses would be gathered from individuals with diverse backgrounds. This section asked questions regarding experience both in terms of years of development experience and regarding project types and industry domains.

The purpose of collecting this data was to determine if there is any correlation between an individual's experience and his views on RE in software development.

4.2.2 Project Success Factors and Priorities

4.2.2.1 Project Success Factors

Project success factors are those factors which influence the overall success of a project. Specifically, three questions that were asked that have been identified as being important project success indicators [Hartman 2000]:

- How do you know when a project is done?
- How do you know if a project was a success?
- What is the relative importance of budget, schedule, and scope/quality?

The answers to these questions provide insight into how a software developer sees a successful project coming together and what they believe to be the most important factors in software development projects. By relating the answers to these questions to the RE phase of a project, a guide can be tailored to providing information that encourages the Requirements Analyst to set-up an RE phase so that the project can be set-up for success.

4.2.2.2 Project Priorities

Respondents were asked to rank the relative importance of Schedule, Budget, and Scope/Quality for their projects. The answers to these questions show whether or not S/W developers find that schedule plays a factor in their decisions when

implementing a S/W solution to a problem. If so, this suggests that a tailored guide for RE in TTM projects would be a practical reference for S/W developers.

4.2.3 Attributes of 'Good' Requirements

This section lists the seven attributes of good requirements as defined by the IEEE [1984]. On a TTM project, it may not be possible, or even desirable, to spend time and effort to ensure that each of the attributes is fulfilled. For instance, one attribute of a good requirements is identified as being unambiguous [IEEE 1984]. Truex [1999] argues that we should not strive for complete unambiguity, because the environment is always changing and there must be flexibility built into the system requirements.

The responses to questions in this section provide guidance in helping software developers choose suitable RE techniques. If some techniques are appropriate for achieving a particular attribute of a requirement that is deemed to be important for a TTM project, then, all other things being equal, that technique should be used over another technique that does not contribute to achieving the same attribute.

4.2.4 Requirements Engineering Techniques

This section provides a list of RE tools and techniques for which the respondents were to indicate both their knowledge level and the perceived usefulness. If a respondent did not have any knowledge of a tool or technique, they were not to specify its perceived usefulness. Respondents were also asked for the perceived usefulness of each tool or technique on TTM versus non-TTM projects.

This section was included in the survey for three reasons:

1. Determine general knowledge of available RE tools and techniques.
2. Determine overall usefulness of tool or technique.
3. Determine whether there was a perceived difference in usefulness of tools and techniques for TTM versus non-TTM projects.

It is the author's belief that there is a general lack of knowledge regarding available RE tools and techniques. If confirmed, this would provide legitimacy for constructing a guide for RE for software development practitioners.

By determining the usefulness of a technique and an idea of whether or not certain techniques are useful on TTM versus non-TTM projects, information could be included in the Guide to RE biased towards the techniques that are most useful for TTM projects.

4.2.5 Personal Preferences regarding Requirements Engineering

This section asks questions concerning personal preferences regarding the RE phase of a project. Questions related to whether or not an individual liked performing RE and the attributes of good Requirements Analysts.

The answers to these questions can be used to further refine the usefulness of RE tools and techniques. If, for example, a number of individuals reported that one downside of RE is the amount of documentation that must be created and maintained, RE techniques which incorporate the construction of documentation as a by-product of the technique would be considered more desirable than another technique which treats the construction of documentation as a separate activity.

4.2.6 Importance of Requirements Engineering

This section asks respondents how much effort should be spent in the RE phase of a project in terms of percentage of overall project effort. Respondents were asked to determine percentages for both TTM and non-TTM projects.

Although not contributing directly to assisting a Requirements Analyst in determining an appropriate RE process, the responses could be used to provide software developers with guidance in terms of the amount of effort that should be spent in the RE phase of a project.

4.3 Results and Analysis

Of the 25 individuals who responded to the survey, 11 filled-out the on-line questionnaire and 14 provided responses via in-person interviews. All questions were answered when the individuals participated in an interview, but the online respondents did not always answer all questions.

4.3.1 Work Experience

4.3.1.1 Work Experience Results

14 respondents answered questions relating to their work experience. The following chart shows the number of years of software development experience for the respondents:

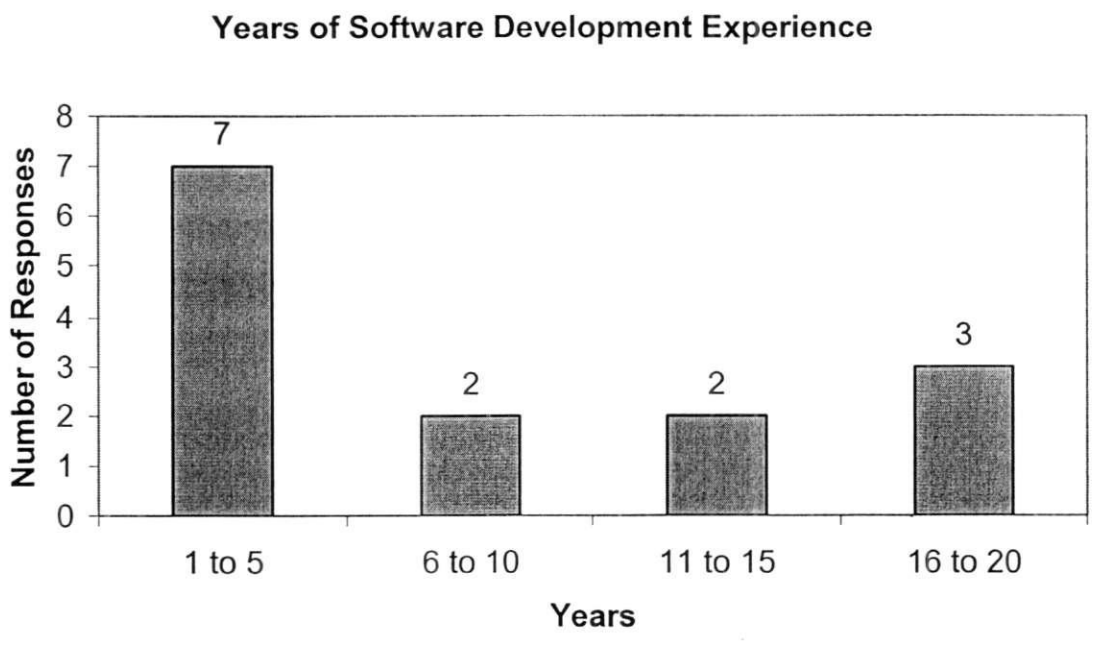

Figure 5 Years of Software Development Experience

Respondents were also asked what percentage of their career was spent in TTM projects. The following chart depicts their answers:

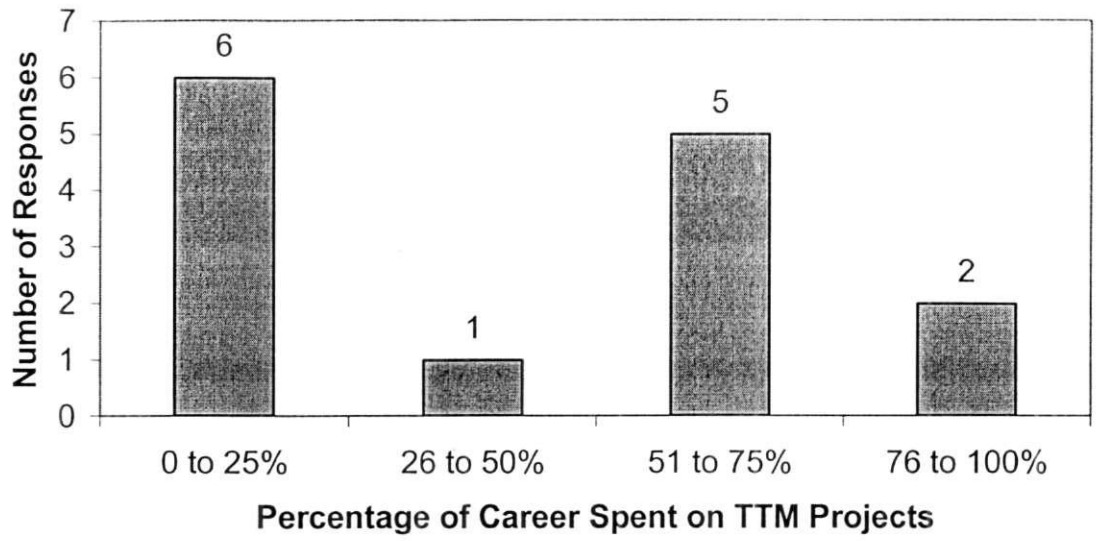

Figure 6 Respondent Experience in TTM Projects

Respondents were then requested to break-down their project experience by the amount of time on projects of different schedule lengths. Each respondent was presented with a set of time frames for which they were to breakdown their project experience. For instance, one respondent might answer:

Table 3 Project Length Sample Response

Project Length	Percentage of Career spent doing Projects of this Length
2 weeks to 1 month	5%
1 to 3 months	10%
3 to 6 months	10%
6 to 12 months	50%
1 to 2 years	25%
2 to 4 years	0%
> 4 years	0%
Total	**100%**

The results from all respondents were averaged for each time division, resulting in the following distribution:

4.3.1.2 Work Experience Analysis and Discussion

Work experience in and of itself does not provide much useful information, other than to note the following points for future reference:

- Approximately half of the respondents had 10 years of software experience or more.
- Approximately half of the respondents have spent the majority of their careers working on TTM projects.
- The most common project lengths are:
  - 6 months to 1 year (26%)
  - 1 to 2 years (21%)
  - 2 to 4 years (26%)
- In total, experience level in each of the 5 phases of a S/W development project lifecycle was almost the same, with Maintenance being the least (15%) and Implementation being the most (25%).

