5-11-2002

Definition and Representation of Requirement Engineering/ Management : A Process-Oriented Approach

Judy-Audrey-Chui-Yik Liaw

DEFINITION AND REPRESENTATION OF REQUIREMENT ENGINEERING / MANAGEMENT: A PROCESS-ORIENTED APPROACH

By

Judy-Audrey-Chui-Yik Liaw

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
For the Degree of Master of Science
in Industrial Engineering
in the Department of Industrial Engineering

Mississippi State, Mississippi

May 11, 2002

DEFINITION AND REPRESENTATION OF REQUIREMENT ENGINEERING /

MANAGEMENT: A PROCESS-ORIENTED APPROACH

By

Judy-Audrey-Chui-Yik Liaw

Approved:

____
Allen G. Greenwood
Associate Professor of Industrial Engineering
(Director of Thesis)

____
A. Wayne Bennett
Dean of College of Engineering

____
Stanley F. Bullington
Professor of Industrial Engineering
(Graduate Coordinator)

____
William N. Smyer
Associate Professor of Industrial Engineering
(Committee Member)

Name: Judy-Audrey-Chui-Yik Liaw

Date of Degree: May 11, 2002

Institution: Mississippi State University

Major Field: Industrial Engineering

Major Professor: Dr. Allen G. Greenwood

Title of Study: DEFINITION AND REPRESENTATION OF REQUIREMENT ENGINEERING/MANAGEMENT: A PROCESS-ORIENTED APPROACH

Pages in Study: 101

Candidate for Degree of Master of Science

Requirements are important in software development, product development, projects, processes, and systems. However, a review of the requirements literature indicates several problems. First, there is confusion between the terms 'requirements engineering' and 'requirements management.' Similarities and/or differences between the two terms are resolved through a literature review; resulting in comprehensive definitions of each term. Second, current literature recognizes the importance of requirements but offers few methodologies or solutions for defining and managing requirements. Hence, a flexible methodology or framework is provided for defining and managing requirements. Third, requirements methodologies are represented in various ways, each with their respective strengths and weaknesses. A tabular view and hybrid graphical view for representing the requirements process are provided.

(115 words)

DEDICATION

I would like to dedicate this research to my parents, John and Alice Liaw, my sister Jessie, my best friend Yu Loong, and my American family – Auntie Dianne Enis, Rose Wells, Jennifer and Thomas Kihlken, and Rebecca Mayo.

ACKNOWLEDGEMENTS

I express my sincere gratitude to the many people without whose selfless assistance this thesis could not have materialized. First of all, sincere thanks are due to Dr. Allen G. Greenwood, my thesis director and major professor, for spending so much time and effort to guide and assist me throughout the thesis process. Expressed appreciation is also due to Dr. Stanley F. Bullington and Dr. William N. Smyer for serving on my committee. Finally, I would like to thank Dr. Larry G. Brown for providing the opportunity to pursue a Master's degree at Mississippi State University.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The word ‘requirement’ is used commonly in everyday life. When I chose a university to apply to, one of the requirements was that the tuition must be less than $15,000 annually. Another requirement that I had was that the university must have a good engineering school, at least ABET accredited. On the other hand, Mississippi State University has a list of requirements that the applicants must meet before being accepted into MSU. For instance, international students must achieve at least a specific TOEFL score. However, requirements are much more than just a checklist to be checked off. (Prior to this research, I was unaware of the vast application and importance of requirements.)

Definition of Requirements

A review of the literature indicates that there are many definitions for the term ‘requirement.’ All of the definitions found in the literature are shown in Table 1. The order that the definitions appear is arranged from narrow to broad view.

Table 1
Definitions of requirements

Source	Definition	Comments
Kulak and Guiney [17]	"A *requirement* is something that a computer application must do for its users" (p.4).	Only covers software development
Dorfman and Thayer, quoted by Leffingwell and Widrig [18],[22]	"A software capability needed by the user to solve a problem to achieve an objective. A software capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documentation" (p.15).	Only covers software development
Robertson and Robertson [27]	"A requirement is something that the product must do or a quality that the product must have" (p.5).	Only covers product development
Hooks and Farry [12]	"Good requirements – defining the job that needs to be done or the characteristics of the product we want to buy, develop, build, modify, or have developed, built, or modified – are essential to improved productivity" (p.xxiii).	Requirements define what needs to be done or what is desired in product development
IEEE Std 1220-1998 [31]	"A statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines)" (p.8).	Requirements are necessary for acceptance of a product or process
Leffingwell and Widrig [18]	"Requirements define capabilities that the systems must deliver, and conformance or lack of conformance to a set of requirements often determines the success or failure of projects" (p.16).	Project success depends on how well the requirements are met or not met
Davis and Zweig [5]	"…those externally observable characteristics of a system that a user, buyer, customer, or other stakeholder desires to have present in the system" (p.61).	Only covers externally viewable characteristics in a system
Harwell *et al.* [10]	"[i]f it mandates that something must be accomplished, transformed, produced, or provided, it is a requirement – period" (para.4).	Indicates that requirements are needed for any activity/process

Each definition points out something important about requirements. It is just too bad that all these important elements do not appear in the same definition. Keywords extracted from the definitions include ‘a thing’, capability, users, must do, must have, define or identify, characteristic, customers, observable, and action (accomplished, transformed, produced, provided).

Therefore, a requirement can be defined as an aspect of a system that defines what it must have or must do in order to accomplish a desired outcome for someone (users, customers, stakeholders, etc.). Davis and Zweig’s notion of “externally observable characteristics” is not included because there are some features that are not observable and yet important to the customers. For instance, everyone knows that electricity is important but some people do not know how current flows.

Importance of Requirements

Why are requirements important? A common reason cited by the literature is cost. For example, software companies could have saved themselves a lot of money had they worked out all the bugs in their software packages before shipping them. However, working out all the bugs in the software can potentially take a long time. Hence, most software companies choose to ship an almost-perfect software and only fix problems if they are detected. Besides creating a bad reputation for the software companies, this also means additional cost for them.

Ten Reasons

A review of the literature indicates the importance of requirements. Ten reasons (not in any particular order) why requirements are important are documented in the Table 2.

Table 2
Ten reasons why requirements are important

No.	Reason
1	"[R]equirements are important because if you don't know what you want, or don't communicate what you want, you reduce your chances of getting what you want" (p.1) [8].
2	"Bell Labs and IBM studies have determined that 80 percent of all product defects are inserted in the requirement definition stage of product development, the stage where you should define a product's needs and uses" (p.3) [12].
3	From an information systems standpoint, requirements determination and structuring occurs in the first phase (analysis phase) of the systems development life cycle (SDLC). Errors in the final system are often caused by inadequate efforts in this phase [11].
4	The Standish Group found that projects that were late and under expectations were caused by the following: lack of user input, and incomplete and changing requirements [18].
5	The more time and effort that NASA spent on the requirements definition stage, the less they spent on budget overrun [12].
6	The European Software Process Improvement Training Initiative (ESPITI) reported that major problems in software development fall into two main categories - requirements specification and managing customer requirements [18].
7	"[W]e have grown to care about requirements because we have seen more projects stumble or fail as a result of poor requirements than for any other reason" (p.2) [17].
8	"Bad requirements result in cost overruns, schedule slips, frustrated and overworked employees, unhappy customers, lost profitability, and limited careers" (p.7) [12].
9	Requirements, known as demanded-quality items, are inputs to the House of Quality in Quality Function Deployment [21].
10	Hooks and Farry cited Dean Leffingwell estimation that "requirements errors accounts for 70 to 85 percent of software project rework costs" (p.8). In addition, Barry Boehm found that half of the total budget was used for rework. This means that there is a high probability that the high cost of rework is due to errors in requirements [12].

This list proves that requirements are important in a variety of areas. This list also indicates that the success or failure of software development, product development, projects, processes, or systems depends heavily on the early stages or requirement definition stages. The more time and effort that is spent upfront defining requirements, the less the development team has to spend (in terms of money and time) later to rectify the problems. Leffingwell and Widrig [18] found that costs of fixing problems during maintenance stage of the software development is twenty times the cost of fixing problems during requirements stage.

This list of reasons indirectly points out that something is done “to” the requirements. In the beginning, requirements have to be defined. Once that is done, requirements need to be tracked, indicating some sort of management is required. These definition and management activities are a part of a process, indicating that requirements are either engineered and/or managed.

Areas of Application for Requirements

Upon investigation, it is found that requirements are embedded in several processes, namely systems engineering, software development, and concurrent engineering. The roles of requirements are examined in the following section.

Systems Engineering

Engineering has traditionally focused on individual phases of a product's life cycle. Market competitiveness has since changed the focus to one of viewing the entire cycle (from concept development to disposal) as a whole [3]. This is in fact the essence of systems engineering. The International Council on Systems Engineering (INCOSE) [15] defines systems engineering as:

> "an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem: operations, performance, test, manufacturing, cost and schedule, training and support, and disposal. Systems Engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems Engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs" (para.1).

This definition demonstrates the importance of customer input. These inputs are transformed into customer requirements, which eventually flow through the entire product development process, and even through the life cycle.

Blanchard and Fabrycky [3] provide another point of view on systems engineering shown in Figure 1.

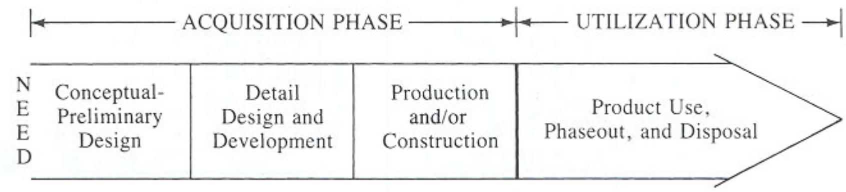

Figure 1: Systems engineering [3]

Systems engineering begins with identifying the need for the system. Once customers' needs are gathered, conceptual design begins. This is where the customer needs are translated into functional requirements. These functional requirements are then passed along to preliminary design where trade-off studies, initial prototyping, etc. are carried out. Detail design and development includes activities such as describing the system design and development, testing, and evaluating prototypes. The system is then analyzed and built in the production and/or construction phase. During the utilization and support phase, the system is assessed, analyzed, and modified, if necessary. The systems engineering cycle ends with a phaseout and disposal of the system. In the past, phaseout and disposal of a product were not considered as the responsibility of the manufacturer.

One way of viewing this is that requirements drive all the other subsequent activities. Blanchard and Fabrycky [3] write that the requirements have "to be well-defined and specified" (p.24). Also, it is important for requirements to be visible throughout the entire process – this is known as traceability.

Software Development

Leffingwell and Widrig [18] said that "[e]ffective requirements management cannot occur without the context of a reasonably well-defined software process…" (p.213). This shows that it is important to examine the activities contained within the software development process. In the past, programmers would write code and only fix "bugs" when they are found. This would repeat until the problems can no longer be fixed. Then Boehm [18] created the stepwise process model, which is made up of several stages: e.g. requirements, design, coding. However, this model has a shortcoming: it is sequential and thus does not allow feedback between stages.

In 1970, Winston Royce [3] developed the "waterfall model," which consists of five to seven steps. The basic steps within this process are requirements, design, coding and unit test, system integration, and operation and maintenance. The main difference between the waterfall model and the stepwise model is that the waterfall model allows feedback at every stage. Other researchers in the software development field criticized this waterfall model, shown in Figure 2, for not addressing the prototyping activity [3]. Even though the waterfall model is popular among software developers, there is a discrepancy between different authors. Blanchard and Fabrycky's [3] representation of the waterfall model is shown in Figure 3.

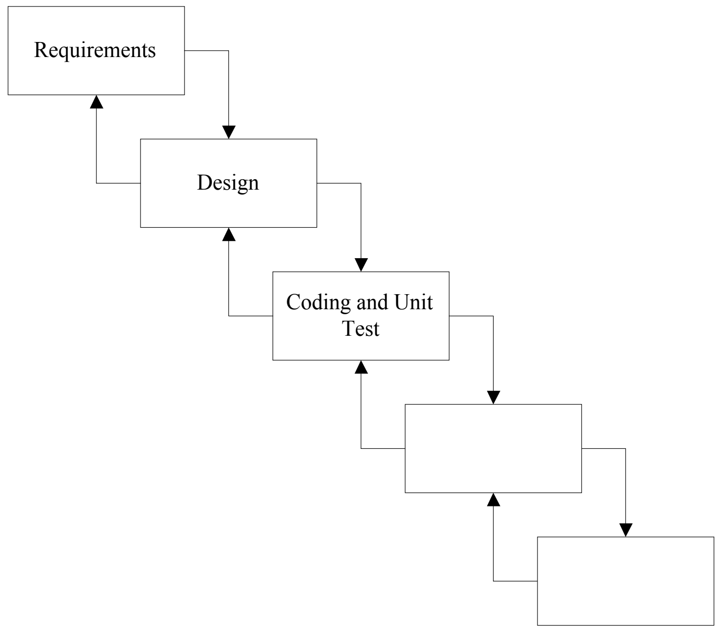

Figure 2: The waterfall model documented in Leffingwell and Widrig's [18] book

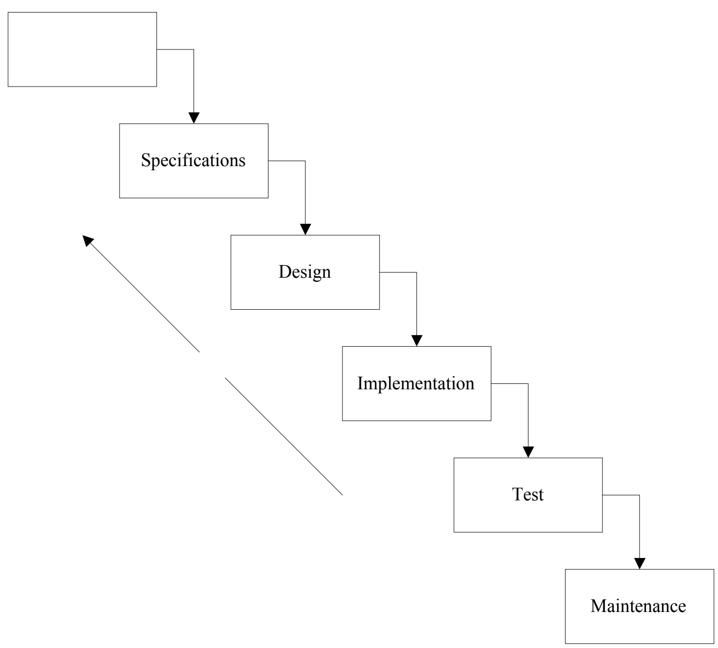

Figure 3: The waterfall model documented in Blanchard and Fabrycky's [3] book

According to Blanchard and Fabrycky [3], the waterfall model is made up of six steps – requirements analysis, specifications, design, implementation, test, and maintenance. Even though the waterfall models presented by both authors are different, one similarity stands out: - both of the models begin with requirements. Again, this supports the notion that something is done onto requirements throughout the entire process.

From the information systems standpoint, there is a similar model called the Systems Development Life Cycle (SDLC) [11]. This model is shown in Figure 4.

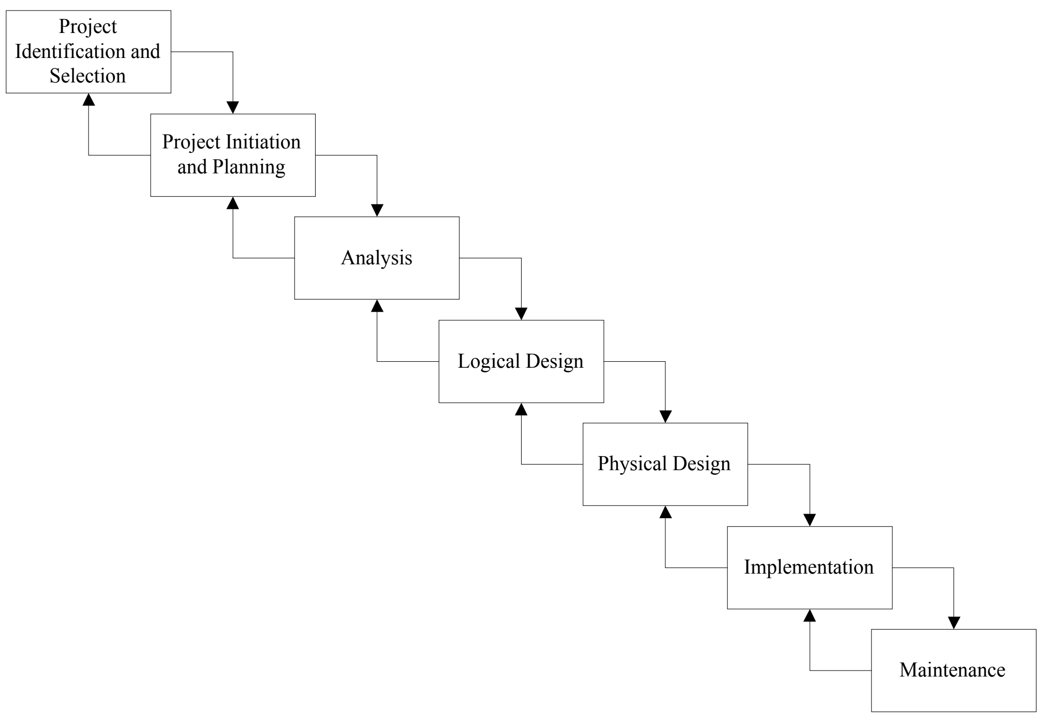

Figure 4: Systems Development Life Cycle (SDLC) [11]

This model is comprised of seven phases, namely project identification and selection, project initiation and planning, analysis, logical design, physical design, implementation, and maintenance. The first phase, project identification and selection, involves identifying the need for the project. This is similar to the first step within the systems engineering process. This is succeeded by the project initiation and planning phase where further investigation is done on the need for the project. If the project is approved, the development team draws up a detailed plan for the project.

Next, the team examines the current system and proposes a new system. This phase, known as the analysis phase, is where the activities related to requirements take place. In order to design the system that the stakeholders desire, the team has to gather the stakeholders' requirements. Then, the team analyzes the current system and decides what needs to be done in order to meet their stakeholders' needs. The team then works on a rough sketch of the proposed system.

The subsequent two stages of the SDLC involve design. The first part of design is the logical design, where all of the functions of the proposed system are specified without the restriction of computer hardware. The logical design is converted into specifications in the physical design phase.

Once the specifications are set, the team turns the specifications into a working system in the implementation phase. Activities included in this phase include coding, testing, and installing the new system. Last but not least, the system is modified periodically in the maintenance phase.

In 1986, Boehm [3] developed the "spiral model" shown in Figure 5. The spiral model, which is read counter clockwise from the center, is based on risk-driven approach. This approach allows each prototype's risks to be evaluated and resolved each cycle before progressing to the next step. The spiral process begins with a need. This need is progressively transformed into the final product through an iterative process. Since Boehm's previous stepwise model was criticized for not including feedback and prototyping, he has included them into this model.

The spiral model is another example where requirements play an important role. For instance, once the need is identified, the system requirements are determined. In addition, each cycle has an activity involving requirements, indicating that requirements 'evolve' throughout the process.

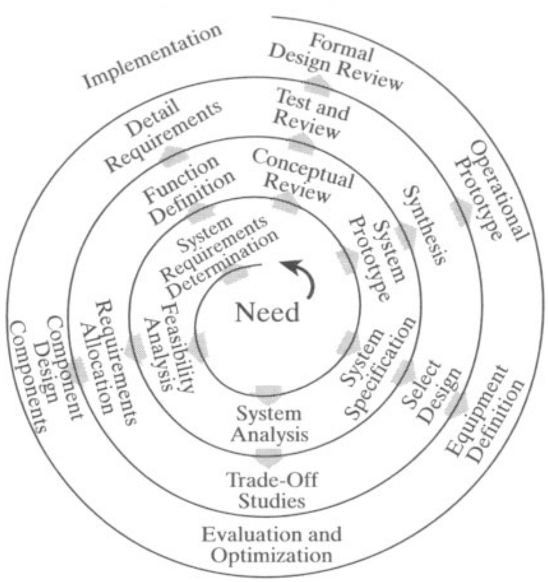

Figure 5: Spiral process model [3]

The spiral model was later succeeded by the "Vee" process model. This process, shown in Figure 6, is created by Forsberg and Mooz [3]. Shaped like the letter 'v', each step is mirrored on the other side by verification to ensure that the goal of each step is achieved. It is no surprise that the "Vee" process begins with defining systems requirements, suggesting the importance of requirements. The next step in the process is to allocate the system functions to subfunctions, followed by designing the components in detail. The next three steps are verifying components, verifying subsystems, and operating and verifying the full system. These three steps fulfill two goals – operation of the final system and ensuring that each step is verified, hence the mirroring effect.

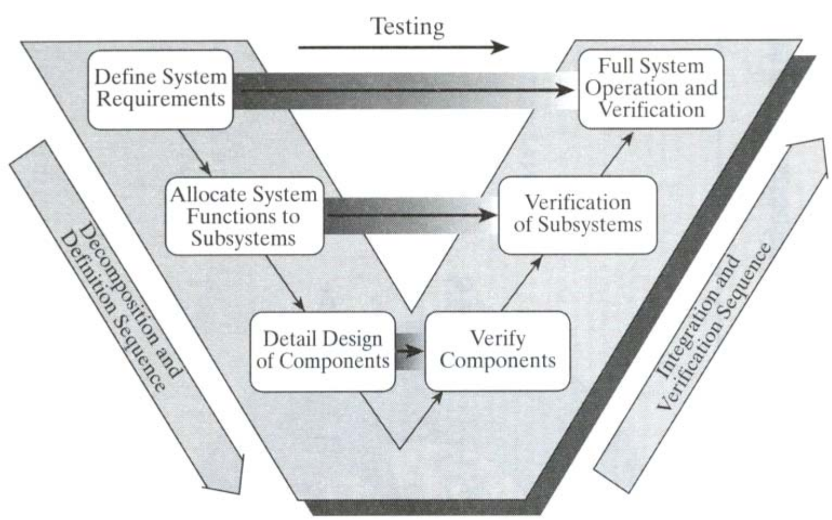

Figure 6: "Vee" process model [3]

The latest model, based on Rational Unified Process (RUP), employ an iterative approach within each phase, including inception, elaboration, construction, and transition [18]. Activities that are carried out during the inception phase include: project scoping, preliminary analysis, scheduling, budgeting, and risk factor estimation. Activities related to requirements are carried out during the elaboration phases. Coding and implementation are performed during the construction phase. The transition phase allows for testing and implementation. Rational Unified Process [24] is discussed further in the next chapter.

One similarity that exists across all models in the software development world is the word ‘requirements’. Every model places some emphasis on defining requirements at the beginning of the process. This indicates that requirements play an important role in each of the alternative processes.

Concurrent Engineering

The Society of Concurrent Product Development (SCPD) [29], formerly known as Society of Concurrent Engineering (SOCE), defines concurrent engineering as a “systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. This approach is intended to cause the developer, from the outset, to consider all elements of the product lifecycle from concept through disposal, including quality control, cost, scheduling and user requirements (Institute for Defense Analyses)” (para.6).

According to Ulrich and Eppinger [36], the generic product development process is composed of planning, concept development, system-level design, detail design, testing and refinement, and production ramp-up. Put simply, product development is like a funnel – it begins with many alternatives and ends with a narrowed alternative through a series of filtration.

The connection between requirements and concurrent engineering can be found in the concept development stage. Activities carried out within this stage include customer needs identification, target specifications, concept generation, concept selection, concept testing, final specifications, project planning, economic analysis, benchmarking, modeling, and prototyping.

Similar to systems engineering and software development, the voice of the customer plays a vital role in concurrent engineering. Customer's needs are collected and translated into design specification, yielding a final product that will satisfy the customers. However, this is much easier said than done. Translating customer needs into design specifications can be quite complicated: one highly acclaimed technique is called Quality Function Deployment (QFD).

Quality Function Deployment (QFD) was first introduced in Japan by Yoji Akao and Katsuyoshi Ishihara [21]. It was successfully applied at a shipyard, specifically Mitsubishi Heavy Industries' Kobe Shipyard, to ensure the production of a high quality ship at every stage of production. Prior to this, quality at every stage has been considered an independent activity. Hence for the first time, quality 'flowed' from the customers needs all the way through the final product.

The most important element in QFD is the House of Quality. This house shows the relationship between customer needs and the product characteristics [19]. Therefore, each engineering decision made (for instance, the size of a nut) can be ultimately traced to one or more customer requirements. However, not much information can be found on the activities that are carried out prior to the House of

Quality. Specifically, it is not clear as to the activity/activities involved in gathering customers' requirements.

The existence of activities related to requirements in all three fields - systems engineering, software development, and concurrent engineering, proves that requirements are widely used. In addition to that, those activities related to requirements are found in the early stages of a process, regardless of the process type. This indicates that requirements do play an important role in shaping the outcome of the process. It also implies that requirements themselves go through a process.

Motivation

A review of the literature indicates the importance of requirements but does not offer many methodologies or solutions for defining and managing requirements. If the literature offers a method for defining requirements, then two main problems surface. First, different requirements methodologies are proposed, suggesting a lack of a standard methodology for requirements for definition and management. Second, the steps within a methodology are usually not well defined. For instance, a step might be to 'develop the vision for the project' but there is no documentation indicating how this might be done or what is required for this to be carried out.

Problem Statement

Based upon a review of the literature, there is confusion between the terms 'requirements engineering' and 'requirements management.' One objective of this research is to investigate the definition of those two terms. Are those two terms interchangeable? If not, what are the differences between 'requirements engineering' and 'requirements management'? In the meantime, this thesis will use both terms as one, i.e. requirements engineering/management.

Secondly, the literature review also shows that different sources suggest different methodologies for defining and managing requirements. This means that there are multiple interpretations of the requirements engineering/management process. Unfortunately, multiple representations only confuse users as to which methodology to use. Therefore, there needs to be one flexible methodology or framework. Users can then apply relevant aspects to meet their needs. The process should to be flexible so that users from different organizations can use the same process by adapting the steps within the process. Users can then add or eliminate steps to fit their need. The importance of making the process customizable is to ensure that the users have a chance to think about issues that may not surface within the proposed process.

Last but not least, the literature review also indicates that there is a problem with representation. Actually, it is not possible to represent the entire process with a single representation method. Again, different sources use different representation methods, as will be discussed later.

Research Objectives

As a result, the following are the objectives for this research.

1. Define requirements engineering and requirements management.

2. Develop a generic process for requirements engineering/management.

3. Develop a process representation scheme.

CHAPTER II

DEVELOPMENT OF A GENERIC REQUIREMENTS ENGINEERING / MANAGEMENT PROCESS

Define requirements engineering and requirements management

In order to achieve the first research objective, a literature review on the terms 'requirements engineering' and 'requirements management' was conducted. This review results in a comprehensive definition of 'requirements engineering' and 'requirements management' respectively.

Requirements engineering (RE) defined

A search on the World Wide Web on the term 'requirements engineering' resulted in more hits on United Kingdom websites. The Requirements Engineering Specialist Group (RESG) of the British Computer Society [26] defines requirements engineering as:

> "[a] key activity in the development of software systems, and is concerned with the identification of the stakeholder goals and their elaboration into precise statements of desired services and behaviour" (para.1).

The definition provided here is oriented towards software development. The phrase "key activity" hints that requirements are vital in software development effort.

The committee of the IEEE Joint International Requirements Engineering Conference [13], to be held September 9 – 13, 2002 in Denmark, defines requirements engineering as:

> "[t]he heart of software development. It is the branch of systems engineering concerned with the real-world goals for, functions of, and constraints on software-intensive systems. It is concerned with how these factors are taken into account during the implementation and maintenance of the system, from software specifications and architectures up to final test cases. RE requires a variety and richness of skills, processes, methods, techniques and tools. In addition, diversity arises from different application domains ranging from business information systems to real-time process control systems, from traditional to web-based systems as well as from the perspective being system families or not" (para.1).

At a glance, this definition is similar to the previous one. However, this definition is more detailed. It specifies that requirements control the entire software development stages. The interesting part is that definition also hints how much work will be required for the requirements engineering effort. A multi-functional team comprised of team members with different skills, knowledge, and background will be required. In addition to that, the team would have to use different tools and techniques.

The recent Symposium on Requirements Engineering [7], held in August 2001 in Toronto, Canada, define requirements engineering (RE) as:

> “[t]he heart of software development. RE is concerned with identifying the purpose of a software system, and the contexts in which it will be used. Hence, RE acts as the bridge between the real world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies. RE is a multi-disciplinary activity drawing on research and experience in software engineering, computer science, business and information systems, human-computer interaction, and social and cognitive sciences. In the 1990’s, significant advances in RE research were made, such as the development of techniques for eliciting and analysing stakeholders’ goals, modelling scenarios that characterise different contexts of use, the use of ethnographic techniques for studying organisations and work settings, and the use of formal methods for analysing safety and security requirements. Despite these advances, RE remains one of the most challenging aspects of software development” (para.1).

This definition points out that requirements is a bridge between people and possible results from the requirements engineering effort. Specifications are also made as to which disciplines are required to be a part of the requirements engineering team. Note that this definition states that RE is still a challenging aspect of software development.

In a paper published in the proceedings of the Second IEEE International Symposium on Requirements Engineering, Bubenko [4] defined requirements engineering as:

> “[t]he area of knowledge concerned with communicating with organisational actors with respect to their visions, intentions, and activities regarding their need for computer support, and developing and maintaining a adequate requirements specification of an information systems" (p.160).

Again, the word “communicating” shows up here too. This is similar to the word “bridge” found in the definition earlier. However, this definition is concerned with only information systems.

Glib [9] define requirements engineering as:

> “[t]he systematic process of determining the complete relevant set of values held by stakeholders, and processing them until a satisfactory level of 'delivery of the required end states' has been made to them. This implies that it must include design, testing, quality control, project management, specification languages and all other relevant disciplines to enable it to succeed" (sec.7).

For the first time, requirements engineering is referred to as a process. It also specifies that any disciplines can be involved – as long as the stakeholders are satisfied.

Zowghi and Offen [38] define requirements engineering to be:

> “…concerned with elucidating real-world goals for the function of, and the constraints on software systems. The major objectives of requirements

> engineering are defining the purpose of a system and capturing its external behavior" (p.247).

Again, the main idea here is on making sure that the goals are achieved on the software systems. This is similar to being a bridge or communicator. At last, someone specified the objective of requirements engineering as defining the system's purpose and external behavior.

In a separate article by the same author, Zowghi [37] expanded on the definition to include activities performed under requirements engineering. According to Zowghi [37]:

> " [t]he major objective of RE is defining the purpose of a proposed system and outlining its external behavior. … RE activities can be divided into five categories:
>
> - requirements elicitation which is the process of exploring, acquiring, and reifying user requirements through discussion with the problem owners, introspection, observation of the existing system, task analysis and so on.
> - requirements modeling where alternative models for the a target composite system are elaborated and a conceptual model of the enterprise as seen by the system's eventual users is produced. This model is meant to capture as much of the semantics of the real world as possible and is used as the foundation for an abstract description of the requirements.

- requirements specification where the various components of the models are precisely described and possibly formalised to act as a basis for contractual purposes between the problem owners and the developers.
- requirements validation where the specifications are evaluated and analysed against correctness properties (such as completeness and consistency), and feasibility properties (such as cost and resources needed).
- requirements management refers to the set of procedures that assists in maintaining the evolution of requirements throughout the development process. These include planning, traceability, impact assessment of changing requirements and so on" (para.1 & 2).

Zowghi indicates that requirements management is indeed a part of requirements engineering. A consultant specializing on requirements engineering, named Ian Alexander [1], explains that requirements engineering include the following activities, "elicitation, analysis of requirements and constraints, modeling of behaviour with scenarios and other techniques, traceability, metrication, review and baselining ... " (para.17). All the activities mentioned by Alexander seem to fit into one of the activities defined by Zowghi. For instance, requirements and constraints analysis probably fall into the requirements validation.

In another article written by Alexander [2], he said that requirements engineering is different from other engineering disciplines. Instead, he asserts that requirements engineering is one “that efficiently and rigorously elicits, organizes, checks, measures, prioritizes and documents what a set of diverse stakeholders want - and helps them to agree on the specification of a solution” (para.7).

Keywords from this definition list of requirements engineering include:

- key activity or heart of software development
- branch of systems engineering
- variety (skills, processes, methods, techniques, tools)
- application diversity
- bridge between people and system
- multi-disciplinary
- communication tool
- systematic process
- define purpose of a system and capture its external behavior, and
- elicit, model, specify, validate, manage

Hence, requirements engineering stems from systems engineering as a bridge between people and system. It is a multi-disciplinary systematic process that elicits, models, specifies, validates, and manages requirements, drawing upon a variety of skills, processes, methods, techniques, and tools.

Requirements management (RM) defined

As for the term ‘requirements management,’ searches on the World Wide Web indicated there are more hits on US-based websites. This suggests that perhaps the term the European countries commonly use is ‘requirements engineering’, while the term Americans commonly use is ‘requirements management’.

Requirements engineering authors Dorfman and Thayer, as quoted in Leffingwell and Widrig [18] and Rational Software’s whitepaper [22], define requirements management as:

> “a systematic approach to eliciting, organizing, and documenting the requirements of the system, and a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system” (p.16).

This definition implies that requirements management is a method for keeping track of requirements changes to ensure that customers and team members are in agreement.

In an article published in a proceeding by the International Council on Systems Engineering (INCOSE), Jones *et al.* [16] quotes from a 1995 article by Stevens and Martin that requirements management is:

> “the identification, derivation, allocation, and control in a consistent, traceable, correlatable, verifiable manner of all the system functions, attributes, interfaces,

> and verification methods that a system must meet including customer, derived (internal), and specialty engineering needs" (sec.2.2).

This definition includes activities that go on within requirements management. Similar to the first definition, Jones *et al.* [16] suggest that requirements management is a systematic method for ensuring that the final result meets the stakeholders' needs.

In another article found on the INCOSE's website, requirements management is said to be made up of capturing, storing, managing, and distributing information [33]. Once again, this indicates that requirements management as management-type activity.

Davis and Zweig [5] defines that requirements management as:

> "the set of activities encompassing the collection, control, analysis, filtering, and documentation of a system's requirements." Requirements management consists of three activities: requirements elicitation (gathering and storing stakeholder needs in a repository), requirements triage (deciding which features to include in the product), and requirements specification (specifying the external behavior of a system to support the features)" (p.61).

Again, this definition specifies the gathering and specifying activities. The new item here is the requirements triage activity.

Lastly, Stevens and Martin [35] from Telelogic, a systems and software developer, said that:

> "Requirements management starts with the definition of requirements and continues through the project, culminating in the acceptance of the product against requirements. … Requirements management could be defined as ensuring:
>
> - we know that the customer wants (quality);
> - the solution efficiently meets these requirements (conformance)" (para.1).

According to Stevens and Martin, requirements management is quite simple – just collect requirements and conform to them.

Several keywords that are associated with requirements management are identify, derive, elicit, collect, store, control, allocate, organize, and document. Therefore, requirements management is a systematic approach for identifying, eliciting, deriving, collecting, organizing, allocating, controlling, and documenting requirements.

Requirements Engineering versus Requirements Management

When the two terms are placed side by side, shown in Table 3, the following key words are observed, suggesting actions performed on requirements. This drives the need for a process view on requirements. Note that similarities are placed at the top of the list.

The International Council on Systems Engineering's (INCOSE) journal, *Insight,* points out the confusion in terms. The editor states that the Requirements Management Working Group members could not agree on the definition of requirements management and requirements engineering. They also could not agree on which one is a subset of the other. The Working Group has since removed the word 'management' from their working group's name [14].

Table 3
Comparison of the term 'requirements engineering' and 'requirements management'

Requirements engineering	**Requirements management**
• systematic • identify • elicit • specify • analyze • translate • model • manage • validate • multi-disciplinary • variety (skills, processes, methods, techniques and tools) • communicate/ bridge • define • develop • maintain • design • test • capture	• systematic • identify • elicit • specify • analyze • derive • collect • allocate • organize • control • document • identify • gather • filter • correlate • verify • information (capture, store, manage, and distribute) • triage

This list indicates two things – first, there is some crossover of activities. This could be due to misuse or misunderstanding of terms. Second, the two terms, requirements engineering and requirements management, are indeed different. It is proposed that requirements engineering and requirements management are separate but related terms. The activities carried out within requirements engineering could be an initial startup for the requirements process. Once that is in place, then the activities within requirements management are carried out. This does not imply that requirements are passed along from requirements engineering to requirements management, but are taken into consideration during the requirements engineering phase. Also, over the course of the product development life cycle, activities would iterate between requirements management and requirements engineering due to the needs for clarification, changing needs, etc. The investigation also implies that the activities performed within requirements engineering are broader than the activities within requirements management. This is indicated by the notion that requirements engineering is a systematic process requiring multi-disciplinary people utilizing a variety of skills, methods, techniques, and tools.

Therefore, it is proposed that requirements engineering is made up of requirements elicitation, requirements modeling, requirements specification, and requirements validation. On the other hand, it is proposed that requirements management is made up of requirements organization, requirements control, and requirements documentation. This provides the basis for further definition. These definitions also serve as a foundation for the next research objective.

In summary, definitions of requirements engineering and requirements management were extracted from the literature. Based on the definitions, a composite definition of requirements engineering and requirements management was developed. However, these definitions illustrate the need for better clarification. A first step to this is to propose components or activities of each term.

Review of Requirements Engineering/Management Activities

The second research objective is to capture all of the activities within the requirements engineering/management process. In this process, the focus is on what the activities are within the process. However, there is a need to also capture other important information on activities and relationships among activities. This need is discussed in the following chapter.

Rational Unified Process's approach

Before the process is defined, a literature review was conducted in order to identify existing requirements engineering/management processes. The review began with the requirements process workflow from the Rational Unified Process (RUP) [24], which is a product of Rational Software Corporation. RUP is well known for its ability to capture the best practices in the software development industry. Preliminary investigation shows that the requirements process by RUP seemed quite complete.

RUP, which utilizes Unified Modeling Language (UML)[1] [25], is a customizable framework for the software engineering process. One of the main features of RUP is that it is web-enabled. This allows users flexibility in accessing RUP through the Internet. RUP divides the software development lifecycle into four

[1] "The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems" [27]. UML is now considered a standard for modeling.

phases – inception (defining the scope), elaboration (planning the tasks), construction (producing the product), and transition (moving the product into end users). There are many activities within each phase, of which each group of activities is categorized as a process workflow. There are six process workflows and three supporting workflows. Each workflow produces models, which then is used by the subsequent workflow. The process workflows include business modeling, requirements, analysis and design, implementation, test, and deployment. The supporting workflows are made up of configuration and change management, project management, and environment. The level of activity for each workflow depends on the phase of the lifecycle. For instance, the requirements process workflow is more active during the inception and elaboration phases. As for construction and transition phases, requirements process workflows do not play a large role.