4.3.2 Project Success Factors and Priorities

4.3.2.1 Project Success Factors Results

The questions regarding project success factors were open-ended, so only general trends can be observed.

The following are the general trends in how respondents determined that a project was **done**:

1. User or QA signoff and/or customer acceptance
2. Client takes ownership
3. All requirements have been satisfied

And these are the general trends in how respondents determined that a project was **successful**:

1. Customer is happy/satisfied (expectations have been met).
2. Estimates have not been grossly exceeded.

3. All requirements have been met.

4.3.2.2 Project Success Factors Discussion and Analysis

The top responses both in terms of project completion and project success involved customer sign-off and satisfaction in the final result. This suggests that there must be some way of evaluating and documenting customer expectations. One method of evaluating and documenting customer expectations is to develop a set of test cases for the requirements. By creating test cases and validating them with the stakeholders at the beginning of the project, we are assuring ourselves that we can accurately determine when a project is done.

For the project to be considered successful, points 1 and 2 are related in that if the estimates have been met, the customer's expectations have implicitly been achieved. The only problem in making this connection is the assumption that estimates have been established for schedule, budget, and scope/quality. If the schedule and budget estimates have been achieved, but the customer is not happy with the quality, their expectations have not been satisfied. This suggests that quality expectations must be determined during the RE phase in order to ensure that a successful product is produced. Again, this highlights the need for a documented set of test cases that, once successfully run, signify the successful completion of the project.

Point 3 under project success factors suggests that once the requirements have been met, the project will be successful. This will only be the case if the requirements adequately reflect the expectations of the client. This appears self-evident, but if the requirements are poorly documented or not sufficient to describe the customer's expectations, all of the requirements might be met, but the product still does not meet the customer's expectations.

To combat the possibility that requirements are satisfied, but the client is unhappy with the final result, a requirement for issue resolution should be made part of the requirements specification. The customer must have some assurance that their

expectations, issues, and general changes in project direction are properly handled. This point justifies the existence of a section in the Guide to RE to explain the importance of RE change management and should be performed in the overall context of the software development cycle.

4.3.2.3 Project Priorities Results

When asked about the relative priority of Budget, Schedule, and Scope/Quality, the respondents answered as follows:

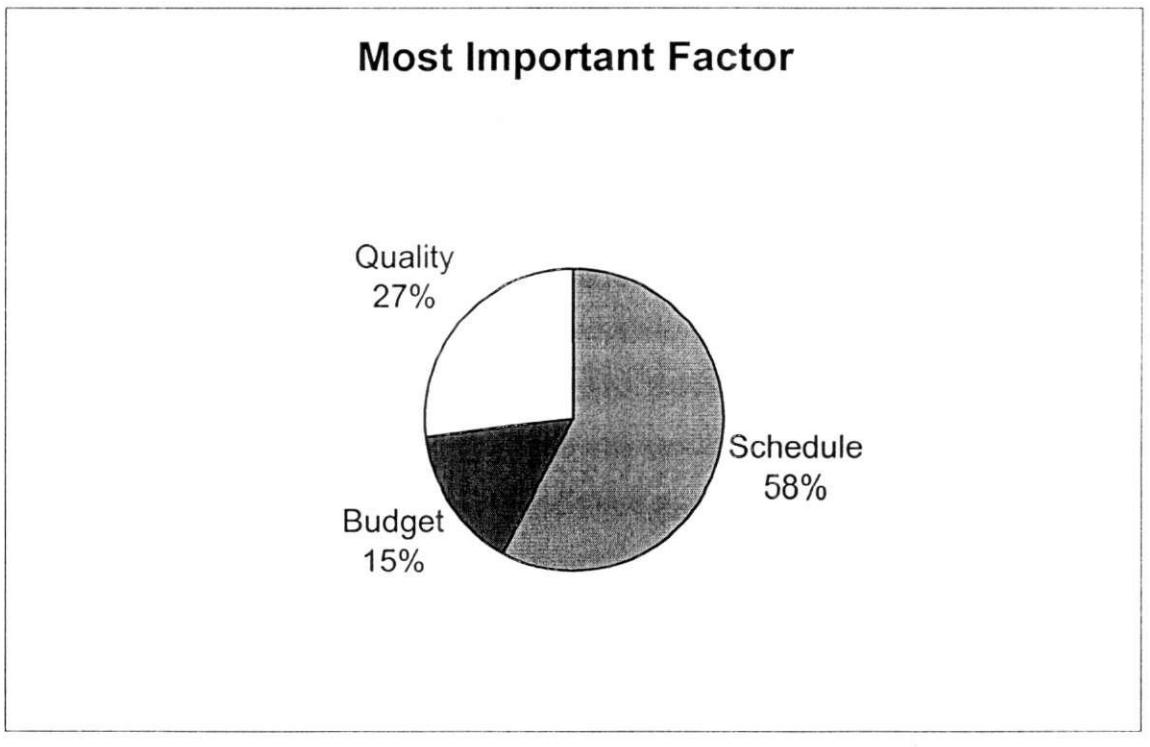

Figure 9 Most Important Factor in Software projects

4.3.2.4 Project Priorities Analysis and Discussion

Schedule was considered the most important project priority by the majority of the respondents. This observation supports the need for a Guide to RE that focuses on TTM projects, simply because practitioners feel that schedule is of great importance.

An analysis of priority importance versus RE technique familiarity and usefulness is included in section 4.3.4.2.

4.3.3 Attributes of 'Good' Requirements

4.3.3.1 Attributes of 'Good' Requirements Results

Respondents were asked to rate RE attributes on a scale of 1 (attribute is of little or no importance), to 5 (attribute is of great importance). The following figure shows the combined responses of all respondents with the maximum score being 100% (all respondents answered 5 for the attribute importance) and the minimum score being 0% (all respondents answered 1 for the attribute importance).

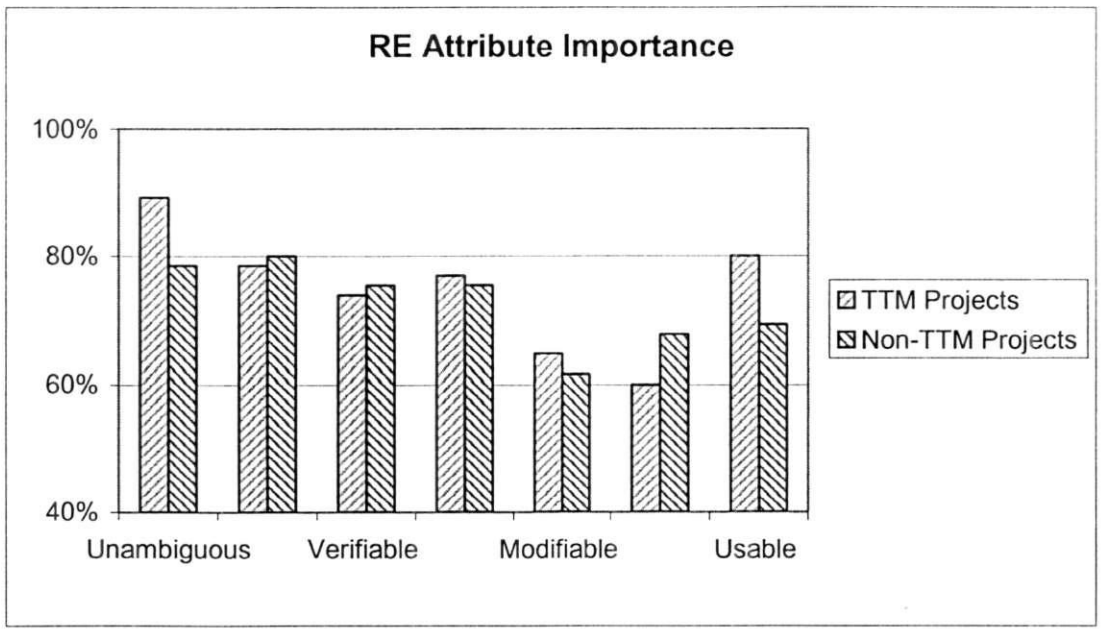

Figure 10 Importance of Requirement Attributes

4.3.3.2 Attributes of 'Good' Requirements Analysis and Discussion

There was little difference between TTM and Non-TTM projects in terms of what individuals felt to be the importance of each attribute. Attributes that were felt to be slightly more important in TTM projects were 'Unambiguous' and 'Usable', while 'Traceable' was felt to be of greater importance for Non-TTM projects. These differences suggest that the respondents are comfortable with more ambiguity in

the requirements for Non-TTM projects than they are on TTM projects. That said, 'Unambiguous' was still deemed the most important attribute for TTM projects, and the second most important attribute for non-TTM projects.

4.3.4 Requirements Engineering Techniques

4.3.4.1 Requirements Engineering Techniques Results

Respondents were asked to rate their familiarity and the perceived usefulness (both for TTM and for non-TTM projects) of several RE techniques on a scale of 1 to 5, with 1 being no familiarity/usefulness and 5 being very high familiarity/usefulness. When rating the usefulness of techniques, only respondents who indicated that they had at least some familiarity were asked as to the effectiveness of the technique.

Figure 11, Figure 12, and Figure 13 depict the scores that each of the RE technique achieved, based on the respondents' ratings. The ratings have been converted to a percent scale, where a perfect score (100%) is achieved by a particular technique receiving a rating of 5 from all respondents, and a score of 0% is the result of all respondents giving the technique a rating of 1. The percent scale was used to allow for comparison of effectiveness across techniques.

Only effectiveness ratings from respondents with some degree of familiarity were used in the overall effectiveness calculation. If this was not done, then techniques for which few people had familiarity would be skewed towards the lower end of the effectiveness scale.

The following formula was used to calculate a techniques effectiveness:

$$\frac{\sum(\text{rating} - 1)}{\text{Number of ratings} * (\text{Maximum rating} - 1)}$$

1 was subtracted from both the individual rating and from the Maximum rating to allow for an overall effectiveness scale that ranged from 0% to 100%.

The following examples illustrate the use of this formula:

Example 1:

If 6 respondents indicated that they had some familiarity with a technique (i.e. rated their familiarity as greater than 1), and each of those respondents indicated that the technique had a usefulness of 4 for a TTM project, then the overall usefulness of the technique for TTM projects would be 75% (6*(4-1)=18 out of a possible 6*(5-1)=24).

Example 2:

If 10 respondents indicated that they had some familiarity with a second technique (i.e. rated their familiarity as greater than 1), and each of those respondents indicated that the technique had a usefulness of 3, then the overall usefulness of the technique was assigned 50% (10*(3-1)=20 out of a possible 10*(5-1)=4).

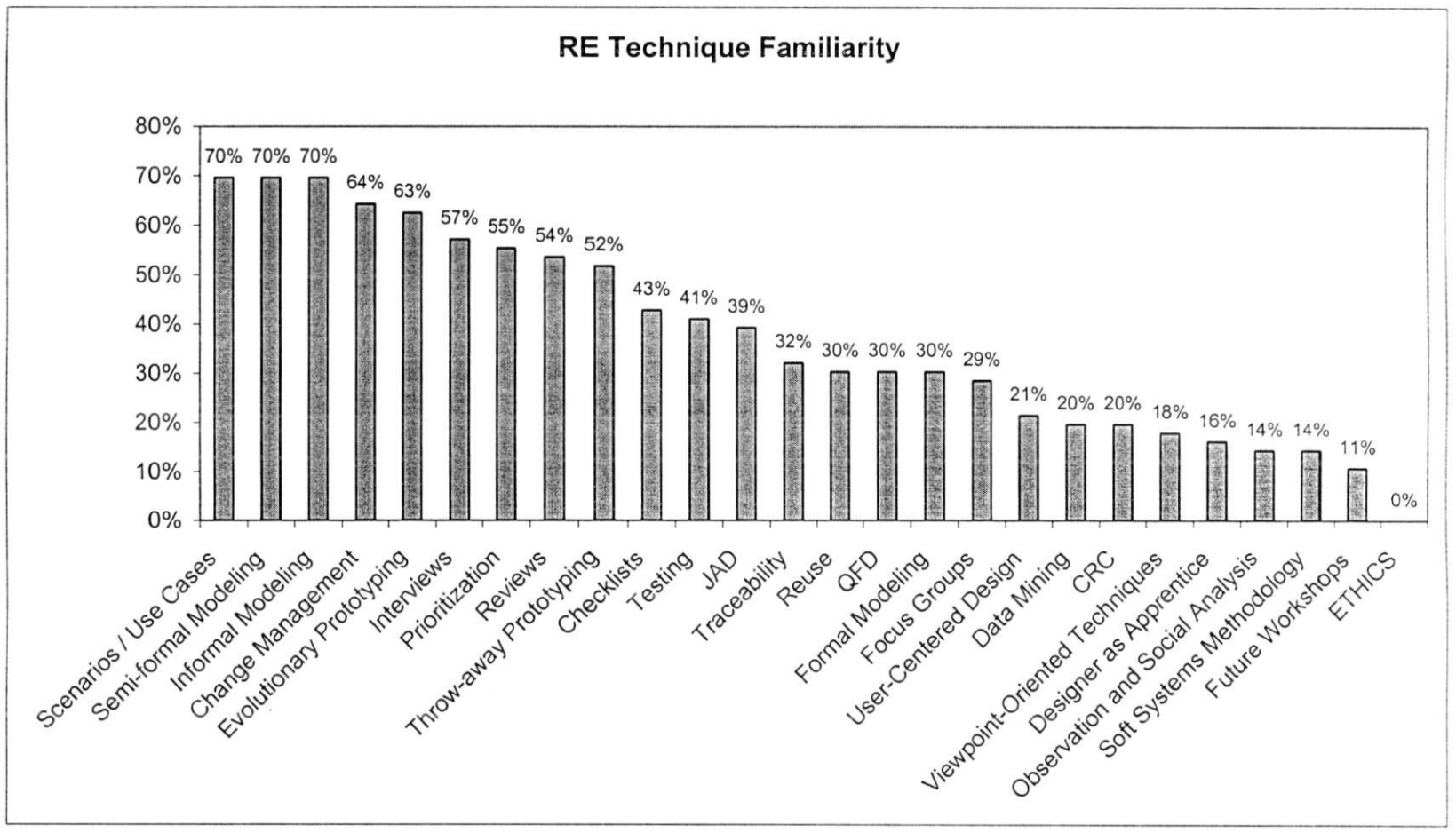

Figure 11 RE Technique Familiarity

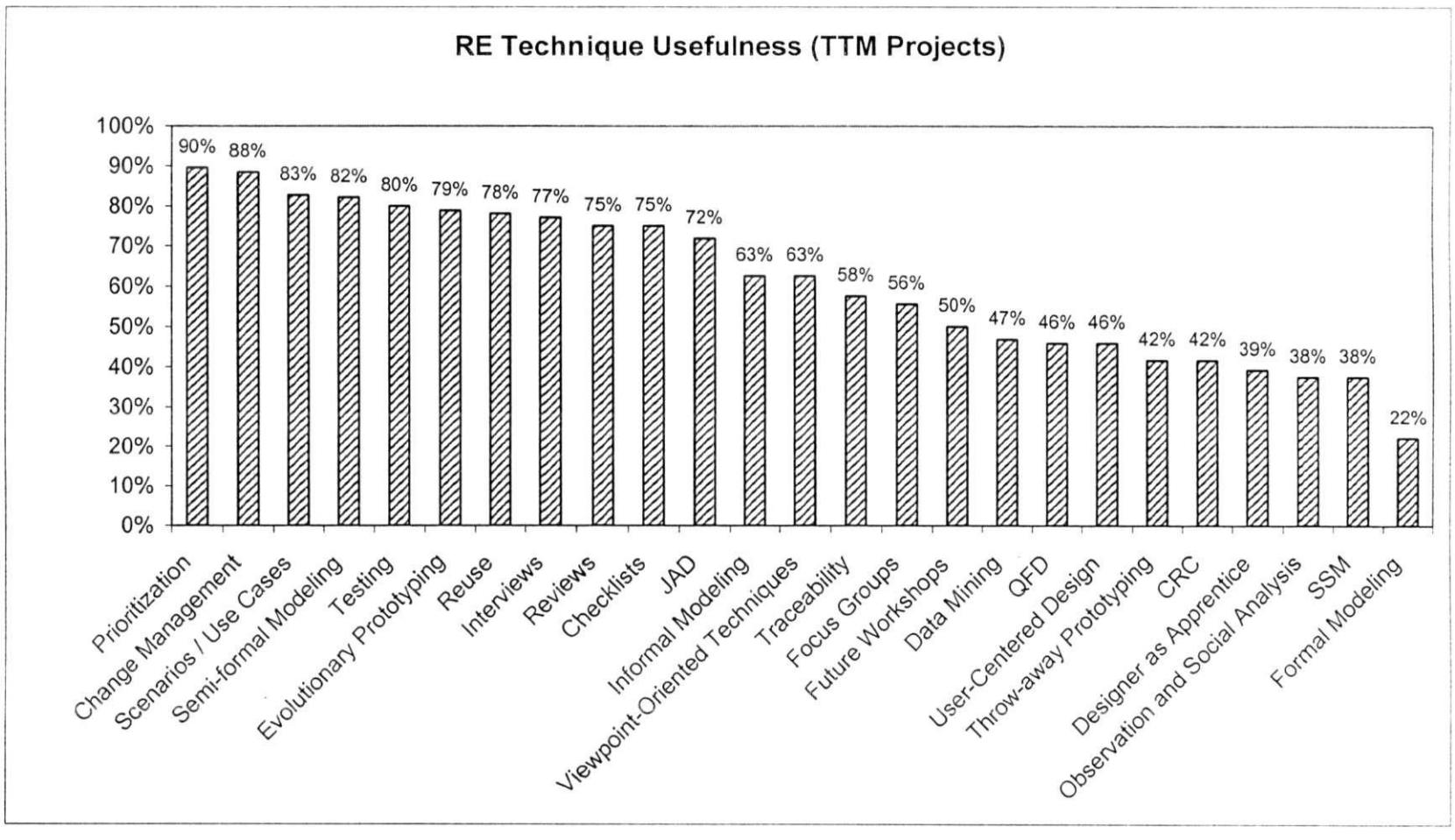

Figure 12 RE Technique Usefulness (TTM Projects)