For the purpose of this research, only the requirements portion of the RUP was examined. The requirements process workflow is divided into six minor workflows – analyze the problem, understand the stakeholder needs, define the system, manage the scope of the system, refine the system definition, and manage changing requirements. Each minor workflow is a combination of the 14 applicable use cases[2]. The use cases are identified in the next section. Each use case then lists what tasks need to be accomplished, documentation required, and the roles involved. All this information is captured as a list of activities. The activities from RUP are used as a baseline for the process and are compiled in a document entitled 'Master

Activity List.' The list is discussed at the end of this section and included as Appendix A. Activities and supporting information from each subsequent source that are examined add to the Master Activity List.

Leffingwell and Widrig's approach

A supplementary source to RUP's requirements process, *Managing Software Requirements A Unified Approach* [18], was identified through a RUP workshop. The authors approach requirements management by requiring teams to learn and master five basic skills. The five basic skills are: analyze the problem, understand user needs, define the system, manage scope, refine the system definition, and build the right system. Each skill is further divided into more specific steps. The authors provide a handy summary at the end of the book of each skill and what it encompasses. However, a lot of important information was lost in the summary. The most crucial discovery was that this book, which was supposed to support RUP's material was actually quite different from RUP. The authors acknowledge a difference in terminology used but it seems more appropriate to use a standardized terms since this is referring to the same process! (This terminology problem becomes more prominent when other sources are introduced.) Table 4 shows the comparison between the use cases define in RUP and the skills by Leffingwell and Widrig [18].

---

[2] "A use case defines a set of use-case instances, where each instance is a sequence of actions a system performs that yields an observable result of value to a particular actor" [24].

Table 4
Comparison between RUP [24] and Leffingwell and Widrig's [18] book

No	Use cases from RUP [24]	Skills from Leffingwell and Widrig [18]
1	Capture a common vocabulary	
2	Develop requirements management plan	
3	Find actors and use cases	
4	Develop vision	Analyze problem
5	Elicit stakeholder request	Acquire user needs
6	Manage dependencies	
7	Review change request	Manage changes to requirements
8	Prioritize use case	
9	Detail a use case	
10	Detail the software requirements	
11	Model the user-interface	
12	Prototype the user-interface	
13	Structure use-case model	
14	Review requirements	

The activities described under "Analyze problem" by Leffingwell and Widrig is not the same as RUP's "Analyze the problem." In fact, it is only similar to the develop vision use case, which is a portion of RUP's "Analyze the problem" workflow. According to RUP, "Analyze the problem" workflow includes "Capturing a common vocabulary", "Develop requirements management plan", "Find actors and use cases", and "Develop vision use cases." A complete listing of the use cases within each RUP workflow is defined in Table 5.

Table 5
Composition of RUP's process workflows and their corresponding use cases

	Rational Unified Process	Use cases													
A	Analyze the problem	1	2	3	4										
B	Understand stakeholder needs	1		3	4	5	6	7							
C	Define the system	1		3	4		6								
D	Manage the scope of the system				4		6	7	8						
E	Refine the system definition									9	10	11	12		
F	Manage changing requirements						6	7						13	14

Gause and Weinberg's approach

A third source, the book entitled *Exploring Requirements Quality Before Design*, by Donald C. Gause and Gerald M. Weinberg [8] was investigated. The authors claim that there are many books written on requirements management's tools and techniques; however, they lack coverage of dealing with people within the requirements management environment. Gause and Weinberg [8] believe that more time has to be spent on people issues if they are provided with the better tool.

To help manage teams, the authors provide advice for selecting team members, conducting meetings, dealing with conflicts, making decisions, and knowing when to end the requirements exploration.

The authors also supply ideas for uncovering requirements. Topics covered under this section include brainstorming, sketching techniques, and naming projects.

One of the most important contributions from Gause and Weinberg [8] deals with ambiguity. The authors warn that ambiguity has a large impact on cost. They state that "[b]illions of dollars are squandered each year building products that don't meet requirements, mostly because the requirements were never clearly understood" (p.17). Therefore, the authors advocate attacking ambiguities at the beginning of the project. In order to get rid of ambiguity, the authors identify sources of ambiguity and discuss techniques for attacking ambiguity.

The later part of the book deals with fine-tuning product functions, attributes, and constraints. The last section covers the quality of requirements including measuring ambiguity, conducting technical reviews, measuring satisfaction, case testing, and studying existing products. Overall, this book is a good source for handling ambiguity but does not make a significant contribution towards defining the activities within requirements engineering/management process.

Hooks and Farry's approach

A fourth source, Hooks and Farry's [12] *Customer-Centered Products Creating Successful Products through Smart Requirements Management*, is written from a management perspective. The authors provide some insight into the American culture that defines how Americans work and think. They [12] attribute this to three out of the "seven cultural forces that define Americans" from Hammond and Morisson's book entitled *The Stuff Americans Are Made Of* [12], i.e. "impatience with time, acceptance of mistakes, and the urge to improvise" as the main causes of product development problems (p.17). Since the usual tendency for people is to want something done immediately, developers often want to jump into the design immediately, thinking that requirements type activities is a waste of time. In addition, people's willingness to accept mistakes makes it acceptable for the developers to make mistakes. Mistakes, sometimes costly, can be prevented had some time been spent up front defining requirements. The third issue is that people expect problems to arise in the middle of projects. So they improvise when necessary, suggesting that improvisation is acceptable. This again can be prevented had developers spent time in the beginning towards defining requirements.

In addition, Americans' work environment may not be conducive for requirements. Hooks and Farry [12] blames this on the five "management myths" in the American workplace.

1. "Everyone knows what this project is about."
2. "Everyone knows how to write requirements."
3. "We already have a requirement management process in place."
4. "Everyone understands our requirements management process."
5. "Nothing can be done about bad requirements." (p.21)

Unfortunately, culture and work environments are not the only culprits for most companies that lack a good requirement definition process. The other contributor is the individual; Hooks and Farry [12] claims that the person in charge of requirements oftentimes "doesn't know what to do, doesn't understand why, would rather be doing something else, or sees no reward" (p.25).

Hence, Hooks and Farry offer what is called the Requirement Management Process Sanity Check. It outlines steps for creating and managing requirements. Like other authors in the requirement engineering/management field, Hooks and

Farry advocate an organization adopt a requirements management process if they have not already done so. Their process is made up of nine iterative steps.

1. Scope product
2. Develop operational concepts
3. Identify interfaces
4. Write requirements
5. Capture rationale
6. Level requirements
7. Assess verification
8. Format requirements
9. Baseline requirements

Each step is further defined in their book. Each chapter includes a sanity checklist to ensure that all the issues are at least addressed and each chapter concludes with a short section on the manager's roles for each step. In addition to the creation of requirements, the authors also dedicate several chapters to the management of requirements. While they seem more like activities, Hooks and Farry define the following "techniques and tools":

1. Set priorities for requirements implementation and use these priorities to phase development
2. Automate requirement management
3. Control change to requirements and assess potential change impact before integrating changes

4. Measure quality of requirements and your progress toward good requirement management

A good requirements management process by itself is simply not enough to ensure success. The key is effective communication throughout the entire nine steps. In addition, someone has to take charge and deal with the culture, management, and individuals themselves. Hooks and Farry close the book by providing advice on how to do so.

Robertson and Robertson's approach

A fifth source is a book entitled *Mastering the Requirements Process* by Suzanne Robertson and James Robertson [27]. The authors found that system analysis is well documented but there is lack of resources for requirements process. This led the authors to come up with a process to help the requirements gathering process. Their process is named "Volere Requirements Process." The main activities of the process include project blastoff, trawl for knowledge, write the specification, quality gateway, analyze, design, build and take stock of the specification.

A major part of project blastoff is preparing for it. Interestingly, Robertson and Robertson paid attention to meeting preparations, such as facility and accommodation planning for participants. Other authors probably assumed that this was usually carried out automatically prior to meetings. However, information such as this is good for first-timers.

For the initial stages of requirements gathering, the authors suggest the use of the requirements shell. This ‘shell’ is a 5” by 8” card on which information is filled progressively. Information recorded include requirement number, requirement type, event/use case, description, rationale, source, fit criteria, customer satisfaction, customer dissatisfaction, dependencies, conflicts, supporting materials, and history. Eventually, all the requirements recorded in the cards will be transferred to an automated tool.

They introduce the notion of a “quality gateway” acts as a requirements filter to see if the particular requirement should be sent to the next stage (analyzing, designing, and building specifications) or be discarded. Basically, the requirements are tested for several qualities namely completeness, traceability, consistency, correctness, ambiguity, and viability. In addition to that, requirements are also checked to ensure that they are indeed requirements and not solutions. Requirements that are there just because it is nice-to-have are not necessary and these are also checked for. This is called ‘gold plating’. One last quality test is to find the requirements that creep or leak into the process after the requirements process is complete.

Another contribution by the authors is the guide for requirements documentation called ‘Volere Requirements Specification Template’. This document is also available online at http://www.systemsguild.com. Presently, the most current version is the 8th edition.

However, there are times when the authors appear to apply new words to existing concepts. For instance, trawling for knowledge is simply gathering requirements. In a book review article by Ivy Hooks [14], she thinks that new terms will only confuse readers. She does not recommend using the Volere process because she finds the process too similar to project management rather than requirements definition process. Nevertheless, Hooks [14] like the idea of the 'gateway quality' as to "sweeping up every requirement, or cutting and pasting from other specifications to create a specification and then trying to undo the bad requirements" (p.24).

IEEE standards on requirements

Three Institute of Electrical and Electronics Engineering (IEEE) standards on requirements were reviewed. The first document, IEEE Std 830-1998 -- IEEE Recommended Practice for Software Requirements Specification [IEEE830], provides guidelines for preparing a Software Requirements Specification (SRS) document. The content of the document is discussed and organization options are also provided.

According to the guidelines, a good SRS document includes three main sections – introduction, overall description, and specific requirements. The introduction portion should include the purpose, scope, definitions, acronyms, and abbreviations, references, and overview. Information included in the overall description is the product perspective, product functions, user characteristics, constraints, and assumptions and dependencies. The third section deals specifically with requirements. The standard recommends that this section include external interfaces, functions, performance requirements, logical database requirements, design constraints, standards compliance, software system attributes, and requirements organization. As with any document, a table of contents, appendixes and index should be provided.

Organization options for the requirements portion can vary from one to another. Annex A of the IEEE standard exemplify organizational options for the third section of the SRS document. Requirements can be organized based on system mode, user class, object, feature, stimulus or functional hierarchy. However, there are times when a combination of a few organizations is required.

The second document reviewed, IEEE Std 1233, 1998 edition -- IEEE Guide for Developing System Requirements Specifications (SyRS) [IEEE1233], discusses the System Requirements Specification document and the development process.

A subtle difference between this document and the previous one discussed is that this standard focuses on system requirements while the previous one concentrates on software requirements. Hence, the SRS is mostly used in-house for software development and SyRS is used as a communication tool between the customer and developers.

The development of the SyRS document involves several steps:

1) Identify requirements,
2) Write (define) requirements,
3) Organize the requirements into the SyRS document,

4) Present the requirements in a textual or model form for the audience.

Information obtained from this standard reinforced activities already found from other sources. As an aside, it is interesting to note that the authors for this standard reference Blanchard and Fabrycky's [3]1990 book entitled *Systems Engineering & Analysis* and also Gause and Weinberg's [8] 1989 book entitled *Exploring Requirements: Quality Before Design*. The authors for this standard provide a sample of the layout for the SyRS document yet stress that that was not the only way to organize the System Requirements Specification.

The third standard, IEEE Std 1220-1998 -- IEEE Standard for Application and Management of the Systems Engineering Process [31], is a revision of IEEE Std 1220-1994. Since this document examines the entire process, the relevant sections from this document include requirements analysis (section 6.1) and requirements validation (section 6.2). The main activity under requirements analysis is definition. Items defined include customer expectations, project and enterprise constraints, external constraints, operational scenarios, measures of effectiveness, system boundaries, interfaces, utilization environments, life cycle process concepts, functional requirements, performance requirements, modes of operations, technical performance measures, design characteristics, and human factors. All these definitions feed into a requirements baseline.

The next section involves checking to ensure that every aspect is covered in the definition stage. The requirements validation process consists of comparison to customer expectations, enterprise and project constraints, and external constraints.

Once this is completed, variances and conflicts can be identified. If necessary, the requirements analysis stage is revisited. Once all the variances and conflicts are resolved, a validated requirements baseline can be established.

Comparison of requirements engineering/management activities

Both similarities and differences exist between the activities by different sources. Table 6 shows the primary use case in the literature. Even though the headings differ from one source to the other, it is clear that no author(s) suggest diving straight into writing requirements. Instead, they recommend some sort of planning and analyzing activities before plunging into requirements. Since all the sources included talking to customers about their needs, it is also clear that the customers' input play an important role in the requirements process. However, note that each source uses different terminology and can potentially create confusion. Hence, a dictionary of commonly used terms should be created. A good starting point is IEEE Std 61.012-1990, IEEE Standard Glossary of Software Engineering Terminology [34].

Table 6
Comparison of primary use cases from the literature

Rational Unified Process, RUP [24]	Leffingwell and Widrig [18]	Gause and Weinberg [8]	Hooks and Farry [12]	Robertson and Robertson [27]	IEEE Std 1220-1998 [31]
Capture a common vocabulary	Understand the problem being solved	Negotiating a common understanding	Scope product	Project blastoff	Requirements analysis
Develop requirements management plan	Understand user needs	Ways to get started	Develop operational concepts	Trawling for knowledge	Requirements validation
Find actors and use cases	Define the system	Exploring the possibilities	Identify interfaces	Write the requirements	
Develop vision	Continuously manage scope and manage change	Clarifying expectations	Write requirements	Quality gateway	
Elicit stakeholder request	Refine the system definition	Greatly improving the odds of success	Capture rationale	Prototype the requirements	
Manage dependencies	Build the right system		Level requirements	Do requirements post mortem	
Review change request	Manage the requirements process		Assess verification	Taking stock of the specification	
Prioritize use case			Format requirements		
Detail a use case			Baseline requirements		
Detail software requirements					
Model the user interface					
Prototype the user interface					
Structure use case model					
Review requirements					

Assimilation of a Master Activity List

Each literature advocates using their method for requirements engineering/management yet the methods that they (the authors) propose is inconsistent. Some provide lots of information while some provide little (if any) information. Overall, the cited literature provides vast information that needed to be captured in a standardized form. Hence, there was a need to pull the information together into one document. Valuable information from each source was assimilated and converted into use cases.

The result of this investigation is a high-level list of tasks list and sources. A portion of this Master Activity List is shown in Table 7. The entire Master Activity List is provided found in Appendix A. However, this list is not adequate because it does not provide information as to the necessary inputs, outputs, supporting documentation, etc. This issue is discussed in the following section.

The main use cases in the Master Activity List are further defined by classifying them either as requirements engineering or requirements management based on the description of the particular use case. They are further divided into key activity categories. Requirements engineering use cases are categorized as elicitation, modeling, specification or validation. Requirements management use cases are categorized as organization, control or documentation. The result of the groupings and categorization is provided in Table 8.

Table 7
A portion of the high level Master Activity List and sources

Use case no.		Name	Source
1		Capture a common vocabulary	RUP [24]
2		Develop requirements management plan	RUP [24]
3		Find actors and use cases	RUP [24]
	3.1	Establish scope of work	R & R [27]
	3.2	Establish adjacent systems that surround the work	R & R [27]
	3.3	Identify connections between the work and the adjacent systems	R & R [27]
	3.4	Identify business events that added the work from the connections	R & R [27]
	3.5	Study the response to the event	R & R [27]
	3.6	Determine best response that the organization can make for the event	R & R [27]
	3.7	Determine product's role in the response	R & R [27]
	3.8	Determine the use case or cases	R & R [27]
	3.9	Derive the requirements for each use case	R & R [27]
…	…	…	…

Table 8
Grouping and Categorization of the Main Use Cases

Use case #	Use case	Requirements Engineering / Requirements Management	Categories
1	Capture a common vocabulary	Requirements engineering	specification
2	Develop requirements management plan	Requirements Management	organization
3	Find actors and use cases	Requirements engineering	specification
4a	Develop vision	Requirements engineering	specification
4b	Project blastoff	Requirements engineering	specification
5a	Elicit stakeholder request	Requirements engineering	elicitation
5b	Trawling for requirements	Requirements engineering	elicitation
6	Identify both external and internal interfaces	Requirements engineering	specification
7	Writing good requirements	Requirements Management	documentation
8	Capture rationale	Requirements Management	control
9	Manage dependencies	Requirements Management	control
10	Verify requirements	Requirements engineering	validation
11	Format requirements	Requirements Management	documentation
12a	Baseline requirements	Requirements Management	control
12b	Check requirements (quality gateway)	Requirements engineering	validation
12c	Check requirements for certain properties	Requirements engineering	validation
13	Prioritize requirements	Requirements engineering	specification
14	Review change requests	Requirements engineering	validation
15	Prioritize use case	Requirements engineering	validation
16	Detail a use case	Requirements engineering	modeling
17	Detail software requirements	Requirements engineering	modeling
18	Model the user interface	Requirements engineering	modeling
19	Prototype the user interface	Requirements engineering	modeling
20	Structure the use case model	Requirements engineering	modeling
21	Do requirements post mortem	Requirements Management	control
22a	Review requirements	Requirements Management	organization
22b	Taking stock of specification	Requirements Management	control

Development a Process Representation Scheme

Review of Representation Methods by Cited Sources

Most of the sources used some form of graphical representation to define their process. Each representation method has its own strengths and weaknesses; they are summarized in Table 9.

Table 9
Representation methods used by the cited sources

Source	Representation method	Strength(s)	Weakness(es)
RUP [24]	Use case diagrams[3] grouped into workflows	Interaction between activities and actors is clear	Sequence is not clear, interactions between use cases are not clear
Leffingwell and Widrig [18]	Use case diagrams[3]	Interaction between activities and actors is clear	Sequence is not clear, interactions between use cases are not clear
Hooks and Farry [12]	N/A[4]	N/A	N/A
Robertson and Robertson [27]	Stylized data flow diagram[5]	Interactions between main activities is clear	Sequence is not that clear
IEEE Std 1220-1998 [31]	Unknown[6]	Sequence is clear	Accountability is not clear, inputs and outputs are not clear

[3] Use case diagrams shows "the relationship among actors (someone or something outside the system that interacts with the system) and use cases within the system" [24].
[4] The authors show their overall process in a waterfall model but did not elaborate much on it in later chapters.
[5] Stylized data flow diagram, composed of bubbles (activities) and arrows (deliverables), presents an iterative and evolutionary process.
[6] There is no indication of the type of chart that was used. It looks similar to a flowchart. This chart uses top down approach, showing the flow and sequence of tasks.

Several issues were identified when attempts were made to represent a generic requirements engineering/management process. Since the activities within the use cases of RUP are not represented in any graphical form, activity diagrams[7] were applied. Activity diagrams worked as long as there was only one main source of information. As more information from different sources were added, it became difficult to track where the information came from because activity diagrams do not allow for source tracking. Efforts to add information to activity diagrams seemed impossible without losing its source.