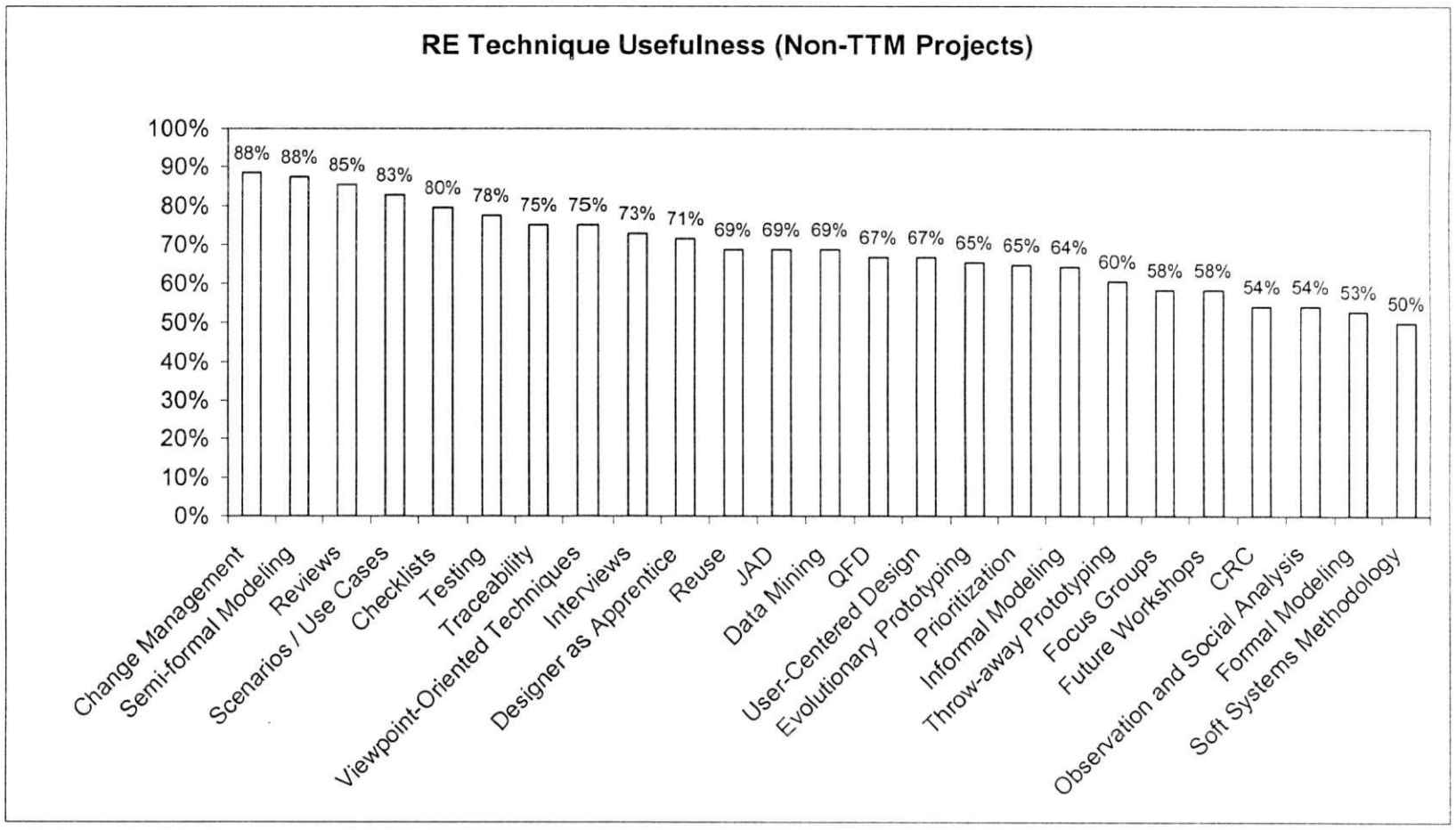

Figure 13 RE Technique Usefulness (Non-TTM Projects)

4.3.4.2 Requirements Engineering Techniques Analysis and Results

The following table shows the top 10 techniques in terms of familiarity, usefulness on TTM projects, and usefulness on non-TTM projects:

Table 4 Top 10 RE Techniques

Rank	Familiarity	Usefulness (TTM)	Usefulness (Non-TTM)
1	Scenarios / Use Cases	Prioritization	Change Management
2	Semi-Formal Modeling	Change Management	Semi-Formal Modeling
3	Informal Modeling	Scenarios / Use Cases	Reviews
4	Change Management	Semi-Formal Modeling	Scenarios / Use Cases
5	Evolutionary Prototyping	Testing	Checklists
6	Interviews	Evolutionary Prototyping	Testing
7	Prioritization	Reuse	Traceability
8	Reviews	Interviews	Viewpoint-Oriented Techniques
9	Throw-away Prototyping	Reviews	Interviews
10	Checklists	Checklists	Designer-as-Apprentice

Although the rankings were not identical, many of the same techniques are included in the top ten for all three categories. Most noticeably, **Scenarios / Use**

Cases, **Semi-Formal Modeling**, and **Change Management** are all ranked in the top five. Not surprisingly, **Prioritization** was thought to be the most useful technique for TTM projects, but did not even register on the top ten techniques for non-TTM projects. Techniques that normally take more effort in terms of the Requirements Analyst's time such as **Traceability**, **Viewpoint-Oriented Techniques**, and **Designer-as-Apprentice** were thought to be useful for non-TTM projects, but not as useful for TTM projects.

The rank correlation coefficient was calculated for three combinations of the RE technique data. A rank correlation coefficient was calculated rather than an absolute correlation coefficient because it is the relative usefulness of a technique that is of interest. All techniques were used in the calculation of the rank correlation coefficient except for ETHICS. No respondent indicated any familiarity with ETHICS, so no information could be determined as to the technique's usefulness on either TTM or Non-TTM projects.

The following table shows the results of this calculation:

Table 5 Rank Correlation Coefficients for RE Techniques

Comparison	**Rank Correlation Coefficient**
Familiarity of Technique versus Usefulness (TTM)	0.729
Familiarity of Technique versus Usefulness (Non-TTM)	0.876
Usefulness (TTM) versus Usefulness Non-TTM	0.679

Since 25 techniques were examined, the sample size of the rank correlation was 25. The following table shows the statistically significant correlation coefficients for a sample size of 25:

Table 6 Statically Significant Correlation Coefficients

Confidence Limit	Minimum Correlation Coefficient
95 %	0.3961
99 %	0.5053
99.9 %	0.6178

All three rank correlation coefficients were greater than the minimum correlation coefficient required for a 99.9 % confidence limit. These results leads to the following observations:

1. Technique familiarity is positively correlated with perceived usefulness of a technique, both for TTM and Non-TTM projects.
2. Perceived usefulness of RE techniques for TTM projects is positively correlated with perceived usefulness of techniques for Non-TTM projects.

Another point of interest is that for usefulness of non-TTM projects, no technique got a score of less than 50% (except for ETHICS for which no respondent was familiar). This suggests that for a non-TTM project, every RE technique is perceived as being more useful than not. In TTM projects, however, approximately one third of the techniques registered a score of less than 50%. Although S/W developers see the needs and usefulness of RE techniques in ensuring a high-quality end product, they consider the range of techniques appropriate for TTM projects to be more limited.

An additional rank correlation coefficient was calculated to determine the correlation between perceived usefulness of RE techniques for TTM projects versus the appropriateness of RE techniques for TTM projects as calculated by the process described in section 3.2. The rank correlation coefficient for these two datasets is 0.366 which is less than the minimum rank correlation coefficient for

95% confidence limits for a sample of this size, meaning that the correlation is not statistically significant within 95% confidence limits.

This result leads to two possible conclusions:

1. The technique for determining appropriateness of RE techniques for TTM projects as described in section 3.2 is not valid; or
2. Since perceived usefulness of RE techniques for TTM projects is highly correlated with technique familiarity, software developers incorrectly perceive some techniques to be more appropriate than others.

The survey conducted for the purposes of this research was only able to ascertain individual familiarity and experience relating to RE techniques, and not the absolute validity of RE technique appropriateness for TTM projects. As a result, it is entirely possible that respondents incorrectly perceive some techniques to be appropriate for TTM projects based on their familiarity with those techniques. To determine which conclusion is correct, a formal experiment would need to be conducted to determine the absolute effectiveness of RE techniques. Given the number of techniques involved, such an experiment is beyond the scope of this research.

The following is an analysis of how individuals who rated Schedule as the most important factor responded to questions regarding RE technique familiarity and usefulness:

Table 7 Comparison of RE Technique Responses and Project Priorities

Perspective	Average Score of Individuals who Indicated Schedule as Most Important	Average Score of Individuals who Indicated a Different Priority than Schedule as being Most Important
RE Technique Familiarity	41%	54%
Usefulness of Techniques for TTM Projects	68%	73%
Usefulness of Techniques for Non-TTM Projects	70%	63%

Respondents who reported that Schedule was normally their highest priority had, on average, less familiarity with the RE tools and techniques, as their average score was less than the average score of individuals who chose Budget or Scope/Quality as being most important in 2 out of 3 categories. This suggests that individuals who mainly work on TTM projects generally have less knowledge about available RE tools and techniques. Intuitively, this makes sense as those individuals who are not time-constrained might be more willing and able to research the RE tools and techniques available to them.

The perceived overall usefulness of RE tools and techniques matched the type of projects in which an individual is normally involved. I.e., individuals for whom schedule was normally the most important factor reported that RE techniques are generally more useful for TTM projects, and individuals for whom schedule is not normally the most important factor reported that RE techniques are generally more useful for non-TTM projects. This suggests that perceived usefulness of RE tools and techniques is influenced by prior experience in a particular type of project.

4.3.5 Personal Preferences regarding Requirements Engineering

4.3.5.1 Personal Preferences Results

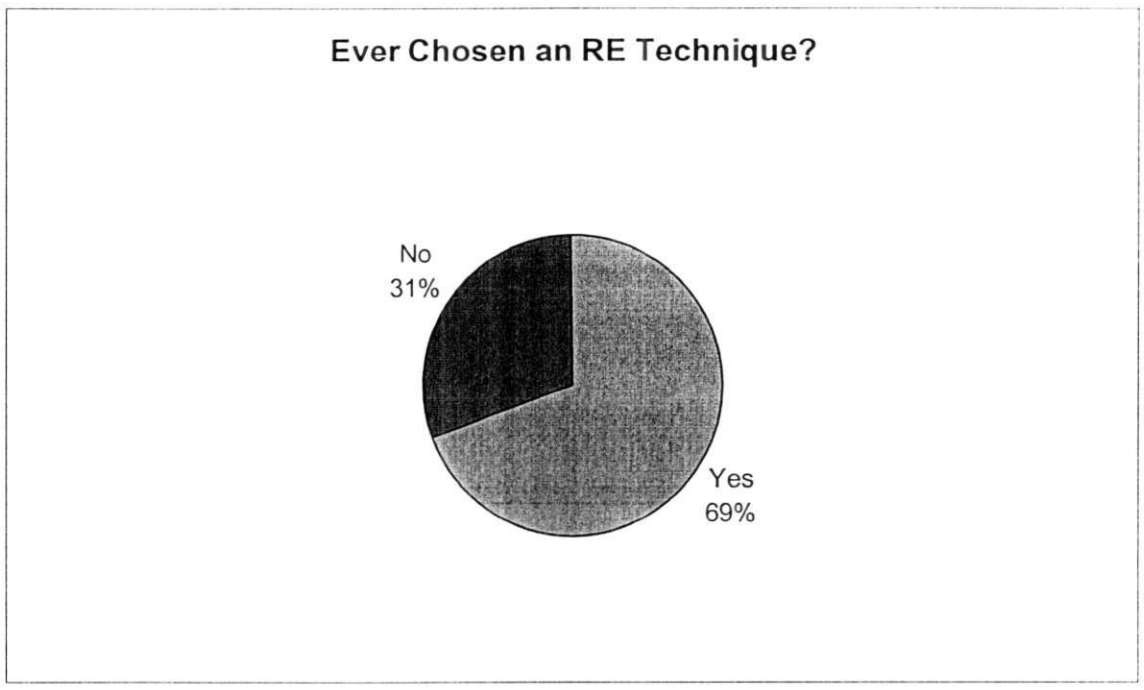

Figure 14 Percentage of Respondents who have Chosen an RE Technique

For those respondents who indicated that they had chosen a technique, they were then asked to identify how the technique was chosen. Most respondents indicated several reasons, and the number of times each reason was reported is identified in the following table:

Reason for Choice of Technique	Number of times Identified
Previous Experience with Technique	7
Facilitates Good Communication	6
Company Standard	4
Tool Support Available	3

Table 8 Reasons for Choosing RE Technique

When asked about the attributes of a good RE technique, all respondents, regardless of whether or not they had chosen a technique before, were asked to respond. The following table shows those responses:

Desirable Quality of RE Technique	Number of times Identified
Easy to Use	8
Facilitates Good Communication	7
Facilitates capture of complete set of requirements	6
Allows for or incorporates traceability	5
Improves the product quality	2
Tool support	2
Incorporates prioritization	1

Table 9 Attributes of Good RE Techniques

Respondents were also asked about what they liked and disliked about RE. The following tables show the responses:

Attributes of RE that are most liked	Number of times Identified
Defining common goals / scope	15
Gaining domain knowledge	7
Exploring possibilities	6
Team building	2

Table 10 Most Liked Attributes of RE

Attributes of RE that are most disliked	Number of times Identified

Attributes of RE that are most disliked	Number of times Identified
Difficulties in communicating with stakeholders	12
Documentation	7
Changing Requirements / Feature Creep	6

Table 11 Most Disliked Attributes of RE

4.3.5.2 Personal Preferences Analysis and Discussion

In almost all of the areas regarding personal preferences in RE, difficulty of communication and/or the importance of communication was high-lighted. This is not unexpected, but it underscores the importance of a good line of communication between the Requirements Analyst and the customer. RE techniques, whether they be applied on a TTM or non-TTM project must enhance the analyst/customer communication in order to be successful.

Another important factor in the usefulness of a technique was the respondents' perceived effectiveness and/or the ease of use of the technique. This is especially important for TTM projects where the Requirements Analyst might not have ample time to become familiar with a new technique. Unless the technique is easy to use, the analyst is more likely to go back to a familiar technique.