Therefore, a more systematic representation method is required to keep track of all the information provided by different sources. This method must allow for addition or deletion of information. In general, there are many ways to represent activities and processes. Examples of these are summarized in Table 10.

[7] Activity diagrams graphically describe the ordering of tasks or activities to accomplish business goals [24].

Table 10
General process representation methods in general

Representation method	Strength(s)	Weakness(es)
Flowchart	Easy to use and understand, flow is clear	Accountability is not clear
Integration Definition for Function Modeling (IDEF0) [6], [23]	Activities within functions are clear, processes can be documented at different levels, inputs and outputs are clear, hierarchical breakdown of function is possible, sequence is clear, easy to use	Accountability is not clear, static – not suitable for frequently changing models, time and cost for carrying out process not taken into account, data stores is not clear, data and material flow is not clear
Integration Definition for Function Modeling (IDEF3) (process-centered view) [20]	Processes flow are clear, precedence relationships or constraints are clear, effects of the constraints on the process are clear	Accountability is not clear
Integration Definition for Function Modeling (object-centered view) (IDEF3) [20]	Changes that occur on objects throughout a process are clear	Accountability is not clear
Relationship maps [28]	Relationships between departments/functions are clear, inputs to and outputs from each department/function is clear	Applicable for organizational level only
Process maps [28]	Accountability is clear, actions taken by departments/functions are clear, goals are clear	Applicable for process level only
Role/responsibility matrix [28]	Responsibilities and goals for each personnel based on function is clear	Applicable for job/performer level only, tabular view
Use case	Standard, written in user language, interaction between actors and use case are clear	Sequence is not clear
Data flow diagram (DFD) [23]	Focuses on the flow of data, inputs and outputs are clear, easy to understand and modify	Logic within processes is not clear, structure of data is not clear, hard to create
Activity diagrams	Sequencing of activities are clear	Hard to keep track of updates
Entity-relationship diagram (ERD)	Relationships and conditions for the relationship are clear	Inexperienced users may find it hard to understand

Integration Definition for Function Modeling (IDEF)

This investigation led to Integration Definition for Function Modeling (IDEF) as the main technique and incorporates other elements from other diagramming techniques. There are many types of IDEF; however IDEF0 and IDEF3 are the most applicable. IDEF0 [6] is used for function modeling and IDEF3 [20] is used for process flow and object transitions.

The basic IDEF0 representation is shown in Figure 7. Activities are named with verb-noun phrases. The method of reading this diagram is <input> are <verb> into <output> according to <control>, using <mechanism>. Inputs and outputs are self-explanatory. Controls are items that restrict the activity; examples include constraints, limitations or conditions on the activity. Mechanisms are methods by which particular activities are achieved.

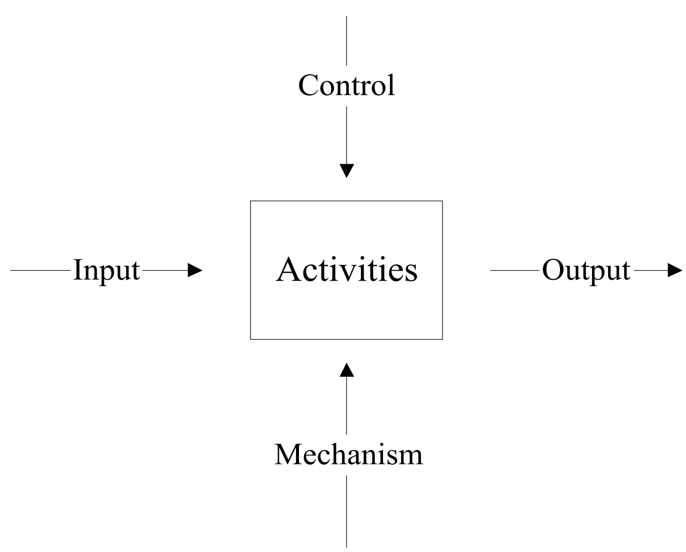

Figure 7: Basic structure of IDEF0

The process-centered view for IDEF3 is shown in Figure 8. Each of the rectangular boxes represents an activity, indicated by the letters. An advantage of IDEF3 is that the arrows indicate precedence or constraints. For instance, in Figure 8, activity A has to be done before activity B begins. This is different from the precedence between activity C and activity E because the single headed arrow indicates that activity E can start with or without the completion of activity D. The junction box after activity B and before activity C and activity D is an OR condition, indicating that one can choose activity C or activity D or both. The junction box before activity F is a synchronous AND. This means that activity E and D must end at the same time and precede activity F. The numbers within each box is for identification purposes.

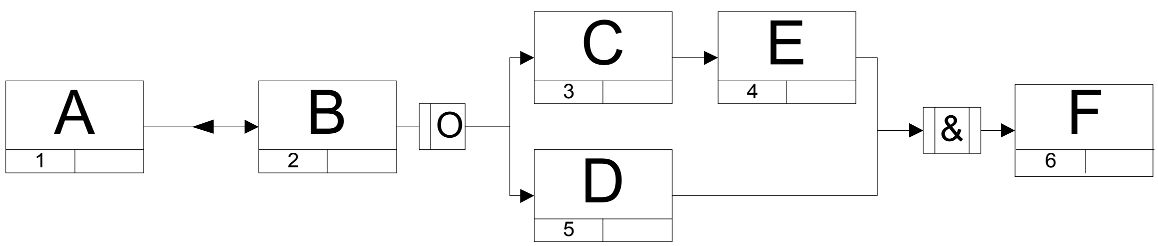

Figure 8: Example of a process-centered view of IDEF3

The state-centered view for IDEF3 is shown in Figure 9. The circles indicate the state an object. For instance, the object changed from p state to q state. The rectangle between state p and state q shows the activity that causes the stage to change from state p to state q. The exclusive OR in the figure indicates that either state r or state s result from activity B, e.g. am object may be considered normal or defective as a result of activity B.

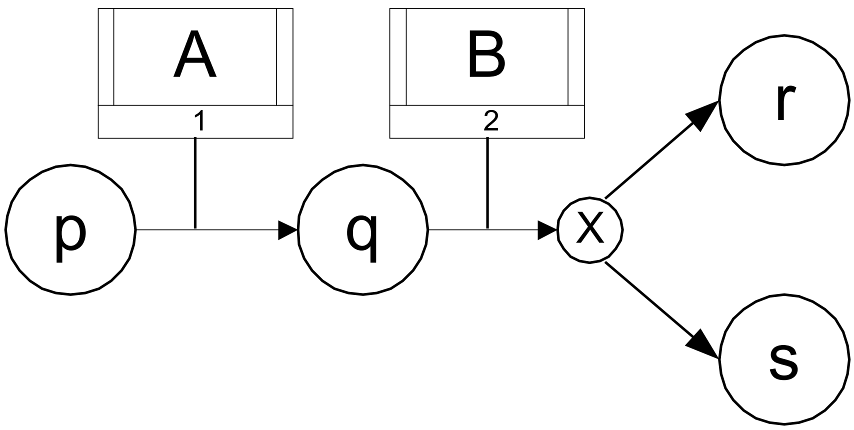

Figure 9: Example of object-centered view of IDEF3

<u>Tabular View</u>

The next step in the research is to represent all of the information that was gathered for the activities from the Master Activity List. However, diagramming was not possible at this point because all that was collected so far was just a list of activity along with sources. Hence, there is a need for a method to capture all the information provided such as a description of what the activity does, who is involved, when is it carried out, and using what means. A table, containing attributes of the tasks and processes as columns, is created in order to incorporate the strengths of the various representation methods. The activity list is expanded to include a description of the activity and also the result/output. Information about the task performer is also desired. Therefore, a column separating primary performer and support performer is created. In order to capture when the activity is to be carried out, two columns are used – input and control (constraints, policies, etc). Each activity uses methodologies and this is captured as guidelines, tools, and/or templates. The last column – notes, is added to include any information that did not directly fit in the other columns. Table 11 shows the main structure of the tabular view, along with an example use case. The description of the example is discussed in the next section.

Information from the six main sources is used to populate the tabular view progressively. Typically the sources do not explicitly specify the information as inputs, outputs, controls, and mechanisms; therefore they have to be gleaned from a textual representation, interpreted, and translated into the table format. However, the approach proposed in this research provides a convenient means to organize the information. The result is a database of activities and associative characteristics for requirements engineering/management.

The lack of information from the sources creates "holes" in the database that indicate a need for more information about a particular activity. For instance, for use case 3.1, "Establish scope of work" (Appendix B), no information is provided on who will do the work or what guidelines and tools are to be used. In other words, the source lists that the scope of the work has to be established but does not provide much guidance on doing so. Additions to the tabular can be made as more sources/information become available. This implies that the database needs to be a living document. The complete database of activities and associative attributes developed in this research is provided in Appendix B.

Table 11
Tabular view of process

	What			Role (Who)		When		How (mechanism)				
Use case no.	Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates	Source	Notes
9.1	A	Activity A	4	Project manager (John)	-	1	-	-	-	-	-	-
9.2	B	Activity B	5	Project manager (John)	-	4	-	-	-	-	-	-
9.3	C	Activity C	6	System analyst (Judy)	-	2	-	-	-	-	-	-
9.4	D	Activity D	12	System analyst (Judy)	John and Jessie	6 & 7, 8	5	10	9	11	-	-
9.5	E	Activity E	7	Customer (Jessie)	-	3	-	-	-	-	-	-

Hybrid Graphical View

The tabular method is very good for helping users structure the problem. In addition, any missing information on a particular activity is more apparent via the tabular method. However, information from the tabular view can be transferred into a hybrid graphical view; hybrid in that it captures the best features of IDEF0, IDEF3 and process maps. Recall that IDEF0 is able to represent functions and their relationships among them hierarchically [6] and IDEF3 is for useful for charting the flow of a process. It also allows representation of semantics (AND, OR, XOR, synchronous AND, and synchronous OR). Process maps are good for indicating activities that span across different organizational units.

Figure 10 is an example of the proposed hybrid graphical view. Swim lanes are included to indicate who or what role is performing the activity. In this example, there are three task performers – John, Judy, and Jessie. John will be in charge of activity A and B, Judy activity C, and Jessie activity E. All John, Judy and Jessie will be required to carry out activity D (the shaded area indicates Judy has primary responsibility). However, activities C and E must be completed prior to the start of activity D. Activity B results in a control for activity D. Activity D uses a set of mechanisms (tool, guideline, template). Activity E has additional information. This is captured in the notes box.

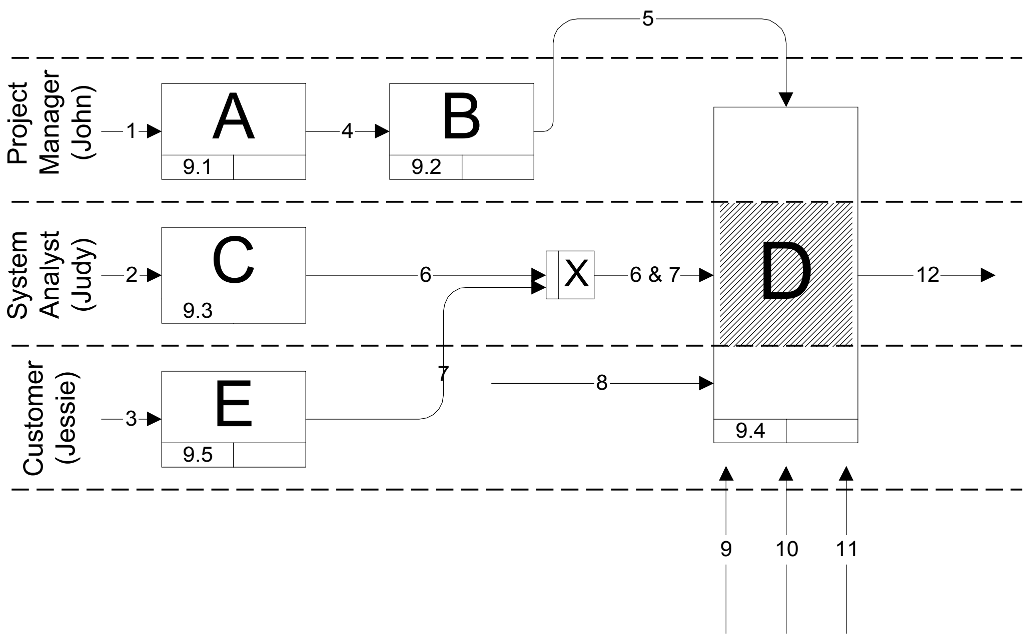

Figure 10: Example of the proposed hybrid graphical view

The next step is to represent the tabular view in the hybrid graphical view. However, this is a major challenge task because the tabular view lacks information in many areas (denoted by the "holes"); these "holes" are represented by a question mark. Therefore, an attempt was made to create a hybrid graphical view based on one use case. Use case 5, which appears complicated in the tabular view, was selected for the example. (Due to space limitation, the entire representation is not included in this thesis.) In doing so, several issues became apparent.

First, information between use cases does not match. Take for instance, use case 5.3 and 5.4 shown in Figure 11.

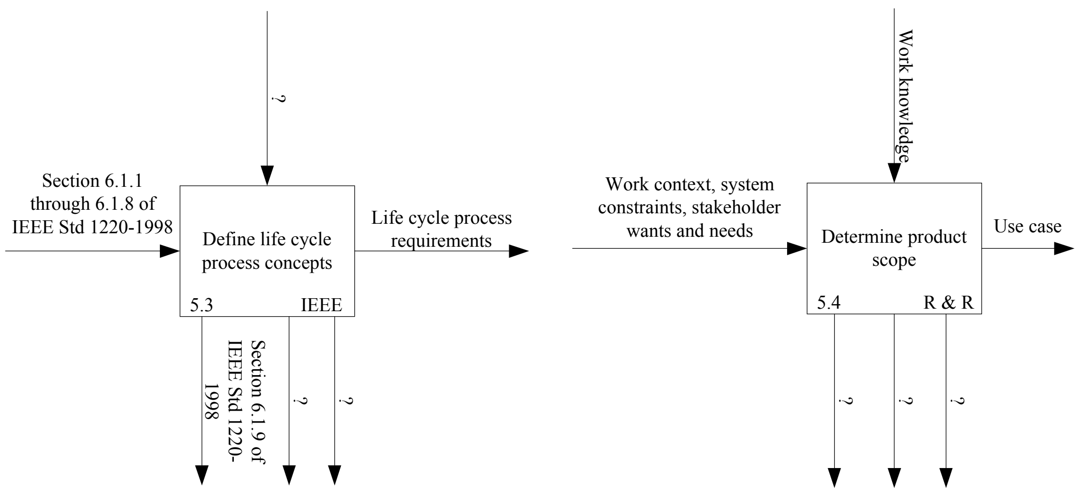

Figure 11: Use cases 5.3 and 5.4

Theoretically, one should be able to trace the flow from the beginning to the end. However, this is not the case in use case 5.3 and 5.4. The output from use case 5.3, life cycle process requirements, should be an input to use case 5.4 but the input for use case 5.4 is work context, system constraints, stakeholder wants, and needs. This is due to the fact that these two use cases originated from different sources. Use case 5.3 originates from IEEE [31] while use case 5.4 is from Robertson and Robertson [27].

Second, information between use cases from the same source also does not match. Take for instance, use case 5.4 and 5.5, as shown in Figure 12.

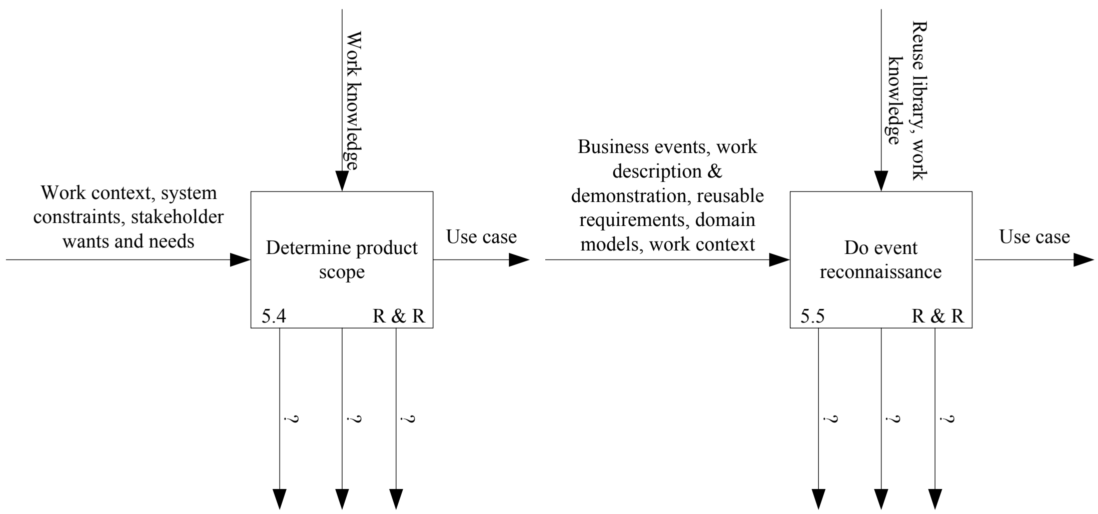

Figure 12: Use case 5.4 and 5.5

This example clearly illustrates that the output from use case 5.4 does not match the input for use case 5.5. Swim lanes were not included in the example because there is only one main person in charge – requirements analyst or systems analyst. It is assumed that the responsibilities played by each role are the same due to the fact that different sources mention different roles.

Hence, this example hybrid graphical view indicate that more work is required in order to create a complete hybrid graphical view similar to the proposed one. Research should be conducted to investigate if certain terms can be combined or if better terminology can be used. Another research issue is to reorganize the order or flow of the use cases. All use cases should be further examined to see if they can be combined or redefined to enhance their integration.

Tabular View versus Hybrid Graphical View

Each view has its own advantages and disadvantages. In addition to helping structure a user's thoughts, the tabular view also allows users to perform such operations as query, filter, and sort, e.g. filter the sources to see the activities that were derived from each source. This advantage for the tabular view automatically becomes a disadvantage for the hybrid graphical view. Compared to the tabular view, information from different sources can become quite complicated in the hybrid graphical view. For instance, there are two guidelines for use case number 5a, elicit stakeholder request, one from Rational Unified Process (RUP) and the other from IEEE standard. In order to keep track of where each guideline came from, the 'guidelines' arrow on the hybrid graphical view would have to include the sources. The situation could get more complicated since each arrow on the hybrid graphical view could have multiple sources. Another advantage of the tabular view is that it allows users to identify areas where further research is required, i.e. the "holes." This may not be as obvious in the hybrid view.

The hybrid graphical view's strength is that it allows users to see the entire flow of the activities within the requirements engineering/management process; whereas the flow is not clear in the tabular view.

CHAPTER III
CONCLUSION

Future Research

The two views developed in this research (tabular and hybrid graphical) can be extended by linking them together. The main reason for doing so is to prevent anomalies due to update, insertion, and deletion. This linkage between the two views would also make maintenance easier; once one of the views is updated, the corresponding changes are reflected in the other view.

Ideally, there should be a direct link between each entity in both views. In other words, each element in the tabular view should be represented in the hybrid graphical view, and vice versa. The information in the tabular view can be represented in and supported by a database where the table columns are the database fields and each use case is a record.

Once this link between the tabular view and hybrid graphical view is set up, other links can also be incorporated. The following table is a list of potential extension links that can be made from both views.