4.3.6 Importance of Requirements Engineering

4.3.6.1 Importance of Requirements Engineering Results

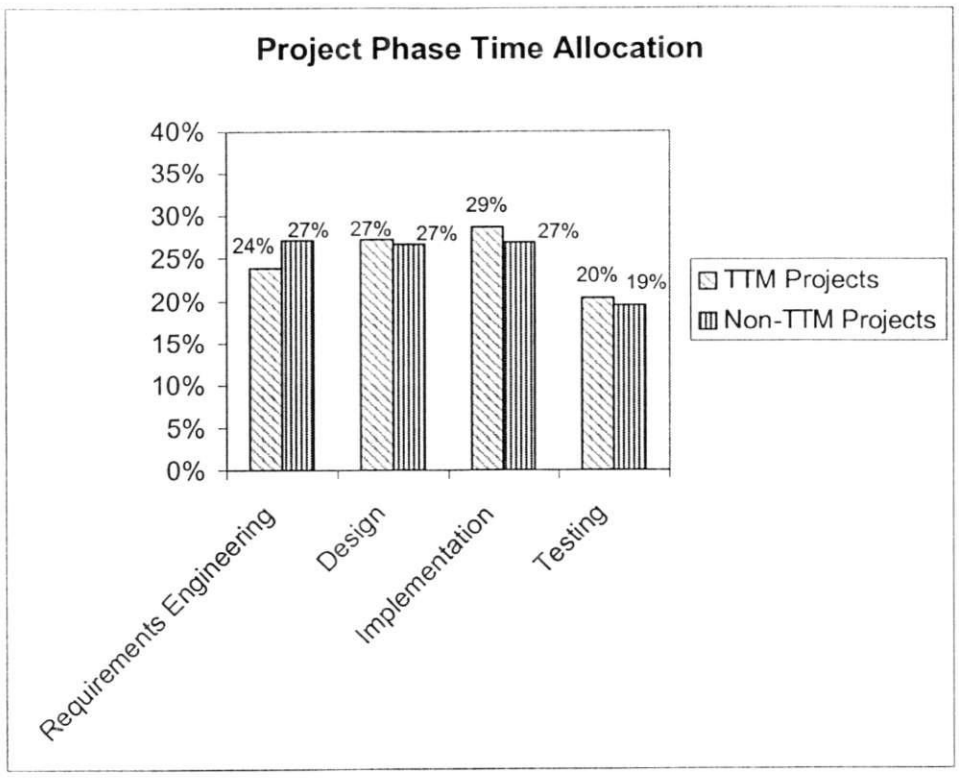

Figure 15 Appropriate Amount of Time per Project Phase

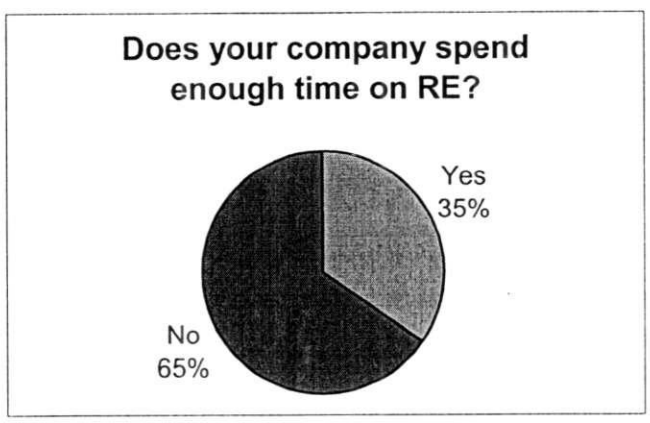

Figure 16 Sufficiency of Time Currently Spent on RE

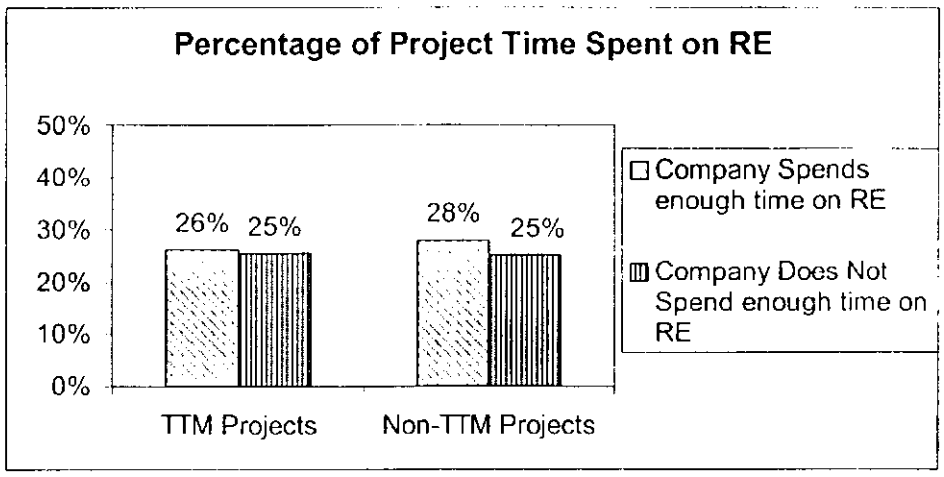

Figure 17 Percentage of Time that Should be Spent on RE

4.3.6.2 Importance of Requirements Engineering Analysis and Results

Although respondents generally thought that appropriate project effort per phase for TTM and non-TTM projects was fairly similar, the biggest difference was in the RE phase. This suggests that people are not sufficiently aware of the problems that incomplete RE can lead to in the overall life cycle of a project. By assuming that just because a project is TTM, one can skimp on the RE phase in order to complete the project, they are missing the concept of the life cycle cost. A TTM project is only ultimately successful if it is built on a solid foundation. Respondents also indicated that whether or not they felt that their company spends enough time on RE, effort allocated to RE should be approximately 25% of the total project effort. One slight difference is found when dealing with non-TTM projects. For non-TTM projects, respondents who felt that their company spends enough time in RE actually identified higher RE effort allocation than those who felt their company did not spend enough effort in RE. This suggests that when an individual's company does not spend enough time in RE, he underestimates the appropriate amount RE effort required.

The majority of respondents (two-thirds) do not believe that enough time is spent on RE. In general, respondents recognize that there is a lack of effort spent on

hashing out requirements details. Although this is recognized, it is difficult to determine how RE effort should be spent. Obviously, in the waterfall method, one could specify that the entire RE should be performed (use up all allocated time for RE) before starting on the implementation. However, the requirements often need to be amended after the users have had a chance to solidify their needs by examining the technical feasibilities and possibilities. Therefore, upfront RE effort is best spent on determining the core, unalterable set of requirements with time set-aside at latter points in the project to revisit the requirements.

4.4 Conclusions and Key Findings

1. There exists a general lack of knowledge regarding available RE techniques.

   Of the 25 techniques surveyed, only one-third of the techniques got a score of over 50%. This suggests that for two-thirds of the techniques examined, the sample group of respondents (on average) had little or no familiarity with the technique.

2. RE Technique usefulness is largely determined by familiarity.

   For both TTM and Non-TTM projects, technique familiarity was positively correlated with perceived RE technique usefulness to within 99.9% confidence limits.

3. Individuals who worked primarily on schedule-driven projects generally have less familiarity with RE techniques than do individuals who work primarily on budget or scope/quality-driven projects.

   On average, respondents who worked primarily on schedule-driven projects reported an overall technique familiarity of 41% as compared to 54% for respondents who primarily work on budget or scope/quality-driven projects.

4. Good communication is a key factor in successful RE.

In response to all of open-ended questions relating to personal preferences in RE, the concept of good communication was highlighted as an important factor.

5. It is important that requirements be unambiguous.

   Of the seven attributes of good requirements, unambiguity was cited as the most important attribute for TTM projects, and the second most important factor for Non-TTM projects.

6. The perceived usefulness of RE techniques for TTM projects is correlated with perceived usefulness for Non-TTM projects but is not correlated with theoretical usefulness of RE techniques on TTM projects.

   Based on rank correlation, the perceived usefulness of an RE technique for a TTM project is positively correlated with the perceived usefulness of the same RE technique for a Non-TTM project, but is not correlated (to within 95% confidence limits) with the theoretical suitability of RE techniques to TTM projects.

7. RE techniques must be easy to use to be used by Requirements Analysts on TTM projects.

   After unambiguity, ease of use was cited as the second most important attribute of a requirement on a TTM project. Ease of use was also the most common response to the question regarding important considerations in choosing an RE technique.

8. Approximately 25% project time should be spent in RE, although majority of respondents do not feel enough time is allocated to RE in their companies.

   On average, respondents felt that 24% of overall project effort should be spent on RE for TTM projects, versus 27% of overall project effort for Non-TTM projects. Only one-third of respondents, however, reported that a sufficient amount of RE was performed by their company.

5 CHAPTER FIVE: CONCLUSION

Objectives

- To summarize the results of the literature survey, analysis of literature, and empirical research.
- To summarize the results of the analysis of the Research data.
- To present conclusions regarding RE in TTM projects.
- To discuss further areas of research pertaining to RE in TTM projects.

5.1 Key Findings of Literature Survey

5.1.1 RE Process

The RE process consists of four phases that must be addressed iteratively if a requirements specification of reasonable quality is to be produced. As identified by Sommerville [1998], those phases are Elicitation, Analysis, Specification, and Validation. By iterating through these phases, the Requirements Analyst gradually discovers the requirements of the system and works to refine the complete set of necessary requirements. Explicitly addressing the separate phases of the RE process, however, is not sufficient to adequately determine the system requirements. Lack of communication and an inability to determine core requirements from supplementary requirements have been identified as common deficiencies in the RE process [Macaulay 1996, Standish Group 1998, McConnell 1995].

5.1.2 TTM Projects

TTM projects are those for which schedule is the driving factor. Although many RE tools and techniques exist, many were developed with the assumption that a stable set of requirements exist, and the Requirements Analyst must simply determine and document them [Costello 1995]. With the changing nature of today's technology, it may be impossible to determine the complete set of requirements of an organization [Truex 1999]. The profitability of TTM projects depends on the ability to quickly determine and satisfy the needs of a particular market niche.

One difficulty with TTM projects is in determining the appropriate level of RE that must be performed in order to construct an adequate requirements document. Although there are a number of different RE techniques to choose from, McPhee [2000] defines a core set of requirements that are appropriate for use on any RE project.

5.1.3 RAD Methodologies

In the context of the literature survey, RAD is a development methodology which assists system developers in decreasing overall development time and/or effort. Card [1995] suggests that there are three fundamental ways of reducing time required to perform a task.:

1. Perform fewer tasks
2. Perform tasks more quickly
3. Perform tasks concurrently

The iterative and spiral methodologies are often considered to be appropriate RAD methodologies as they assist the developer in adapting to changing conditions.

5.1.4 RE Techniques

A survey of existing RE tools and techniques found 26 techniques that are used in industry and/or academia. A general description of each of the techniques was documented, and the techniques were classified as to their applicability to the RE phases (elicitation, analysis, specification, validation). Each technique was rated in terms of its applicability to a TTM project by applying each technique against Card's [1995] methods for reducing schedule time required for a project.

5.1.5 Additional RE Technique Selection Methodologies

Two other RE technique methodologies are available to assist Requirements Analysts in determining appropriate RE techniques. Bickerton [1992] describes a methodology using organizational factors to choose appropriate techniques, while Maiden [1996] uses social and cognition factors concerned to help determine the most appropriate technique.

Both methodologies have their merits, but they do not focus specifically on projects for which schedule is the main project driver. Bickerton's method provides only a broad description of how to evaluate organizational factors, and the majority of techniques fall into one category. Maiden's methodology provides a choice of only

12 techniques, all being focused on cognitive mapping techniques such as brainstorming.

5.2 Summary of Empirical Research Results and Analysis

A survey was developed and administered to members of the software development community, and the following conclusions were reached following an analysis of the data provided by the respondents:

1. There exists a general lack of knowledge regarding available RE techniques.
2. RE Technique usefulness is largely determined by familiarity.
3. Individuals who worked primarily on schedule-driven projects generally have less familiarity with RE techniques than do individuals who work primarily on budget or scope/quality-driven projects.
4. Good communication is a key factor in successful RE.
5. It is important that requirements be unambiguous.
6. The perceived usefulness of RE techniques for TTM projects is correlated with perceived usefulness for Non-TTM projects but is not correlated with theoretical usefulness of RE techniques on TTM projects.
7. RE techniques must be easy to use to be used by Requirements Analysts on TTM projects.
8. Approximately 25% project time should be spent in RE, although majority of respondents do not feel enough time is allocated to RE in their companies.

5.3 Conclusions Regarding RE for TTM projects

It is both possible and desirable to perform relatively in-depth RE for a TTM project. The main focus when performing RE in TTM projects is to ensure that strong communication is built between the stakeholders, and that everyone involved is made to focus on prioritizing the requirements so that the most important

requirements are satisfied, even if time restraints mean that not all requirements can be satisfied.

When choosing a set of appropriate RE techniques for use on a project, the Requirements Analyst must evaluate the techniques based on how well they facilitate completing both the RE phase quickly as well as adequately contributing to the reduction in schedule time throughout the lifecycle of the product. Indeed, the project should be reevaluated to ensure that it is a TTM project. If the schedule is not a driving factor, the Requirements Analyst must determine if it is appropriate to go to great lengths to shorten the schedule time required to complete the project.

5.4 Areas of Future Research

There are a number of areas where the research presented in this thesis could be expanded:

1. Develop a practical guide to Requirements Engineering for software developers to use on time-to-market projects.
2. Evaluate appropriate RE techniques for use when one of the other project drivers is most important (Budget or Scope/Quality).
3. Evaluate lifecycle methodologies appropriate for TTM projects rather than focusing solely on RE techniques.
4. Perform controlled experiments to evaluate the actual effectiveness of the RE techniques that were found to be theoretically suitable for TTM projects.

6 DEFINITIONS, ABBREVIATIONS AND ACRONYMS

6.1 Definitions

Table 12 Definitions

Term	Definition
Analyst	An individual whose role is to examine the system in terms of requirements and implementation and to determine the relationships between different facets of the system.
Complete	Having all necessary parts or elements.
Consistent	Free from variation or contradiction.
Domain Expert	A domain expert has detailed knowledge of the problem and/or solution domain.
E-Commerce	A product that uses electronic communication as the principle means of facilitating business transactions.
Functional Requirement	A functional requirement is a requirement of a system that in place to meet a specific stake-holder need.
Method	A method is a way, technique, or process for doing something.
Model	A model is a simplification of the system and is developed in order that the system, or a specific portion of the system, and its behavior might be better understood.
Modifiable	May be changed to give a different or clarified meaning.
Non-Functional Requirement	A non-functional requirement is a requirement imposed by the environment in which the system is to exist.
Rapid Application Development	Any development methodology or activity whose overall impetus is to increase speed of application development over that of the traditional waterfall development life cycle.

Term	Definition
Requirement	According to the IEEE Standard 610, a requirement is: 1. A condition or capacity needed by a user to solve a problem or achieve a certain objective. 2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. 3. A documented representation of a condition or capability as in 1 or 2.
Requirements Analysis	The process of analyzing requirements in terms of their impact and interaction with other requirements or with the system itself.
Requirements Analyst	An individual responsible for performing Requirements Engineering.
Requirements Engineering	The process of eliciting, analyzing, specifying, and validating system requirements.
Requirements Elicitation	The process of determining stake-holder requirements.
Requirements Specification	The process of specifying and documenting a set of requirements.
Requirements Validation	The process of determining that the set of requirements is correct for a proposed system.
Short-Term Project	A project whose life cycle is measured in terms of weeks or months (less than 6).
Specialist	An individual with specialized knowledge or skills in a particular area.
Switching Costs	The costs associated with switching from one vendor or supplier to another.

Term	Definition
System	The artifact to be developed in accordance with the stake-holder requirements.
System Designer	An individual responsible for the design, or a portion of the design, of the system.
System Stakeholders	Individuals, or groups of individuals, who will be directly or indirectly impacted by the system.
System Users	Individuals who will us the system in some manner.
Technique	A method of accomplishing a desired aim.
Time-to-Market	A project in which schedule is the most critical factor because reaching market in a specified time frame is essential to achieving profitability goals.
Traceable	Connectable to various system artifacts.
Unambiguous	Clear and precise.
Usable	Convenient and practicable for use in deriving, and determining compliance of, system artifacts.
Verifiable	Capable of being established as truthful or accurate.
Waterfall Life Cycle	A software life cycle in which the development phases (Requirements Engineering, Design, Implementation, and Testing) are distinct phases, and a particular phase does not start until its previous phases has been totally completed.

6.2 Abbreviations and Acronyms

Table 13 Abbreviations and Acronyms

Abbreviation or Acronym	Elaboration
RAD	Rapid Application Development

RE	Requirements Engineering
SSM	Soft-Systems Methodology
S/W	Software
TTM	Time-to-Market

7 REFERENCES

[Akao 1990]	Akao, Y. (ed), **Quality Function Deployment**. Productivity Press, Cambridge M.A. 1990.
[American 1996]	**The American Heritage Dictionary of the English Language**. Houghton Miffin Company.
[August 1991]	August, J.H. **Joint Application Design: The Group Session Approach to System Design**. Englewood Cliffs, NJ: Yourdon Press. 1991.
[Beyer 1995]	Beyer, H.R., Holtzblatt, K **Apprenticing with the Customer**, Communications of the ACM 38(5), pp.45-53. 1995.
[Bickerton 1992]	Bickerton, M., Siddiqi, J. **The Classification of Requirements Engineering Methods**. IEEE International Symposium on Requirements Engineering, pp. 182-186. 1992.
[Boehm 1988]	Boehm, B.W. **Software Engineering Economics**, Prentice-Hall, Englewood Cliffs, New Jersey. 1988.
[Booch 1998]	Booch, G., Jacobson, I., Rumbauch, J. **The Unified Modeling Language User Guide**, Addison-Wesley. 1998.
[Brooks 1995]	Brooks, F.P. **The Mythical Man Month**. Addison Wesley Longman, Inc. 1995

[Buren 1998]	Buren, J.V., Cook, D.A., **Experiences in the Adoption of Requirements Engineering Technologies**, Crosstalk: The Journal of Defense Software Engineering. December. 1998.
[Card 1995]	Card, D.N., **The RAD Fad: Is Timing Really Everything?**, IEEE Software, September 1995, pp.20-22. 1995.
[Checkland 1990]	Checkland, P.B., Scholes, J., **Soft Systems Methodology in Action**. Chichester: John Wiley and Sons. 1990.
[Costello 1995]	Costello, R. J., and Liu, D., **Metrics for Requirements Engineering**, J. Systems Software, 1995, 29:39-63.
[Cumo 1994]	Cuomo, D.L. & Bowen, C.D. **Understanding the usability issues addressed by three user-system interface evaluation techniques**. Interacting with Computers, Vol. 6, No. 1, pp.86-108. 1994.
[Curtis 1998]	Curtis, B., Krasner, H. and Iscoe, N. **A Field Study of the Software Design Process for Large Systems.** Communications of the ACM. 31(11), 1268-1287. 1988.
[Dano 1997]	Dano, B., Briand, H., Barbier, F., **A Use Case Driven Requirements Engineering Process**, Requirements Engineering Journal, Volume 2, Number 2, pp. 79-91. 1997.

[Dagwell 1983]	Dagwell, R. and Weber, R., **System Designers' User Models: A Comparative Study and Methodological Critique,** Communications of the ACM, Volume 26, No. 11, November 1983, pp. 987-997.
[Davis 1993]	Davis, A. M. Weidner, Marilyn D. **Software Requirements: Objects, Functions, and States**. Prentice Hall. 1993.
[Drury 1999]	Drury, D.H., Farhoomand, A., **Information Technology Push/Pull Reactions**, The Journal of Systems and Software. 47, pp.3-10. 1999.
[Gasson 1995]	Gasson, S. **The Reality of User-Centered Design**, Journal of End User Computing, 1999, Issue 4, pp. 3-13.
[Goguen 1997]	Goguen, J.A., **Formal Methods: Promises and Problems**, IEEE Software, Jan/Feb, pp.73-83. 1997.
[Harlson 1997]	Harlson, J. Ryan, K. **A Cost-Value Approach to Prioritizing Requirements**, IEEE Software, September/October, pp.68-88. 1997.
[Hartman 2000]	Hartman, F. **Don't Park Your Brain at the Door : A Practical Guide to Improving Shareholder Value with Smart Management**. Project Management Institute Publishing, 2000.
[IEEE 1984]	**IEEE Guide to Software Requirements Specifications**, IEEE Std 830-1984. IEEE Inc., 345 East 47th St, New York, NY 10017, USA. 1984.

[Jarke 1999]	Jarke, M., Bui, X.T., Carroll, J.M., **Scenario Management: An Interdisciplinary Approach**, Requirements Engineering Journal. 1999.
[Knight 1989]	Knight, K. (ed), **Participation in Systems Development**, G.P. Publishing. 1989.
[Lam 1997]	Lam, W., McDermid, J.A., Vickers, A.J., **Ten Steps Towards Systematic Requirements Reuse**, Requirements Engineering Journal, Volume 2, Number 2, pp.102-113. 1997.
[Lavazza 2000]	Lavazza, L. **Enhancing Requirements and Change Management through Process Modeling and Measurement**, Proceedings of the 4th International Conference on Requirements Engineering. 2000.
[Macaulay 1996]	Macaulay, L. A. **Requirements Engineering**. Springer-Verlag. 1996.
[Maiden 1996]	Maiden, N., Rugg, G., **ACRE: Selecting Methods For Requirements Acquisition**, Software Engineering Journal 11(3), pp. 183-192. 1996
[McConnell 1998]	McConnell, S. **Software Project Survival Guide**. Microsoft Press. 1998.
[McConnell 1996]	McConnell, S. **Rapid Development: Taming Wild Software Schedules**. Microsoft Press. 1996

[McPhee 2000]	McPhee, C., andEberlein, A. **An Approach to Evaluating Requirements Engineering Methods for Applicability to time-to-market Projects**, Proceedings of the 13th International Conference on Software & Systems Engineering and their Applications (ICCSEA2000), Paris, France.
[Morris 1993]	Morris, D. Tamm, B. **Concise Encyclopedia of Software Engineering**. Pergamon Press. 1993
[Mumford 1986]	Mumford, E., **Effective Systems Design & Requirements Analysis: The ETHICS Approach**. MacMillan. 1986.
[Nikula 2000]	Nikula, U., Sajaniemi, J., Kalviainen, H., **A State-of-the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises**. Telecom Business Research Center Lappeenranta Technical Report, Lappeenranta, Finland. 2000
[Olsen 1995]	Olsen, N.C., **Survival of the Fastest: Improving Service Velocity**, IEEE Software, September 1995, pp. 28-38.
[Paulk 1993]	Paulk, M. et al**, Key Practices of the Capability Maturity Model, Version 1.1**, CMU/SEI Technical Report. 1993.
[Preece 1994]	Preece, J., Rogers, Y., Sharp, J., Benyon, D., Holland, S. & Carey, T. **Human-Computer Interaction**. Addison-Wesley. Wokingham. UK. 1994.