Table 12
Extension links from the tabular and hybrid graphical view

Columns or Entities	Extended links
Notes	Text document
Mechanism (tools)	Specific tool or software (located locally or on the web)
Mechanism (guidelines)	Standards, checklists, references, tutorials, rules, regulations (located locally or on the web)
Mechanism (templates)	Text document, graphical tool (located locally or on the web)
Role (entity) - primary and support	Personnel information, contact information, organizational unit

The requirements engineering/management process should then be tested in industry. The steps within the requirements engineering/management process would be customized to fit their needs. Feedbacks from the industry application would provide further improvements to the generic process, as they would refine and/or extend the use cases.

Another important future activity is to combine and/or eliminate activities within the process since the process is now in its "purest" form (i.e. documented exactly based on each source). This process refinement, along with industries' feedback, would result in a generic process for requirements engineering/management.

A further enhancement would be to develop an implementation tool, most likely in a hypermedia environment, i.e. a web page site with links to tools, guidelines, etc.

Conclusion

Requirements are important and can often determine the success of the end product. However, the current literature does not provide sufficient information to adequately define requirements as a process. Inconsistent and vague information was the motivation for this research which attempted to assimilate the information into one common framework. As a step to meet that need, this research accomplished three objectives: defining requirements engineering and requirements management, developing a generic process for requirements engineering/management, and developing a process representation scheme.

During the extensive research on the terms 'requirements engineering' and 'requirements management', various definitions were found. All these definitions were compiled into a common yet comprehensive definition of requirements engineering and requirements management. It is proposed that both terms are separate but related terms. It also proposed that requirements engineering is composed of requirements elicitation, requirements modeling, requirements specification, and requirements validation, while requirements management is composed of requirements organization, requirements control, and requirements documentation.

The investigation on requirements engineering/management process concludes that no generic methodology currently exists. Therefore, the vast information provided by the six main sources was assimilated and converted into a Master Activity List. However, this list has its limitations because this list only specifies the activities. There is a need to include information about who carries out the activity, when the activity is carried out or what is required to carry out the activity, etc. in the Master Activity List. This need was later fulfilled in the next research objective.

In addition, a means to represent the requirements engineering/management process does not currently exist. This shortcoming, along with the need for a structured approach to capture the supporting information about a particular activity, prompted the creation of a tabular view and a hybrid graphical view. These two views complement one another. The tabular view is a good method for structuring user's thoughts. However, it does not show the flow of the activities. This inadequacy is fulfilled by the hybrid graphical view.

Then again, these two views – tabular view and hybrid graphical view, yielded in several issues that became apparent after the views were created. First, there are disconnects between use cases due to the fact that the use cases originated from different sources. Second, disconnects are still visible even within use cases from the same sources. These two issues indicate a need to further examine the use cases in the tabular view to see if the use cases can be combined, eliminated or refined to yield a generic process for requirements engineering/management.

REFERENCES CITED

[1] Alexander, I., "Systems Engineering - a Requirements Engineer's Viewpoint." Retrieved October 2001, from http://easyweb.easynet.co.uk/~iany/consultancy/systems_engineering/systems_engineering.htm

[2] Alexander, I., "What is RE Anyway?." Retrieved October 2001, from http://easyweb.easynet.co.uk/~iany/consultancy/what_is_re.htm

[3] Blanchard, B.S., Fabrycky, W. J., *Systems Engineering and Analysis*, 3rd Edition, New Jersey: Prentice Hall, 1998, p.17-43.

[4] Bubenko, J.A., Jr, "Challenges in requirements engineering," *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, 1995, p160-162. Retrieved October 2001, from IEEE Xplore database.

[5] Davis, A. M., Zweig, A. S., "Requirements management made easy," *PM Network*, December 2000, p61-63.

[6] Draft Federal Information Processing Standards Publication 183, "Integration Definition for Function Modeling (IDEF0)," December 1993. Retrieved November 2001, from http://www.idef.com/idef0.html

[7] Fifth IEEE International Symposium on Requirements Engineering held August 27-31, 2001 in Toronto, Canada. Retrieved October 2001, from http://www.re01.org/

[8] Gause, D. C., Weinberg, G. M., *Exploring Requirements Quality Before Time*, New York: Dorset House Publishing Co., 1989.

[9] Gilb, T, "Viewpoints: Towards the Engineering of Requirements," *Requirements Engineering*, 1997 (2), p165-169. Retrieved October 2001, from http://rej.co.umist.ac.uk/Volume-2/Issue-3/Viewpoints.html

[10] Harwell, R., Aslaksen, E., Hooks, I., Mengot, R., Ptack, K., "What is a Requirement?," in *Proceedings of the Third International Symposium of the NCOSE,* 1993. Retrieved September 2001, from http://www.incose.org/rwg/what_is.html

[11] Hoffer, J.A, George, J. F., Valacich J.S., *Modern Systems & Analysis Design*, 2nd edition, Reading: Addison Wesley Longman, Inc., 1999, p.24-31.

[12] Hooks, I. F., Farry, K., *Customer-Centered Products Creating Successful Products Through Smart Requirements Management*, New York: Amacom, 2001.

[13] IEEE Joint International Requirements Engineering Conference to be held Sept 9-13, 2002 at University of Essen, Denmark. Retrieved October 2001, from http://www.re02.org/

[14] International Council on Systems Engineering, "Special Issue on Requirements - Sharing the Vision, *Insight*, Winter 1999-2000, Vol. 2 (4).

[15] International Council on Systems Engineering, "What is Systems Engineering?" Retrieved February 2002, from http://www.incose.org/whatis.html

[16] Jones, D. A., Kar, P.C., Gaasbeek, J. R. V., Hollenbach, F., Bell, M., Ellinger, R.S., "Interfacing requirements management tools in the requirements management process - a first look," *Proceedings of the Seventh International Symposium of the INCOSE,* Vol. 2, August 1997. Retrieved September 2001, from http://www.incose.org/rwg/97_paper_inter/inter_rmt.html

[17] Kulak, D., Guiney, E, *Use Cases Requirements in Context*, New York: Addison-Wesley, 2000.

[18] Leffingwell, D., Widrig, D., *Managing Software Requirements A Unified Approach*, Boston: Addison-Wesley, 2000.

[19] Magrab, E. B., *Integrated Product and Process Design and Development The Product Realization Process*, Boca Raton: CRC Press, 1997.

[20] Mayer, R..J., et al., *Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report*, College Station, TX: Knowledge Based Systems, Inc., September 1995. Retrieved November 2001, from http://www.idef.com/idef3.html

[21] Mizuno, S., Akao, Y., *QFD The Customer-Driven Approach to Quality Planning and Deployment*, Hong Kong: Nordica International, Ltd., 1994.

[22] Oberg, R, Probasco, L., Ericsson, M., "Applying Requirements Management with Use Cases." Retrieved October 2001, from http://www.rational.com/products/whitepapers/100622.jsp

[23] Pandya, K. V., Karlsson, A., Sega, S., Carrie, A., "Towards the Manufacturing Enterprises of the Future," *International Journal of Operations & Production Management*, Vol.17 (5), p. 502-521.

[24] Rational Software Corporation, "Rational Unified Process," Version 2001.03.00.

[25] Rational Software Corporation, "UML Resource Center." Retrieved November 2001, from http://www.rational.com/uml/index.jsp

[26] Requirements Engineering Specialist Group (RESG) of the British Computer Society. Retrieved October 2001, from http://www.resg.org.uk/

[27] Robertson, S, Robertson, J., *Mastering the Requirements Process*, Great Britain: Biddles Ltd., 1999.

[28] Rummler, G.A., Brache, A.P., *Improving Performance How to Manage the White Space on the Organization Chart*, 2nd edition, San Francisco: Jossey-Bass Inc., 1995.

[29] Society of Concurrent Product Development. Retrieved February 2002 from http://www.soce.org/index.htm

[30] Software Engineering Standards Committee of the IEEE Computer Society, *IEEE Guide for Developing System Requirements Specifications*, December 1998. Retrieved October 2001, from IEEE Xplore database.

[31] Software Engineering Standards Committee of the IEEE Computer Society, *IEEE Std 1220-1998, IEEE Standard for Application and Management of the Systems Engineering Process*, January 1999, p34-42. Retrieved October 2001, from IEEE Xplore database.

[32] Software Engineering Standards Committee of the IEEE Computer Society, *IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications*, October 1998. Retrieved October 2001, from IEEE Xplore database.

[33] Software Productivity Solutions, Inc., "Analysis of Automated Requirements Management Capabilities - Requirements Management Technology Overview," June 1994. Retrieved January 2001, from http://www.incose.org/tools/reqsmgmt.html

[34] Standards Coordinating Committee of the IEEE Computer Society, *IEEE Std 61.012-1990, IEEE Standard Glossary of Software Engineering Terminology*, Institute of the Electrical and Electronics Engineers, Inc., December 1990, p.7-82. Retrieved October 2001, from IEEE Xplore database.

[35] Stevens, R., Martin, J., "What is Requirements Management?" Retrieved October 2001, from http://www.telelogic.com/download/paper/what_is_req_mgmt.pdf

[36] Ulrich, K. T., Eppinger, S. D., *Product Design and Development*, 2nd edition, Boston: McGraw Hill Companies, Inc, 2000, p.14-18.

[37] Zowghi, D., "Requirements Engineering Scope." Retrieved October 2001, from Joint Research Centre for Advanced Systems Engineering Web site: http://www.jrcase.mq.edu.au/~didar/seweb/scope.html

[38] Zowghi, D., Offen, Ray, "A Logical Framework for Modeling and Reasoning about the Evolution of Requirements," in *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, 1997, 247-257. Retrieved October 2001, from IEEE Xplore database.

APPENDIX A

MASTER ACTIVITY LIST

Use case no.					Name	Source
1					Capture a common vocabulary	RUP
	1.1				Find common terms	RUP
	1.2				Evaluate results	RUP
2					Develop requirements management plan	RUP
	2.1				Establish traceability	RUP
	2.2				Choose requirements attributes	RUP
	2.3				Map to tools	RUP
	2.4				Write the plan	RUP
3					Find actors and use cases	RUP
	3.1				Establish scope of work	R & R
	3.2				Establish adjacent systems that surround the work by looking outside the organization	R & R
	3.3				Identify connections between the work and the adjacent systems	R & R
	3.4				Identify business events that added the work from the connections	R & R
	3.5				Study the response to the event	R & R
	3.6				Determine best response that the organization can make for the event	R & R
	3.7				Determine product's role in the response	R & R
	3.8				Determine the use case or cases	R & R
		3.8.1			Find actors	RUP
		3.8.2			Find use cases	RUP
		3.8.3			Describe how actors and use cases interact	RUP
		3.8.4			Package use cases and actors	RUP
		3.8.5			Present the use-case model in the use-case diagrams	RUP
	3.9				Derive the requirements for each use case	R & R
	3.10				Develop a survey of the use-case model	RUP
	3.11				Evaluate results	RUP
4a					Develop vision	RUP
	4.1				Gain agreement on the problem being solved	L & W and RUP
	4.2				Identify primary need	H & F
	4.3				Understand root causes	L & W
	4.4				Circulate problem statement	L & W
	4.5				Revise where necessary	L & W
	4.5				Review and obtain agreement	H & F
	4.6				Identify stakeholders and users	RUP and HHP
	4.7				Obtain stakeholders' needs	HHP
	4.8				Identify goals and objectives	H & F

Use case no.					Name	Source
	4.9				Distribute and discuss goals and objectives with stakeholders	H & F
	4.10				Determine mission statement or business case (if any)	H & F
	4.11				Distribute it and gain consensus	H & F
	4.12				Identify budgets	H & F
	4.13				Identify schedule	H & F
	4.14				Define solution system boundaries	L & W and RUP
	4.15				Identify constraints to be imposed on the system	L & W and RUP
	4.16				Determine if work can be realistically done within budget and schedule constraints	H & F
	4.17				Identify major assumptions	H & F
	4.18				Validate assumptions	H & F
	4.19				Assign responsibilities	H & F
	4.2				Formulate problem statement	RUP
	4.21				Define features of the system	RUP
	4.22				Evaluate results	RUP
4b					Project blastoff	R & R
	4.1				Prepare for blastoff meeting	R & R
		4.1.1			Define blastoff objectives	R & R
		4.1.2			Plan physical arrangements	R & R
			4.1.2.1		Determine participants	R & R
			4.1.2.2		Plan facilities and accommodation for participants	R & R
		4.1.3			Communicate with participants	R & R
			4.1.3.1		Send each participant an agenda and list of participants	R & R
	4.2				Run blastoff meeting	R & R
		4.2.1			Determine product purpose	R & R
		4.2.2			Determine the work context	R & R
			4.2.2.1		Ask if there is a physical entity that represents domain	R & R
			4.2.2.2		Ask if domain provides data, policy or both to the work	R & R
			4.2.2.3		Identify sources of information for this domain	R & R
		4.2.3			Do first-cut risk analysis	R & R
			4.2.3.1		Identify risks that are most likely to happen	R & R
			4.2.3.2		Identify risks that would have the greatest impact of becoming a problem	R & R
			4.2.3.3		Assess probability of risk becoming a problem	R & R

Use case no.					Name	Source
			4.2.3.4		Assess its cost and schedule impact	R & R
			4.2.3.5		Identify actions to take if risks come true	R & R
		4.2.4			Identify the stakeholders	R & R
			4.2.4.1		Inform stakeholders that they are stakeholders and that they will be consulted about requirements	R & R
			4.2.4.2		Inform stakeholders of time required and type of participation	R & R
		4.2.5			Partition the work	R & R
		4.2.6			Consider non-events	R & R
		4.2.7			Determine system terminology	R & R
		4.2.8			Define project constraints	R & R
		4.2.9			Identify domains of interest	R & R
	4.3				Finalize blastoff	R & R
		4.3.1			Write blastoff report	R & R
		4.3.2			Review blastoff results	R & R
		4.3.3			Hold follow-up blastoff	R & R
		4.3.4			Make initial estimate	R & R
5a					Elicit stakeholder request	RUP
5b					Trawling for requirements	R & R
	5.1				Determine sources for requirements	RUP
	5.2a				Gather information	RUP
	5.2b				Learn the work	R & R
		5.2.1			Review current situation	R & R
		5.2.2			Apprentice with the user	R & R
		5.2.3			Determine essential requirements	R & R
		5.2.4			Brainstorm the requirements	R & R
		5.2.5			Create structured interviews	L & W
		5.2.6			Conduct 5 to 15 interviews	L & W
		5.2.7			Summarize interviews	L & W
		5.2.8			Do document archeology	R & R
		5.2.9			Make requirements video	R & R
		5.2.10			Run use case workshop	R & R
		5.2.11			Build event models	R & R
		5.2.12			Build scenario models	R & R
			5.2.12.1		Define technical performance measures (TPMs)	IEEE
			5.2.12.2		Define design characteristics	IEEE
			5.2.12.3		Define human factors	IEEE
		5.2.13			Run requirements workshop	L & W and RUP
		5.2.14			Brainstorming	L & W
		5.2.15			Mind map requirements	R & R
		5.2.16			Collect requirements via Volere Snow Cards	R & R
		5.2.17			Reduce ideas	L & W

Use case no.					Name	Source
			5.2.17.1		Pruning	L & W
			5.2.17.2		Grouping ideas	L & W
			5.2.17.3		Feature definition	L & W
			5.2.17.4		Prioritization	L & W
		5.2.18			Create storyboards for innovative concepts	L & W
		5.2.19			Create operational concepts	H & F
			5.2.19.1		Develop concept for each phase of the lifecycle	H & F
				5.2.19.1.1	Outline normal operation and environment	H & F
				5.2.19.1.2	Outline abnormal operation and environment	H & F
			5.2.19.2		Consider viewpoints of all stakeholders	H & F
			5.2.19.3		Assess human interface standard	H & F
			5.2.19.4		Create use cases	L & W
		5.2.20			Role play	L & W
		5.2.21			Create prototypes	L & W
	5.3				Define life cycle process concepts	IEEE
	5.4				Determine product scope	R & R
		5.4.1			Set priorities for each feature	L & W
		5.4.2			Assess effort for each feature	L & W
		5.4.3			Estimate risk for each feature	L & W
		5.4.4			Reduce scope based on priorities, effort, and risk	L & W
		5.4.5			Determine baseline for each release of Vision Document	L & W
		5.4.6			Get customer agreement on scope	L & W
		5.4.7			Advocate and practice iterative development	L & W
		5.4.8			Study the adjacent systems	R & R
			5.4.8.1		Look for business opportunities for how product can help to achieve the product purpose within the product constraints	R & R
			5.4.8.2		Analyze dataflow between adjacent system and a process	R & R
		5.4.9			Define use case boundary for each business event	R & R
			5.4.9.1		Consider business opportunities	R & R
			5.4.9.2		Review the work knowledge	R & R
				5.4.9.2.1	Define the actor names	R & R
				5.4.9.2.2	Define the use case name	R & R
				5.4.9.2.3	Define the use case boundary data	R & R
				5.4.9.2.4	Record the product context by adding the use case to a use case diagram	R & R

Use case no.					Name	Source
				5.4.9.2.5	Keep track of business event name(s) that is/are related to this use case	R & R
	5.5				Do event reconnaissance	R & R
		5.5.1			Gather business event knowledge	R & R
			5.5.1.1		Look for business documents that might contain knowledge about work related to the event	R & R
			5.5.1.2		Look for any documents that might contain requirements buried in depth	R & R
			5.5.1.3		List the names of sources of the work context	R & R
			5.5.1.4		Determine if there is any domain models that contain knowledge about this event	R & R
			5.5.1.5		Determine if there is any reusable requirements that contain knowledge about this event	R & R
		5.5.2			Choose appropriate trawling techniques	R & R
	5.6				Ask clarification questions	R & R
	5.7				Evaluate results	RUP
6					Identify both external and internal interfaces	H & F
	6.1				Identify product interface	H & F
	6.2				Search for industry standard, application programmer's interface (API) or interface control document (ICD)	H & F
		6.2.1			Create ICD substitute if existing interface document is not found	H & F
	6.3				Monitor interface change outside control	H & F
	6.4				Obtain agreement from people from other side of external interface	H & F
	6.5				Simplify interfaces as much as possible	H & F
	6.6				Document product interfaces	H & F
	6.7				Distribute product interface documentation	H & F
	6.8				Track interface through development to ensure reality match documentation	H & F
7					Writing good requirements	H & F
	7.1				Identify potential requirements	R & R
	7.2				Identify functional requirements	R & R
	7.3				Identify composite requirements	R & R
	7.4				Formalize requirements	R & R
		7.4.1			Organize requirements into parent-child requirements	L & W
	7.5				Formalize system constraints	R & R
	7.6				Identify non-functional requirements	R & R
		7.6.1			Define usability	L & W

Use case no.					Name	Source
			7.6.1.1		Specify required training time for users to be marginally productive	L & W
			7.6.1.2		Specify measurable task times for typical tasks or transactions that end users will carry out	L & W
			7.6.1.3		Compare usability of the new system to other state-of-the-art systems that the user community knows and likes	L & W
			7.6.1.4		Specify existence and required features of online help systems, wizards, tool tips, user manuals, and other forms of documentation and assistance	L & W
			7.6.1.5		Follow conventions and standards that have been developed for the human-to-machine interface	L & W
		7.6.2			Define reliability	L & W
		7.6.3			Define performance	L & W
		7.6.4			Define supportability	L & W
	7.7				Write functional fit criteria	R & R
	7.8				Write non-functional fit criteria	R & R
	7.9				Define customer value	R & R
	7.10				Identify dependencies and conflicts	R & R
8					Capture rationale	H & F
9					Manage dependencies	RUP
	9.1				Assign attributes	RUP
	9.2				Establish levels	H & F
		9.2.1			Verify that requirement relate to level above	H & F
		9.2.2			Check if requirement allow more than one architecture or design option for the next level	H & F
		9.2.3			Check if requirement leads to solution - delete requirement if so	H & F
		9.2.4			Check if requirement is to be verified at this level	H & F
	9.3				Establish allocation (top down)	H & F
		9.3.1			Make sure that every requirement is allocated	H & F
		9.3.2			Check for duplicate requirements	H & F
		9.3.3			Check if requirements need to be allocated to more than one area	H & F
		9.3.4			Check if an interface is implied, simple and controllable	H & F
	9.4				Establish and verify traceability	RUP
		9.4.1			Make sure requirement tracing system is in place	H & F

Use case no.					Name	Source
		9.4.2			Make sure that every requirement can be traced back to a higher-level requirement	H & F
		9.4.3			Resolve duplication between levels	H & F
		9.4.4			Eliminate orphan requirements	H & F
	9.5a				Create a document tree	H & F
		9.5.1			Identify approval levels and segregate requirements accordingly	H & F
		9.5.2			Identify external contracts and segregate requirements that will be contractually binding to each outside party	H & F
		9.5.3			Segregate requirements for frequent revision	H & F
		9.5.4			Segregate requirements into manageable document sizes	H & F
	9.5b				Enter requirements in Modern Software Requirements Specifications (SRS) package	L & W
	9.6				Manage changing requirements	L & W
	9.7				Evaluate SRS	L & W
		9.7.1			Inspect quality of each individual specification	L & W
		9.7.2			Inspect quality for use-case model (use-case specifications, and use-case actors)	L & W
		9.7.3			Inspect quality for the entire Modern SRS	L & W
	9.8				Manage changing requirements	RUP
10					Verify requirements	H & F
	10.1				Screen requirements for subjective words	H & F
	10.2				Identify verficational stakeholders	H & F
	10.3				Decide what to verify and validate	L & W
		10.3.1a			Verify and validate everything	L & W
		10.3.1b			Use a hazard analysis to determine verify and validate necessities	L & W
	10.4				Decide how each requirement will be verified	L & W and H & F
		10.4.1			Compare to customer expectations	IEEE
		10.4.2			Compare to enterprise and project constraints	IEEE
		10.4.3			Compare to external constraints	IEEE
	10.5				Decide when each requirement will be verified	H & F
	10.6				Write requirements to cut time, cost, and special equipment required to verify products	H & F

Use case no.					Name	Source
	10.7				Decide how each requirement will be validated	L & W
		10.7.1			Perform acceptance testing	L & W
		10.7.2			Perform validation testing	L & W
		10.7.3			Perform validation traceability	L & W
		10.7.4			Perform requirements-based testing	L & W
	10.8				Establish validated requirements baseline	IEEE
	10.9				Build verification matrix	H & F
11					Format requirements	H & F
	11.1a				Organize requirements of complex hardware and software system	L & W
		11.1.1			Refine a system into subsystems	L & W
		11.1.2			Create requirements specification for each subsystem	L & W
		11.1.3			Refine subsystems into its subsystems (optional)	L & W
	11.1b				Organize requirements for product families	L & W
		11.1.1			Develop a product-family Vision Document	L & W
		11.1.2			Develop a set of use cases to show interactions among various applications	L & W
		11.1.3			Develop a common software requirements specification	L & W
		11.1.4			Develop a separate Vision Document, Software Requirements Specification, and a use case model for each product in the family	L & W
	11.2				Create Vision Document	L & W
	11.3				Create product position statement	L & W
	11.4				Circulate and gain agreement	L & W
	11.5				Create use cases in Vision Document (appendix)	L & W
	11.6				Publish Vision Document	L & W
	11.7				Assign owner to Vision Document (product champion)	L & W
	11.8				Utilize delta Vision Document	L & W
12a					Baseline requirements	H & F
	12.1				Find format, grammar, spelling , and typographical errors	H & F
	12.2				Look for ambiguities, unverified assumptions, unverified assumptions, TBDs, implementation, lack of rationale or unintelligible rationale, and lack of traceability	H & F
	12.3				Look for content errors, conflicts or missing requirements	H & F

Use case no.					Name	Source
	12.4				Assess product development risk	L & W and H & F
	12.5				Measure requirement quality	H & F
12b					Check requirements (quality gateway)	R & R
	12.1				Review requirements fit criteria	R & R
	12.2				Review requirements relevance	R & R
	12.3				Review requirement viability	R & R
	12.4				Identify gold-plated requirements	R & R
	12.5				Review requirements completeness	R & R
	12.6				Test requirements traceability	R & R
	12.7				Review requirements for consistent terminology	R & R
	12.8				Place customer rating on requirements	R & R
12c					Check requirements for certain properties	IEEE
13					Prioritize requirements	H & F
	13.1				Define priority classes	H & F
	13.2				Classify the requirements	H & F
		13.2.1			Assign 1's and 3's first - everything else default to 2	H & F
	13.3				Resolve the differences	H & F
	13.4				Create priority-based development schedules	H & F
	13.5				Maintain the priorities	H & F
14					Detail software requirements	RUP
	14.1				Collect software requirements artifacts	RUP
	14.2				Detail the software requirements	RUP
	14.3				Generate supporting reports	RUP
	14.4				Assemble the software requirements specification	RUP
15					Prioritize use case	RUP
	15.1				Prioritize use cases and scenarios	L & W and RUP
	15.2				Document the use-case view	L & W and RUP
	15.3				Evaluate results	L & W and RUP
16					Detail a use case	RUP
	16.1				Detail flow of events of the use case	RUP
	16.2				Structure the flow of events of the use case	RUP
	16.3				Illustrate relationships with actors and other use cases	RUP
	16.4				Describe special requirements of the use case	RUP
	16.5				Describe communication protocols	RUP

Use case no.					Name	Source
	16.6				Describe pre-conditions of the use case <optional>	RUP
	16.7				Describe post-conditions of the use case <optional>	RUP
	16.8				Describe extension points <optional>	RUP
	16.9				Evaluate results	RUP
17					Review change request	RUP
	17.1				Plan for changes to happen	L & W
	17.2				Baseline requirements	L & W
	17.3				Maintain responsibility for Vision Doc	L & W
	17.4				Schedule CCB review meeting	RUP
	17.5				Setup default reports and queries to assist in this effort	L & W
	17.6				Monitor SRS process	L & W
	17.7				Lead Change Control Review Board	L & W
	17.8				Retrieve change requests for review	RUP
		17.8.1			Submission of a new change request	RUP
		17.8.2			Update of an existing change request	RUP
		17.8.3			Consider postponing change request for a new release cycle	RUP
	17.9				Review submitted change requests	RUP
	17.10				Perform a thorough change impact assessment	H & F
	17.11				Use change control system to capture changes	L & W
	17.12				Make changes hierarchically	L & W
	17.13				Audit trail of history	L & W
18					Model the user interface	RUP
	18.1				Describe characteristics of related actors	RUP
	18.2				Create a use-case storyboard	RUP
	18.3				Describe flow of events - storyboard	RUP
	18.4				Capture usability requirements on the use-case storyboard	RUP
	18.5				Find boundary classes needed by the use-case storyboard	RUP
		18.5.1			Describe responsibility of boundary classes	RUP
		18.5.2			Describe attributes of boundary classes	RUP
		18.5.3			Describe relationships between boundary classes	RUP
		18.5.4			Present usability requirements on boundary classes	RUP
		18.5.5			Present the boundary classes in global class diagrams	RUP
		18.5.6			Evaluate results	RUP

Use case no.					Name	Source
	18.6				Describe interactions between boundary objects and actors	RUP
	18.7				Complement the diagrams of the use-case storyboard	RUP
	18.8				Refer to the user-interface prototype from the use-case storyboard	RUP
19					Prototype the user interface	RUP
	19.1				Plan the prototype	R & R
	19.2				Design the user-interface prototype	RUP
	19.3				Build prototype	R & R
		19.3.1			Build low fidelity prototype	R & R
		19.3.2			Build high fidelity prototype	R & R
	19.4				Evaluate the prototype	R & R
		19.4.1			Test high fidelity prototype with users	R & R
		19.4.2			Test low fidelity prototype with users	R & R
		19.4.3			Get feedback on user-interface prototype	RUP
		19.4.4			Identify new and changed requirements	R & R
		19.4.5			Evaluate prototyping effort	R & R
	19.5				Implement user-interface prototype	RUP
20					Structure use case model	RUP
	20.1				Establish include-relationships between use cases	RUP
	20.2				Establish extend-relationships between use cases	RUP
	20.3				Establish generalizations between use cases	RUP
	20.4				Establish generalizations between actors	RUP
	20.5				Evaluate results	RUP
21					Do requirements post mortem	R & R
	21.1				Gather input for review	R & R
		21.1.1			Conduct private individual reviews	R & R
		21.1.2			Conduct separate meetings with groups	R & R
		21.1.3			Facilitator reviews facts	R & R
	21.2				Do post mortem	R & R
		21.2.1			Hold post mortem review meeting	R & R
		21.2.2			Produce post mortem report	R & R
	21.3				Build a requirements filter	R & R
		21.3.1			Identify filtration criteria	R & R
		21.3.2			Select relevant requirement types	R & R
		21.3.3			Add new filtration criteria	R & R
22a					Review requirements	RUP
22b					Taking stock of the specification	R & R
	22.1				Review specification content	R & R
		22.1.1			Identify missing requirements	R & R
		22.1.2			Identify customer value ratings	R & R

Use case no.					Name	Source
		22.1.3			Identify requirement interaction	R & R
		22.1.4			Identify prototyping opportunity	R & R
		22.1.5			Find missing custodial requirements	R & R
	22.2				Evaluate requirements risk	R & R
		22.2.1			Look for likely risks	R & R
		22.2.2			Quantify each risk	R & R
	22.3				Estimate effort	R & R
		22.3.1			Identify estimation input	R & R
		22.3.2			Identify efforts for events	R & R
		22.3.3			Estimate requirements effort	R & R
	22.4				Publish reviewed specification	R & R
		22.4.1			Design form of specification	R & R
		22.4.2			Assemble the specification	R & R

APPENDIX B

TABULAR VIEW

					What			**Who**		**When**		**How (mechanism)**			**Source**	**Notes**
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
1					Capture a common vocabulary	Common terms are identified and documented	Glossary	System analyst	Customer, end user, and stakeholder		Vision, business case, business rules, business use-case model, business object model, stakeholder requests, use-case model, use case		RequisitePro		RUP	
1												IEEE Std 610.12-1990 (IEEE Standard Glossary of Software Engineering Terminology)			IEEE	
1												IEEE Std 830-1998 (IEEE Recommended Practice for Software Requirements Specifications)			IEEE	
1												IEEE Std 1233, 1998 edition (IEEE Guide for Developing System Requirements Specifications			IEEE	
	1.1				Find common terms	Terms describing business objects and real-world objects are identified		System analyst	Customer, end user, and stakeholder						RUP	
	1.2				Evaluate results			System analyst	Customer, end user, and stakeholder						RUP	
2					Develop requirements management plan	Attributes are identified and linked to tools	Requirements management plan	System analyst	Customer, end user, and stakeholder		-	Requirements management plan, important decisions in requirements	RequisitePro		RUP	
	2.1				Establish traceability			System analyst	Customer, end user, and stakeholder						RUP	
	2.2				Choose requirements attributes	Essential attributes (such as risk, benefit, effort, stability, and architectural impact) are identified		System analyst	Customer, end user, and stakeholder						RUP	
	2.3				Map to tools			System analyst	Customer, end user, and stakeholder				RationalRose, RequisitePro, Rational ClearQuest		RUP	
	2.4				Write the plan			System analyst	Customer, end user, and stakeholder			Requirements management plan			RUP	
3					Find actors and use cases	Actors and use cases are identified and documented	Use case models, actors, use cases, supplementary specifications	System analyst	Customer, end user, and stakeholder		Glossary, vision, stakeholder requests use-case modeling guidelines, business use-case model, business object model	Use-case workshop, storyboarding	Rational Rose		RUP	
	3.1				Establish scope of work	Business activity including actor, work, and adjacent systems are determined							Context diagram		R & R	
	3.2				Establish adjacent systems that surround the work by looking outside the organization										R & R	
	3.3				Identify connections between the work and the adjacent systems										R & R	
	3.4				Identify business events that added the work from the connections										R & R	
	3.5				Study the response to the event										R & R	
	3.6				Determine best response that the organization can make for the event										R & R	
	3.7				Determine product's role in the response										R & R	
	3.8				Determine the use case or cases							Jacobson, Ivar et al's book "Object-Oriented Software Engineering - A Use Case Driven Approach" [Addison-Wesley, 1992]			R & R	
		3.8.1			Find actors			System analyst	Customer, end user, and stakeholder						RUP	
		3.8.2			Find use cases			System analyst	Customer, end user, and stakeholder						RUP	
		3.8.3			Describe how actors and use cases interact			System analyst	Customer, end user, and stakeholder						RUP	
		3.8.4			Package use cases and actors			System analyst	Customer, end user, and stakeholder						RUP	
		3.8.5			Present the use-case model in the use-case diagrams			System analyst	Customer, end user, and stakeholder						RUP	
	3.9				Derive the requirements for each use case										R & R	
	3.10				Develop a survey of the use-case model			System analyst	Customer, end user, and stakeholder						RUP	
	3.11				Evaluate results			System analyst	Customer, end user, and stakeholder						RUP	
4a					Develop vision	Problem statement is formulated	Vision, initial requirements attributes, initial supplementary specifications	System analyst	Customer, end user, and stakeholder		Stakeholder requests, business rules, business use-case model, business object model	Brainstorming, fishbone diagrams, Pareto diagrams	RequisitePro		RUP	
	4.1				Gain agreement on the problem being solved	Definition of the problem is written and agreed upon		System analyst	Customer, end user, and stakeholder			Problem statement			L & W and RUP	
	4.2				Identify primary need	A short statement indicating motivation for the project						Table 4-4: Project scope sanity check			H & F	
	4.3				Understand root causes	Real problem and real cause are identified							Fishbone diagram		L & W	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
	4.4				Circulate problem statement										L & W	
	4.5				Revise where necessary										L & W	
	4.5				Review and obtain agreement				Customer, marketing, development, downstream organization			Table 4-4: Project scope sanity check			H & F	
	4.6				Identify stakeholders and users			System analyst	Customer, end user, and stakeholder						RUP and HHP	
	4.7				Obtain stakeholders' needs							Part of requirements gathering activity			HHP	
	4.8				Identify goals and objectives							Table 4-4: Project scope sanity check			H & F	
	4.9				Distribute and discuss goals and objectives with stakeholders	An aim and method for achieving target is discussed						Table 4-4: Project scope sanity check			H & F	
	4.10				Determine mission statement or business case (if any)							Table 4-4: Project scope sanity check			H & F	Business case is usually for commercial products
	4.11				Distribute it and gain consensus							Table 4-4: Project scope sanity check			H & F	
	4.12				Identify budgets							Table 4-4: Project scope sanity check			H & F	
	4.13				Identify schedule							Table 4-4: Project scope sanity check			H & F	
	4.14				Define solution system boundaries	Area containing solution system is identified	Actors, system	System analyst	Customer, end user, and stakeholder				Block diagram		L & W and RUP	
	4.14											Section 6.1.6 of IEEE Std 120-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
	4.15				Identify constraints to be imposed on the system	Restrictions on the system are identified	Constraints	System analyst	Customer, end user, and stakeholder			Table 4-4: Potential system constraints			L & W and RUP	
	4.16				Determine if work can be realistically done within budget and schedule constraints							Table 4-4: Project scope sanity check			H & F	
	4.17				Identify major assumptions							Table 4-4: Project scope sanity check			H & F	
	4.18				Validate assumptions							Table 4-4: Project scope sanity check			H & F	
	4.19				Assign responsibilities							Table 4-4: Project scope sanity check			H & F	
	4.2				Formulate problem statement			System analyst	Customer, end user, and stakeholder						RUP	
	4.21				Define features of the system			System analyst	Customer, end user, and stakeholder						RUP	
	4.22				Evaluate results			System analyst	Customer, end user, and stakeholder						RUP	
4b					Project blastoff	Necessary pieces required to begin the project and to ensure project is viable and well-founded	Purpose of the project, client, customer, stakeholders, users, constraints, names, relevant facts and assumptions, and scope of the work, estimated cost, risk, and go/no go decision	Facilitator	Blastoff team						R & R	
	4.1				Prepare for blastoff meeting		Blastoff meeting plan, required facilities			Project intention, potential stakeholders					R & R	
	4.1											Chapter 8: Making meetings work for everybody, chapter 13: Facilitating in the face of conflict			G & W	
		4.1.1			Define blastoff objectives	Deliverables are determined	Blastoff objectives, work context model, stakeholders identified, anticipated developers, system events event/use case models, system terminology, scenario models	Facilitator	Blastoff team	Project intention					R & R	
		4.1.2			Plan physical arrangements	Necessary physical arrangements are planned to produce blastoff objectives	Meeting location, meeting schedule, direction to meeting location, name and contact details of the facilitator, dates and times, estimated time required for blastoff, list of participants			Blastoff objectives					R & R	
			4.1.2.1		Determine participants	Potential stakeholders are determined									R & R	
			4.1.2.1									Chapter 7: Getting the right people involved			G & W	
			4.1.2.2		Plan facilities and accommodation for participants	Meeting places and accommodations are determined									R & R	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
		4.1.3			Communicate with participants		Blastoff meeting plan			Blastoff objectives, meeting schedule, meeting location, blastoff participants					R & R	
			4.1.3.1		Send each participant an agenda and list of participants	Participants must be aware of what they are going to do and that their participation is valuable									R & R	
	4.2				Run blastoff meeting		Major risks, blastoff meeting plan, project constraints, product purpose, business events, work context, system terminology, identified stakeholders			Potential stakeholders, project intention, stakeholder wants and needs, intended operating environment, blastoff meeting plan	Requirements skeleton				R & R	
		4.2.1			Determine product purpose	Statement of what product is at the end of the project	Product purpose, advantage, measure of success, reasonable, feasibility, achievable	Blastoff team		Stakeholder wants and needs, project intention, blastoff meeting plan					R & R	
		4.2.1										Chapter 14: Functions			G & W	
		4.2.2			Determine the work context	Intended work for study and surrounding systems are defined	Work context, context interfaces			Domains of interest, product purpose, stakeholder wants and needs	Requirements skeleton	James and Suzanne Robertson's book "Complete Systems Analysis - the Workbook, the Textbook, the Answers"			R & R	
			4.2.2.1		Ask if there is a physical entity that represents domain										R & R	
			4.2.2.2		Ask if domain provides data, policy or both to the work										R & R	
			4.2.2.3		Identify sources of information for this domain										R & R	
		4.2.3			Do first-cut risk analysis		Major risks				Requirements skeleton	Capers Jones' book "Assessment and Control of Software Risks"			R & R	
			4.2.3.1		Identify risks that are most likely to happen										R & R	
			4.2.3.2		Identify risks that would have the greatest impact of becoming a problem										R & R	
			4.2.3.3		Assess probability of risk becoming a problem										R & R	
			4.2.3.4		Assess its cost and schedule impact										R & R	
			4.2.3.5		Identify actions to take if risks come true										R & R	
		4.2.4			Identify the stakeholders	People who have an interest in the product is identified	Stakeholder name, specialization, estimated amount of involvement time			Potential stakeholders					R & R	Principal stakeholders include users, client and customers. Other stakeholders include the list on pages 36 - 38
			4.2.4.1		Inform stakeholders that they are stakeholders and that they will be consulted about requirements										R & R	
			4.2.4.2		Inform stakeholders of time required and type of participation										R & R	
		4.2.5			Partition the work	Work context is divided into business events	Business events			Stakeholder wants and needs, work context	Requirements skeleton				R & R	
		4.2.6			Consider non-events	"What-if" events are explored	New data flows are added to the work context diagram [work context, business events]			Work context and business events	Requirements skeleton				R & R	
		4.2.7			Determine system terminology	Common terms are identified and documented	System terminology			Context interfaces	Requirements skeleton				R & R	Similar to capture a common vocabulary