[Pressman 1997]	Pressman, R.S. **Managing Change for Rapid Development**, IEEE Software, March/April, pp.120-122. 1997
[Rosenberg 1998]	Rosenberg, L.H., Hammer, T.F, Huffman, L.L, **Requirements, Metrics, and Testing**, 15th Annual Pacific Northwest Software Quality Conference. 1998.
[Royce 1987]	Royce, W. W., **Managing the development of large software systems: Concepts and techniques**, Ninth IEEE International Conference on Software Engineering, IEEE Computer Society Press, Los Alamitos, California, pp. 328-338. 1987.
[Reubenstein 1991]	Reubenstein, H. B., Waters, R. C. **The Requirements Apprentice: Automated Assistance for Requirements Acquisition,** IEEE Transactions on Software Engineering, vol. 17 No. 3, pp. 226-240. 1991.
[Rosenberg 1998]	Rosenberg, L.H., Hammer, T.F, Huffman, L.L., **Requirements, Metrics, and Testing**, 15th Annual Pacific Northwest Software Quality Conference. 1998.
[Rothman 2000]	Rothman, J., **The 4 R's of Software Process Improvement**, Crosstalk, May, pp.26-28. 2000.
[Schull 2000]	Shull, F., Rus, I., Basili, V., **How Perspective-Based Reading Can Improve Requirements Inspections,** IEEE Computer Magazine, Vol.33, No.7, July. 2000.

[Shapiro 1998]	Shapiro, C., Varian, H.R., **Information Rules: a Strategic Guide to Network Economy**, Harvard Business School Press. Wiley. 1998.
[Sommerville 1998]	Sommerville, I., Kotonya, G., **Requirements Engineering: Processes and Techniques**. John Wiley & Son Ltd. 1998.
[Sommerville 1997]	Sommerville, I., Sawyer, P., Sommerville, A.. **Requirements Engineering: A Good Practice Guide** John Wiley & Son Ltd. 1997
[Standish Group 1998]	The Standish Group, **CHAOS (Application Project and Failure)**. standishgroup.com. 1998.
[Stark 1998]	Stark, G., Skillicorn, A., Ameele, R., **An Examination of the Effects of Requirements Changes on Software Releases**, Crosstalk: The Journal of Defense Software Engineering. December. 1998.
[Templeton 1994]	Templeton, J.F., **The Focus Group : A Strategic Guide to Organizing**, Conducting and Analyzing the Focus Group Interview, Probus Publishing Co. 1994
[Truex 1999]	Truex, D.P. Baskerville, R. Klein, H., **Growing Systems in Emergent Organizations**, Communications of the ACM (August 1999/Vol. 42., No. 8), pp. 117-123. 1999.
[Vitalari 1983]	Vitalari, N. and Dickenson, G., **Problem Solving for Effective System Analysis: An Experimental Exploration**. Communications of the ACM, Vol. 26, No. 11, November, 1983 pp. 948-956.

[Wells 2000]	Wells, J.D. www.extremeprogramming.org, **The Rules and Practices of Extreme Programming**, 2000.
[Wiegers 1999]	Wiegers, K.E. **First Things First: Prioritizing Requirements**, Software Development Magazine, September. 1999.
[Yourdon 2000]	Yourdon, E. **Modern Structured Analysis**, Yourdon Press. 2000.
[Zhong 1998]	Zhong, T, Liu, L.Y., Li, J., Chung, J., Guttemukkala, V., **Business-to-Business E-Commerce with Open Buying on the Internet**, Proceedings of the International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems., IEEE Computer Society Press. 1998.

APPENDIX A: RE FOR TTM SURVEY

1. Work Experience

Question 1.1:

What is your current job?
(If several answers apply, please select your main job responsibility.)

- ○ Software Developer
- ○ Software Tester
- ○ Software Project Manager
- ○ Other [ ]

Question 1.2:

How many people work in software development in your company?

[ ]

Question 1.3:

What is your company's main business?

[ ]

Question 1.4:

What percentage of your career have you spent working on projects in the following schedule length categories:

NOTE: Schedule length is defined as the time period between starting Requirements Elicitation to delivering the product (beginning of maintenance phase).

	Percentage of Career Experience
2 weeks or less	[ ] %
Between 2 weeks and 1 month	[ ] %
1 to 3 months	[ ] %
3 to 6 months	[ ] %
Between 6 months and 1 year	[ ] %
1 to 2 years	[ ] %
2 to 4 years	[ ] %

Total	100%

Question 1.5: What percentage of your career have you spent working on Time-to-Market (TTM) projects?

[ ] %

Question 1.6: Rate your experience level in the following software development activities:
(1 = no experience to 5 = very experienced)

	Experience Level
Requirements Engineering	
Design	
Implementation	
Testing	
Maintenance	

---

2. Project Success Factors

Question 2.1: How do you know when a project is done?

Question 2.2: How do you determine if the project has been successful?

Question 2.3: Rank the following factors in order of their influence on decision making for the types of projects that you **commonly** work on.

	Rank
Schedule	
Budget	
Quality	

---

3. Attributes of 'Good' Requirements

Question 3.1: What is the importance of the listed characteristics of requirements?
(1 = not important to 5 = extremely important)

	TTM Projects	Non-TTM Projects
Unambiguous		
Complete		
Verifiable		
Consistent		
Modifiable		
Traceable		
Usable		

4. Requirements Engineering Techniques

Question 4.1:

Indicate your familiarity with, and usefulness of, the following Requirements Engineering methods and techniques.
Some methods and techniques overlap one another, so please indicate your familiarity with, and usefulness of, the technique as described on the RE Techniques page.
If you are not familiar with a technique, you do not need to indicate its usefulness.
(1 = not familiar to 5 = very familiar)
(1 = not useful to 5 = very useful)

	Familiarity	Usefulness (TTM projects)	Usefulness (Non-TTM projects)
Interviews			
Data Mining			
Joint Application Design (JAD)			
Quality Function Deployment (QFD)			
Cooperative Requirements Capture			
Designer as Apprentice			
Observation and Social Analysis			
Informal Modeling (text, rich pictures)			

Semi-formal Modeling (DFD, state-charts, UML)	☐	☐	☐
Formal Modeling (Z, VDM, SDL)	☐	☐	☐
Scenarios / Use Cases	☐	☐	☐
Focus Groups	☐	☐	☐
Future Workshops	☐	☐	☐
Soft Systems Methodology (SSM)	☐	☐	☐
ETHICS	☐	☐	☐
User-Centered Design (UCD)	☐	☐	☐
Throw-away Prototyping	☐	☐	☐
Evolutionary Prototyping	☐	☐	☐
Requirements Reuse	☐	☐	☐
Traceability	☐	☐	☐
Viewpoint-oriented Techniques	☐	☐	☐
Requirements Checklists	☐	☐	☐
Requirements Prioritization	☐	☐	☐
Requirements Testing	☐	☐	☐
Requirements Reviews	☐	☐	☐
Requirements Change Management	☐	☐	☐

Question 4.2: Have you ever explicitly chosen to use a particular RE technique on a project?

○ Yes ○ No

If Yes, what made you chose that technique over another technique?

Question 4.3: In general terms, what makes a particular RE technique useful?

5. Personal Preferences regarding Requirements Engineering

Question 5.1: In general, do you enjoy eliciting and analyzing requirements?

Yes No

Question 5.2: What do you like most about eliciting and analyzing requirements?

Question 5.3: What do you like least about eliciting and analyzing requirements?

Question 5.4: Are certain members of your team better at performing RE than others?

Yes No

If yes, what makes some people better than others at performing RE?

6. Importance of Requirements Engineering

Question 6.1: In your opinion, does your company/project do enough RE?

○ Yes ○ No

Question 6.2: In your opinion, what percentage of effort **should** be allotted to the following activities:
(If you have found that these activities overlap, please indicate approximate values for projects that you **commonly** work on.)

	TTM Projects	Non-TTM Projects
Requirements Engineering	%	%
Design	%	%
Implementation	%	%
Testing	%	%
Total	**100%**	**100%**

7. Submit Questionnaire

Contact Information: How did you hear about this questionnaire?

○ SENG List Server
○ SRE List Server
○ Contacted by phone
○ News Group
○ Internet Search
○ Other

Your answers to this questionnaire will remain completely confidential, but if you would like to be informed as to the progress and results of this research, please fill-in the following information:

Name:
E-mail:
Phone (include country code):

Would you like to receive updates on the progress of this research?

○ Yes ○ No

May we contact you to further discuss your answers to this questionnaire? ☐ Yes ☐ No

APPENDIX B: PUBLICATIONS

McPhee, C. and Eberlein, A. (2000) **An Approach to Evaluating Requirements Engineering Methods for Applicability to time-to-market Projects**, Proceedings of the 13th International Conference on Software & Systems Engineering and their Applications (ICCSEA2000), Paris, France.

McPhee C. and Eberlein A. (2001) **Requirements Engineering for Time-to-Market Projects**, submitted to the 9th Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS 2002), Lund, Sweden