					What			**Who**		**When**		**How (mechanism)**			**Source**	**Notes**
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
		4.2.8			Define project constraints	Limitations on the way product is produced are identified	List of solution constraints, implementation environment constraints, partner application constraints, commercial off-the-shelf software constraints, anticipated workplace environment constraints, time constraints, and financial constraints			Stakeholder wants and needs, project intention, intended operating environment					R & R	
		4.2.8										Section 6.1.2 and 6.1.3 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
		4.2.8										Chapter 16: Constraint			G & W	
		4.2.9			Identify domains of interest	Areas of interest are identified	Domains of interest			Product purpose	Requirements skeleton				R & R	
	4.3				Finalize blastoff		System constraints, work context, business events, initial estimates, go/no go decision, blastoff report			Blastoff meeting plan, stakeholder wants and needs	Requirements skeleton, requirements template				R & R	
		4.3.1			Write blastoff report	Report of activities from the blastoff is written	Blastoff report, work context, business events, system constraints			Initial estimates	Requirements skeleton which consists of work context diagram, stakeholder list, manpower list, preliminary event or use case list, system terminology, major risks, initial estimates of effort, recommendation to proceed or not				R & R	
		4.3.2			Review blastoff results	Requirements skeleton is compared with requirements template	Go/no go decision, requirement questions			Blastoff meeting plan	Requirements skeleton, requirements template	Jim Hughsmith and Lynne Nix in "Feasibility Analysis - Mission Impossible" *Software Development*, July 1996			R & R	
		4.3.3			Hold follow-up blastoff	Outstanding requirements questions are answered	Requirements skeleton			Requirement questions, stakeholder wants and needs	Requirements skeleton				R & R	
		4.3.4			Make initial estimate	First estimate of effort is made									R & R	Allow generous area for learning curve
5a					Elicit stakeholder request		Stakeholder requests and use-case model	System analyst	Customer, end user, and stakeholder		Vision and change request	Requirements workshop, interviewing, brainstorming and idea reduction, storyboarding, role playing, review existing requirements	RequisitePro		RUP	
5a												Section 6.1.1 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
5b					Trawling for requirements	Requirements are found	List of requirements, some of which maybe not inappropriate	Requirements analyst	Users, customers, and clients						R & R	Inappropriate requirements will be weed out later
	5.1				Determine sources for requirements			System analyst	Customer, end user, and stakeholder						RUP	
	5.1									Customers, users, managers, industry standards, development process, and others					HHP	Sources of requirements
	5.2a				Gather information			System analyst	Customer, end user, and stakeholder						RUP	
	5.2a											Section 7.1.1 of IEEE Std 1233, 1998 edition (IEEE Guide for Developing System Requirements Specifications			IEEE	
	5.2b				Learn the work	Work is studied from user's point of view	Event for prototyping	Requirements analyst		Stakeholder wants and needs, work description and demonstration	Work knowledge				R & R	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
		5.2.1			Review current situation	The current situation where users face are examined	Current situation model	Requirements analyst		Work description and demonstration, stakeholder wants and needs	Work knowledge				R & R	
		5.2.2			Apprentice with the user	Analyst becomes an apprentice to the user - sits with user to learn the job by observing and asking questions	Model of the observed work [work knowledge]	Requirements analyst	Users	Stakeholder wants and needs	Work knowledge				R & R	
		5.2.3			Determine essential requirements	An abstract structure or pattern to the work is determined	Event for prototyping	Requirements analyst	Users	Current situation model, stakeholder wants and needs	Work knowledge	Observation and interpretation of users (skill and how they see themselves when they work) over a period of time			R & R	
		5.2.4			Brainstorm the requirements	Ideas for requirements are brainstormed	List of requirements (unedited)	Requirements analyst		Stakeholder wants and needs	Work knowledge				R & R	
		5.2.5			Create structured interviews	Context-free questions are created based on a template						Figure 9-1: The Generic, Almost Context-Free interview			L & W	Use context-free questions (i.e. ask about nature of problem and not solution). Questionnaires does not substitute interviews!
		5.2.5										Chapter 6: Context-free questions			G & W	
		5.2.6			Conduct 5 to 15 interviews										L & W	
		5.2.7			Summarize interviews										L & W	R & R recommends using interviews with other techniques
		5.2.8			Do document archeology	Documents and files that the organization currently uses are inspected	System terminology + data models	Requirements analyst		Business documents	Work knowledge	Questions on page 100 of R & R			R & R	R & R recommends using this technique with other techniques
		5.2.9			Make requirements video	Video recording of brainstorm, workshops, interviews, observations, etc. can be effectively used as a recording tool (information and body languages)	Event for prototyping	Requirements analyst		Stakeholder wants and needs	Work knowledge				R & R	
		5.2.10			Run use case workshop		Event for prototyping	Requirements analyst	Appropriate customer/user	Essential steps that take place in an event	Work knowledge				R & R	
		5.2.11			Build event models	The whole system is broken up into events	Models of events [work knowledge]	Requirements analyst		Stakeholder wants and needs	Work knowledge	Data flows between adjacent systems and work context as a result of temporal event			R & R	
		5.2.12			Build scenario models	Models of the way users operate an intended system is recorded	Scenario models	Requirements analyst	Users	Stakeholder wants and needs	Work knowledge		Any format and medium that the user is comfortable with		R & R	
		5.2.12										Section 6.1.12 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
			5.2.12.1		Define technical performance measures (TPMs)	Key indicators of system performance are identified						Section 6.1.13 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
			5.2.12.2		Define design characteristics	Design characteristics (such as color, texture, size, anthropomorphic limitations, weight, and buoyancy) are identified and defined						Section 6.1.14 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
			5.2.12.3		Define human factors	Human factor considerations (such as design space limits, climatic limits, eye movement, reach, ergonomics, cognitive limits, and usability) affecting operation of products are identified and examined						Section 6.1.15 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
		5.2.13			Run requirements workshop							Chapter 10: Requirements workshop			L & W and RUP	
		5.2.14			Brainstorming							Chapter 11: Brainstorming and idea reduction			L & W	
		5.2.14										Chapter 10: Idea generation meetings			G & W	
		5.2.15			Mind map requirements	Representation of requirements in drawing and text									R & R	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
		5.2.16			Collect requirements via Volere Snow Cards	Pre-printed cards filled out as information becomes available									R & R	Sample of Snow Card is on page 102
		5.2.17			Reduce ideas							Chapter 11: Brainstorming and idea reduction			L & W	
			5.2.17.1		Pruning							Chapter 11: Brainstorming and idea reduction			L & W	
			5.2.17.2		Grouping ideas							Chapter 11: Brainstorming and idea reduction			L & W	
			5.2.17.3		Feature definition							Chapter 11: Brainstorming and idea reduction			L & W	
			5.2.17.4		Prioritization							Chapter 11: Brainstorming and idea reduction			L & W	
		5.2.18			Create storyboards for innovative concepts							Chapter 12: Storyboarding			L & W	
		5.2.19			Create operational concepts	Operation of the product is imagined and documented in user language						Table 5-1: Operational concepts completeness sanity check			H & F	Approach depends on whether you are product developer or product procurer. Software developers call them 'use cases'; space-craft developers - 'operation plans' or 'design reference mission'; people simply know them as 'scenarios'.
		5.2.19										Section 6.1.4 and 6.1.8 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
			5.2.19.1		Develop concept for each phase of the lifecycle							Table 5-1: Operational concepts completeness sanity check			H & F	
				5.2.19.1.1	Outline normal operation and environment							Table 5-1: Operational concepts completeness sanity check			H & F	
				5.2.19.1.2	Outline abnormal operation and environment							Table 5-1: Operational concepts completeness sanity check			H & F	
			5.2.19.2		Consider viewpoints of all stakeholders							Table 5-1: Operational concepts completeness sanity check			H & F	
			5.2.19.3		Assess human interface standard							Table 5-1: Operational concepts completeness sanity check			H & F	
			5.2.19.4		Create use cases							Chapter 13: Applying use cases			L & W	
		5.2.20			Role play							Chapter 14: Role playing			L & W	Similar techniques include scripted walkthroughs and Class-Responsibility-Collaboration (CRC) cards
		5.2.21			Create prototypes							Chapter 15: Prototyping			L & W	
	5.3				Define life cycle process concepts	Life cycle process requirements are determined to develop, produce, test, distribute, operate, support, train, and dispose of products under development	Life cycle process requirements			Section 6.1.1 through 6.1.8 of IEEE Std 1220-1998		Section 6.1.9 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
	5.4				Determine product scope		Use case (to be used in product scope)	Requirements analyst		Work context, system constraints, stakeholder wants and needs	Work knowledge				R & R	
		5.4.1			Set priorities for each feature										L & W	
		5.4.2			Assess effort for each feature										L & W	
		5.4.3			Estimate risk for each feature										L & W	
		5.4.4			Reduce scope based on priorities, effort, and risk										L & W	
		5.4.5			Determine baseline for each release of Vision Document								Version number		L & W	
		5.4.6			Get customer agreement on scope							Guiding principle for scope management: "Underpromise and overdeliver" (page 209)			L & W	
		5.4.7			Advocate and practice iterative development										L & W	
		5.4.8			Study the adjacent systems	Event-response model is used as learning tool	Business event boundary + business opportunities			Business event boundary, system constraints, work context	Work knowledge				R & R	

					What			**Who**		**When**		**How (mechanism)**			**Source**	**Notes**
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
			5.4.8.1		Look for business opportunities for how product can help to achieve the product purpose within the product constraints										R & R	
			5.4.8.2		Analyze dataflow between adjacent system and a process							Questions on page 302 in R & R			R & R	
		5.4.9			Define use case boundary for each business event		{Actor name} + use case name + use case boundary data + {business event name} {this leads to product scope]			Stakeholder wants and needs, business event boundary + business opportunities					R & R	R & R recommends using a leveled use case diagram if there are more than 15-20 use cases
			5.4.9.1		Consider business opportunities										R & R	
			5.4.9.2		Review the work knowledge										R & R	
				5.4.9.2.1	Define the actor names										R & R	
				5.4.9.2.2	Define the use case name										R & R	
				5.4.9.2.3	Define the use case boundary data										R & R	
				5.4.9.2.4	Record the product context by adding the use case to a use case diagram										R & R	
				5.4.9.2.5	Keep track of business event name(s) that is/are related to this use case										R & R	
	5.5				Do event reconnaissance		Business documents, business event boundary + knowledge sources + trawling techniques	Requirements analyst		Business events, work description and demonstration, reusable requirements, domain models, work context	Reuse library, work knowledge				R & R	
		5.5.1			Gather business event knowledge		Business documents, business event boundary + knowledge sources			Work description and demonstration, business events, work context, domain models, reusable requirement, business documents	Reuse library, work knowledge				R & R	
			5.5.1.1		Look for business documents that might contain knowledge about work related to the event										R & R	
			5.5.1.2		Look for any documents that might contain requirements buried in depth										R & R	
			5.5.1.3		List the names of sources of the work context										R & R	
			5.5.1.4		Determine if there is any domain models that contain knowledge about this event										R & R	
			5.5.1.5		Determine if there is any reusable requirements that contain knowledge about this event										R & R	
		5.5.2			Choose appropriate trawling techniques	Considerations are made on the appropriate trawling techniques	Business event boundary + knowledge + trawling techniques			Business event boundary + knowledge sources	Work knowledge	Considerations and guidelines are found on page 304 and 305 in R & R			R & R	
	5.6				Ask clarification questions	Requirement questions and system constraint questions are reviewed	Work knowledge	Requirements analyst		Stakeholder wants and needs, system constraint questions, requirement questions	Work knowledge	Requirements template			R & R	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
	5.7				Evaluate results			System analyst	Customer, end user, and stakeholder						RUP	
6					Identify both external and internal interfaces	Animate or live user and inanimate external users are identified to clarify scope, aid risk assessment, reduce development costs, and improve customer satisfaction.						Table 6-4: Product interface identification sanity check			H & F	When developing new product, the matrix may be noted for future investigation until the product is in design.
6												Section 6.1.7 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
	6.1				Identify product interface							Table 6-1: Checklist for individual interface exploration			H & F	
	6.2				Search for industry standard, application programmer's interface (API) or interface control document (ICD)	Interface requirements that product must meet are found						Table 6-4: Product interface identification sanity check			H & F	
		6.2.1			Create ICD substitute if existing interface document is not found							Table 6-4: Product interface identification sanity check			H & F	
	6.3				Monitor interface change outside control	Changes from outside sources are monitored for risk assessment purposes						Table 6-4: Product interface identification sanity check			H & F	
	6.4				Obtain agreement from people from other side of external interface	Interface documentation are agreed upon and documented accordingly	Interface requirement specification (IRS) or interface requirement document (IRD)					Table 6-4: Product interface identification sanity check			H & F	
	6.5				Simplify interfaces as much as possible							Table 6-4: Product interface identification sanity check			H & F	
	6.6				Document product interfaces	Product interfaces (both internal and external) are documented						Table 6-4: Product interface identification sanity check			H & F	
	6.7				Distribute product interface documentation							Table 6-4: Product interface identification sanity check			H & F	
	6.8				Track interface through development to ensure reality match documentation							Table 6-4: Product interface identification sanity check			H & F	
7					Writing good requirements	Requirements are put into simple and specific statements	Clear, verifiable, and attainable needs expressed in requirements					Chapter 6 by Hooks and Farry, table 7-4: Individual requirement sanity check, "Getting it right the first time - writing better requirements" by Quality Systems and Software, "Writing Good Requirements" by Ivy Hooks, "Characteristics of Good Requirements" by Pradip Kar and Michelle Bailey.			H & F	Attempting to write requirements before defining scope, operational concepts, and interface can lead to inconsistent and incomplete requirements.
7												Section 6 of IEEE Std 1233, 1998 edition (IEEE Guide for Developing System Requirements Specifications			IEEE	
	7.1				Identify potential requirements	Potential requirements are recorded	Requirements in the form of "The product shall…" along with sources, rationale and associated use case (I.e. requirements)	Requirements analyst		Potential requirements from trawling process	Product scope, work knowledge				R & R	
	7.2				Identify functional requirements	Real work (independent of how work will be carried out) are identified.	Functional requirements in the form of "The product shall…" along with sources, rationale and associated use case	Requirements analyst		Actor's task in use cases	Requirements template, work knowledge	Functional requirements are characterized by verbs	Use cases	Appendix B	R & R	Sources of requirements include any artifact that describes products' actions
	7.2											Section 6.1.10 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
	7.2											Also known as required capabilities			HHP	
	7.3				Identify composite requirements	Requirements that does not have its own testable fi criteria are identified	Composite requirements for each use case, summarizing several testable individual requirements, along with rationale (a.k.a. high level requirements)	Requirements analyst		Requirements, functional requirements	Work knowledge, product scope				R & R	
	7.4				Formalize requirements	Requirements are recorded into a formal requirements template	Collection of filled-out Volere shell cards and Volere Requirements Specification Template (sections: functional requirements and non-functional requirements) [formalized requirements]	Requirements analyst		Requirements, functional requirements, composite requirements	Work knowledge, requirements template		Requirements shell	Appendix B	R & R	
		7.4.1			Organize requirements into parent-child requirements	Requirements are organized hierarchically for increased specificity									L & W	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
	7.5				Formalize system constraints	System constraints are recorded into the Volere Requirements Specification Template	Formalized system constraint	Requirements analyst		System constraints, business events	Requirements template, work knowledge			Appendix B	R & R	
	7.5											Also known as required constraints or design constraints			HHP and L & W	
	7.6				Identify non-functional requirements	Characteristics or qualities that product must have to perform what it must do are identified	Properties that product must have to support functional requirements [non-functional requirements]	Requirements analyst		Functional requirements + use case	Requirements template, work knowledge	Non-functional requirements are characterized by adjectives, non-functional requirement types checklist, chapter 7 of R & R	Prototypes	Appendix B	R & R	Non functional requirement types include: look and feel, usability, performance, operational, maintainability, security, cultural and political, and legal
		7.6.1			Define usability	To-be users' knowledge about the new system has to be considered						"User's Bill of Rights" (page 239)			L & W	
			7.6.1.1		Specify required training time for users to be marginally productive										L & W	
			7.6.1.2		Specify measurable task times for typical tasks or transactions that end users will carry out										L & W	
			7.6.1.3		Compare usability of the new system to other state-of-the-art systems that the user community knows and likes										L & W	
			7.6.1.4		Specify existence and required features of online help systems, wizards, tool tips, user manuals, and other forms of documentation and assistance										L & W	
			7.6.1.5		Follow conventions and standards that have been developed for the human-to-machine interface										L & W	
		7.6.2			Define reliability	Issues such as availability, mean time between failures (MTBF), mean time to repair (MTTR), accuracy, defect rate, and bugs per type are considered									L & W	
		7.6.3			Define performance	Response time, throughput, capacity, and degradation modes are considered									L & W	
		7.6.4			Define supportability	Issues such as enhancements and repairs are considered									L & W	
	7.7				Write functional fit criteria	Criteria for knowing whether solution meets functional requirements are set	A functional criteria for each functional requirement (recorded in the Volere Requirements Specification Template	Requirements analyst	Client, testers	Functional requirements, scale of measurement, requirements	Work knowledge			Appendix B	R & R	
	7.7											Section 6.1.11 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
	7.7											Also known as performance requirement			HHP	Performance requirement must be coupled with each required constraints and required capabilities
	7.8				Write non-functional fit criteria	Criteria for knowing whether solution meets non-functional requirements are set	A non-functional criteria for each non functional requirement (recorded in the Volere Requirements Specification Template	Requirements analyst	Client, testers	Non-functional requirements and scale of measurement, requirements	Work knowledge			Appendix B	R & R	
	7.9				Define customer value	Customer satisfaction and dissatisfaction values are discovered	Understanding between team and client on clients' priorities and basis for making choices about which/when/whether to implement requirements	Requirements analyst	Client	Clients satisfaction and dissatisfaction values, requirements	Work knowledge			Appendix B	R & R	
	7.9											Section 6.1.5 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
	7.9											Chapter 21: Measuring satisfaction			G & W	

					What			**Who**		**When**		**How (mechanism)**			**Source**	**Notes**
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
	7.10				Identify dependencies and conflicts	Conflicting requirements are recorded	Conflicting requirements	Requirements analyst		Requirements	Work knowledge			Appendix B	R & R	
8					Capture rationale	Explanations why requirements exist, assumptions made, relevant findings of design studies, and other useful information are recorded.	Reasons, assumptions, operational relationships, and design decisions supporting each requirement			Requirements		Table 8-1: Requirement rational satiny check			H & F	
9					Manage dependencies	Attributes are assigned, traceability established and verified	Updated requirements attributes, updated requirements management plan, updated vision	System analyst	Customer, end user, and stakeholder		Requirement management plan, requirements attributes, vision, change requests, use-case model, supplementary specifications, design model, test model, risk list, stakeholder requests		RequisitePro		RUP	
	9.1				Assign attributes			System analyst	Customer, end user, and stakeholder						RUP	
	9.2				Establish levels	Requirement levels are identified to keep the big picture in mind, decrease development problems, and prevent administrative gridlock	Updated requirements with different levels, each level defining what the each level must do					Table 9-1: Requirement levels sanity check			H & F	
		9.2.1			Verify that requirement relate to level above							Table 9-1: Requirement levels sanity check			H & F	
		9.2.2			Check if requirement allow more than one architecture or design option for the next level							Table 9-1: Requirement levels sanity check			H & F	
		9.2.3			Check if requirement leads to solution - delete requirement if so							Table 9-1: Requirement levels sanity check			H & F	
		9.2.4			Check if requirement is to be verified at this level							Table 9-1: Requirement levels sanity check			H & F	
	9.3				Establish allocation (top down)	Systems-level requirements are matched to part(s) that must accomplish the requirement	Requirements are matched with part requirements			Systems-level requirements		Table 9-2: Requirement allocation sanity check			H & F	
		9.3.1			Make sure that every requirement is allocated							Table 9-2: Requirement allocation sanity check			H & F	
		9.3.2			Check for duplicate requirements							Table 9-2: Requirement allocation sanity check			H & F	
		9.3.3			Check if requirements need to be allocated to more than one area							Table 9-2: Requirement allocation sanity check			H & F	
		9.3.4			Check if an interface is implied, simple and controllable							Table 9-2: Requirement allocation sanity check			H & F	
	9.4				Establish and verify traceability	Each requirement is checked to ensure that it came from a parent requirement at system level		System analyst	Customer, end user, and stakeholder			Table 9-3: Requirement tracing sanity check			RUP	
		9.4.1			Make sure requirement tracing system is in place							Table 9-3: Requirement tracing sanity check			H & F	
		9.4.2			Make sure that every requirement can be traced back to a higher-level requirement							Table 9-3: Requirement tracing sanity check			H & F	
		9.4.3			Resolve duplication between levels							Table 9-3: Requirement tracing sanity check			H & F	
		9.4.4			Eliminate orphan requirements							Table 9-3: Requirement tracing sanity check			H & F	Orphan requirements may signal from top-level requirements are missing
	9.5a				Create a document tree	Requirements are recorded in a document tree structure requirements specification	Document tree structure requirements specification					Table 9-4: Document tree sanity check			H & F	Document tree helps structure requirements
		9.5.1			Identify approval levels and segregate requirements accordingly							Table 9-4: Document tree sanity check			H & F	
		9.5.2			Identify external contracts and segregate requirements that will be contractually binding to each outside party							Table 9-4: Document tree sanity check			H & F	
		9.5.3			Segregate requirements for frequent revision							Table 9-4: Document tree sanity check			H & F	
		9.5.4			Segregate requirements into manageable document sizes							Table 9-4: Document tree sanity check			H & F	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
	9.5b				Enter requirements in Modern Software Requirements Specifications (SRS) package	A collection of artifacts describing the complete external behavior of the system is documented		Development team		Vision Document		Appendix C: Modern SRS Package Template	Technical approach methods include: pseudocode, finite state machines, decision trees, activity diagrams, entity relationship models, object-oriented analysis, and structured analysis		L & W	
	9.6				Manage changing requirements										L & W	
	9.7				Evaluate SRS							Chapter 27: Quality measures of software requirements			L & W	
		9.7.1			Inspect quality of each individual specification	The following qualities are checked: correct, unambiguous, complete, consistent, ranked for importance and stability, verifiable, modifiable, traceable, and understandable.						Chapter 27: Quality measures of software requirements			L & W	
		9.7.2			Inspect quality for use-case model (use-case specifications, and use-case actors)							Books by Booch (1999) and Jacobson, Booch, and Rumbaugh (1999) and chapter 27: Quality measures of software requirements			L & W	
		9.7.3			Inspect quality for the entire Modern SRS	Modern SRS package that has a good Table of Contents, index, revision history, and glossary						Chapter 27: Quality measures of software requirements			L & W	
	9.8				Manage changing requirements			System analyst	Customer, end user, and stakeholder			Reassess requirements attributes and traceability, manage change hierarchically			RUP	
10					Verify requirements	Requirements are checked to make sure that they support verification	Updated requirements which are verifiable					Table 10-3: Verification assessment sanity check			H & F	
10												Section 6.2 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
10												Traceability	RequisitePro		L & W	Verification = make sure that you are doing the right thing
	10.1				Screen requirements for subjective words							Table 10-1: Certain words flag unverifiable requirements,			H & F	
	10.2				Identify verificational stakeholders							Table 10-3: Verification assessment sanity check			H & F	
	10.3				Decide what to verify and validate										L & W	
		10.3.1a			Verify and validate everything										L & W	
		10.3.1b			Use a hazard analysis to determine verify and validate necessities										L & W	
	10.4				Decide how each requirement will be verified	Requirements can be verified via inspection, test, demonstration, and analysis						Table 10-3: Verification assessment sanity check from H & F			L & W and H & F	
		10.4.1			Compare to customer expectations	Requirements are checked against customer expectation to ensure they represent customers' needs, requirements, and constraints			End-user, marketing, etc.	Requirements provided by customers		Section 6.2.1 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
		10.4.2			Compare to enterprise and project constraints	Requirements are checked against enterprise and project constraints. This is to ensure correct representation and that requirements stay within enterprise and project policies and procedures, acceptable risk levels, plans, resources, technology limitations, objectives, decisions, standards, and other constraints.						Section 6.2.2 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
		10.4.3			Compare to external constraints	Requirements are checked against external constraints. This would include national and international laws; external interface requirements with existing or evolving requirements, platforms, or products; applicable general specification and standard provisions; and competitive product capabilities and characteristics						Section 6.2.3 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
	10.5				Decide when each requirement will be verified							Table 10-3: Verification assessment sanity check			H & F	
	10.6				Write requirements to cut time, cost, and special equipment required to verify products							Table 10-3: Verification assessment sanity check			H & F	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
	10.7				Decide how each requirement will be validated										L & W	Validation = make sure that the system is doing what's supposed to do
		10.7.1			Perform acceptance testing										L & W	
		10.7.2			Perform validation testing										L & W	
		10.7.3			Perform validation traceability										L & W	
		10.7.4			Perform requirements-based testing										L & W	
	10.8				Establish validated requirements baseline							Section 6.2.5 of IEEE Std 1220 - 1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
	10.9				Build verification matrix							Table 10-3: Verification assessment sanity check			H & F	
11					Format requirements	Requirements are organized into a standard format	Well-organized requirements			List of requirements		Table 11-1: Items your specification may need to cover, table 11-2: specification standards and sources, table 11-3: Requirement document format sanity check			H & F	Requirements can be organized based on operational concepts, major functions, etc..
11												Section 7.3 of IEEE Std 1233, 1998 edition (IEEE Guide for Developing System Requirements Specifications			IEEE	
	11.1a				Organize requirements of complex hardware and software system	Requirements are organized and documented in a requirements specification	Hierarchy of specifications								L & W	
		11.1.1			Refine a system into subsystems		Partitions and allocations between subsystems						Systems engineering		L & W	
		11.1.2			Create requirements specification for each subsystem	External behavior of the system is described									L & W	
		11.1.3			Refine subsystems into its subsystems (optional)										L & W	
	11.1b				Organize requirements for product families		Requirements organization for a software product family								L & W	
		11.1.1			Develop a product-family Vision Document										L & W	
		11.1.2			Develop a set of use cases to show interactions among various applications										L & W	
		11.1.3			Develop a common software requirements specification	Specific requirements for shared functionality are defined									L & W	
		11.1.4			Develop a separate Vision Document, Software Requirements Specification, and a use case model for each product in the family										L & W	
	11.2				Create Vision Document	A high level abstraction of problem and solution is documented in a Vision Document						Figure 7-1: Template for software product Vision Document			L & W	
	11.3				Create product position statement										L & W	
	11.4				Circulate and gain agreement										L & W	
	11.5				Create use cases in Vision Document (appendix)										L & W	
	11.6				Publish Vision Document										L & W	
	11.7				Assign owner to Vision Document (product champion)	A person or a small team is assigned to maintain the project vision						Chapter 18: The champion			L & W	
	11.8				Utilize delta Vision Document	Changes and updates are recorded in the delta Vision Document									L & W	
12a					Baseline requirements	Requirements are considered completed at this point and are ready for design	"Cleaned" set of requirements			Requirements					H & F	
12a												Section 6.1.16 of IEEE Std 1220-1998 (IEEE Standard for Application and Management of the Systems Engineering Process)			IEEE	
	12.1				Find format, grammar, spelling , and typographical errors	Requirements are checked for typos	"Redlined" requirements	Elected editor		Requirements		Table 12-1: Editorial sanity check			H & F	
	12.2				Look for ambiguities, unverified assumptions, unverified assumptions, TBD, implementation, lack of rationale or unintelligible rationale, and lack of traceability	Requirements are examined for obvious problems		Requirement engineers or elected requirement writer		Requirements		Table 12-2: Requirement "goodness" sanity check			H & F	Assumed TBD = to be determined
	12.2											Chapter 2: Ambiguity in stating requirements, chapter 3: Sources of ambiguity, chapter 9: Reducing ambiguity from start to finish			G & W	
	12.3				Look for content errors, conflicts or missing requirements	Requirements are examined for content	Recommendations (and reasons) for each requirement	Selected reviewers from stakeholders		Requirements	Operational concepts	Table 12-3: Requirement content sanity check			H & F	
	12.4				Assess product development risk							Table 12-4: Risk assessment sanity check			L & W and H & F	Risks may surface from requirement volatility, technical feasibility, budget, and schedule
	12.5				Measure requirement quality	Quality of the requirements are examined for rooms for improvements	Analyzed data on requirements			Data on requirements	Requirement count, baseline review redlines, discrepancy analysis, change analysis	Table 16-1: Measuring requirement quality sanity check			H & F	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
12b					Check requirements (quality gateway)	Requirements are checked for completeness, traceability, consistency, relevancy, correctness, ambiguity, being solution-bound, gold-plating, and creep to avoid requirements creep and requirements leakage	Accepted requirements, excluded requirements to be sent back for revision or omitted completely	Requirements analyst		Formalized requirements				Appendix B	R & R	Who does Quality Gateway is determined by the organization's culture
	12.1				Review requirements fit criteria	Communicable limits are set so that they can be tested	Rejected requirement, requirement questions, fit reviewed requirement	Requirements analyst	Testers	Formalized requirements, formalized system constraint	Requirements template, product scope, work knowledge			Appendix B	R & R	
	12.2				Review requirements relevance	Requirements are checked to make sure that they are within product context and also that they are not solutions	Rejected requirement, system constraint questions, requirement questions, accepted system constraint, relevance reviewed requirement	Requirements analyst		Completeness reviewed requirement	Requirements template, product scope, work knowledge, requirements specification			Appendix B	R & R	Abstract requirements are usually not solutions
	12.3				Review requirement viability	Requirements are checked to make sure that they are workable within the project	Rejected requirements, requirement questions, viability reviewed requirement	Requirements analyst		Formalized requirements	Requirements template, product scope, work knowledge, requirements specification			Appendix B	R & R	
	12.4				Identify gold-plated requirements	Requirements are checked to make sure that they are absolutely necessary for the project	Gold-plated requirements are omitted (if not, gold-plated ones are flagged), requirement questions, accepted requirement	Requirements analyst		Strategic plan for product, viability reviewed requirement	Requirements specification			Appendix B	R & R	Gold-plated requirements maybe kept for political or personality reasons
	12.5				Review requirements completeness	Requirements are checked to make sure that they are complete	Requirements with all required components filled out	Requirements analyst	Stakeholders	Formalized requirements			Volere shell	Appendix B	R & R	
	12.6				Test requirements traceability	Requirements are checked to make sure that there is a connection with deliverables	Traceable requirements (complete with unique identifier, indicator of type of requirement or constraint, references to all business events and use cases, references to dependent requirements, references to other requirements, and consistent use of terminology)	Requirements analyst		Formalized requirements					R & R	
	12.7				Review requirements for consistent terminology	Requirements are checked to make sure that each is understood by all in the same way	Clear and unmistakable requirements	Requirements analyst		Formalized requirements				Appendix A	R & R	
	12.8				Place customer rating on requirements	Requirements are checked to make sure that they are of some importance	Weighted requirements	Requirements analyst	Client, customers, stakeholders	Formalized requirements					R & R	QED can be substituted for this step
12c					Check requirements for certain properties	Requirements are checked to ensure that they are unique, normalized, linked, complete, consistent, bounded, modifiable, configurable, and granular.	Complete requirements					Section 4.2 and 6..2 of IEEE Std 1233, 1998 edition (IEEE Guide for Developing System Requirements Specifications			IEEE	
13					Prioritize requirements	Requirements are grouped based on relative importance						Table 13-1: Prioritizing requirements sanity check			H & F	
	13.1				Define priority classes	Priority numbering is decided					Essential, nonnegotiable, and urgent requirements : 1; useful, slightly deferrable requirements: 2; merely desirable, flexible, or "someday" requirements: 3	Table 13-1: Prioritizing requirements sanity check			H & F	
	13.2				Classify the requirements	Requirements are classified by priorities						Table 13-1: Prioritizing requirements sanity check			H & F	Easier to classify most important ones and least important ones…all the rest are in between
		13.2.1			Assign 1's and 3's first - everything else default to 2							Table 13-1: Prioritizing requirements sanity check			H & F	
	13.3				Resolve the differences	Agreement on priority is granted						Table 13-1: Prioritizing requirements sanity check			H & F	
	13.4				Create priority-based development schedules	Timelines for each requirement is created						Table 13-1: Prioritizing requirements sanity check			H & F	
	13.5				Maintain the priorities	Priorities are checked often to assure that they are being followed						Table 13-1: Prioritizing requirements sanity check			H & F	
14					Detail software requirements		Updated requirement attributes, detailed supplementary specifications, software requirements specification	Requirements specifier			Vision, glossary, use case model, use case supplementary specifications, requirements attributes, requirement management plan, user-interface prototype		SoDa		RUP	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
	14.1				Collect software requirements artifacts			Requirements specifier							RUP	
	14.2				Detail the software requirements			Requirements specifier							RUP	
	14.3				Generate supporting reports			Requirements specifier							RUP	
	14.4				Assemble the software requirements specification			Requirements specifier							RUP	
15					Prioritize use case	Use cases are prioritized and documented	Updated requirements attributes, software architecture document, refined glossary	Software architect	-		Vision, use case model, requirements, attributes, iteration plan, glossary				RUP	
	15.1				Prioritize use cases and scenarios			Software architect							L & W and RUP	
	15.2				Document the use-case view			Software architect							L & W and RUP	
	15.3				Evaluate results			Software architect							L & W and RUP	
16					Detail a use case	Use cases are detailed by describing special requirements, communication protocols, pre-conditions, post-conditions, and extension points	Use case, updated supplementary specifications, requirements attributes	Requirements specifier			Vision, stakeholder requests, glossary use case, use case model, supplementary specifications, use-case modeling guidelines, requirements management plan		RequisitePro, RationalRose		RUP	
	16.1				Detail flow of events of the use case			Requirements specifier							RUP	
	16.2				Structure the flow of events of the use case			Requirements specifier							RUP	
	16.3				Illustrate relationships with actors and other use cases			Requirements specifier							RUP	
	16.4				Describe special requirements of the use case			Requirements specifier							RUP	
	16.5				Describe communication protocols			Requirements specifier							RUP	
	16.6				Describe pre-conditions of the use case <optional>			Requirements specifier							RUP	
	16.7				Describe post-conditions of the use case <optional>			Requirements specifier							RUP	
	16.8				Describe extension points <optional>			Requirements specifier							RUP	
	16.9				Evaluate results			Requirements specifier							RUP	
17					Review change request	Requests for change are evaluated	Updated change request	Change control manager	Change control board		Change request		ClearQuest		RUP	
	17.1				Plan for changes to happen	Allowance for inevitable and necessary changes are considered	Plan for managing changes								L & W	
	17.2				Baseline requirements	A version number is assigned to requirements	Old and new requirements are distinguished, making new requirements more manageable								L & W	
	17.3				Maintain responsibility for Vision Doc										L & W	Small project: product champion; large project: change control board
	17.4				Schedule CCB review meeting			Change control manager	Change control board						RUP	
	17.5				Setup default reports and queries to assist in this effort										L & W	
	17.6				Monitor SRS process										L & W	
	17.7				Lead Change Control Review Board										L & W	
	17.8				Retrieve change requests for review			Change control manager	Change control board						RUP	
		17.8.1			Submission of a new change request										RUP	
		17.8.2			Update of an existing change request										RUP	
		17.8.3			Consider postponing change request for a new release cycle										RUP	
	17.9				Review submitted change requests			Change control manager	Change control board						RUP	
	17.10				Perform a thorough change impact assessment										H & F	
	17.11				Use change control system to capture changes										L & W	
	17.12				Make changes hierarchically										L & W	
	17.13				Audit trail of history										L & W	
18					Model the user interface		Refined use case storyboards, refined actors, boundary class	User-interface designer			Use case, actors, supplementary specifications, vision, stakeholder requests, user-interface guidelines				RUP	
	18.1				Describe characteristics of related actors			User-interface designer							RUP	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
	18.2				Create a use-case storyboard			User-interface designer							RUP	Steps can be alternated or performed in parallel.
	18.3				Describe flow of events - storyboard			User-interface designer							RUP	
	18.4				Capture usability requirements on the use-case storyboard			User-interface designer							RUP	
	18.5				Find boundary classes needed by the use-case storyboard			User-interface designer							RUP	
		18.5.1			Describe responsibility of boundary classes			User-interface designer							RUP	
		18.5.2			Describe attributes of boundary classes			User-interface designer							RUP	
		18.5.3			Describe relationships between boundary classes			User-interface designer							RUP	
		18.5.4			Present usability requirements on boundary classes			User-interface designer							RUP	
		18.5.5			Present the boundary classes in global class diagrams			User-interface designer							RUP	
		18.5.6			Evaluate results			User-interface designer							RUP	
	18.6				Describe interactions between boundary objects and actors			User-interface designer							RUP	
	18.7				Complement the diagrams of the use-case storyboard			User-interface designer							RUP	
	18.8				Refer to the user-interface prototype from the use-case storyboard			User-interface designer							RUP	
19					Prototype the user interface		User interface prototype	User-interface designer			Use case storyboard, boundary class, actor, supplementary specifications, user-interface guidelines				RUP	Steps can be alternated or performed in parallel.
	19.1				Plan the prototype		Prototyping plan	Requirements analyst		Event for prototyping, prototyping opportunity	Prototypes			Appendix A	R & R	
	19.2				Design the user-interface prototype			User-interface designer							RUP	
	19.3				Build prototype		Prototypes, context of prototype, objective of prototype, low fidelity prototype, high fidelity prototype, prototype building effort	Requirements analyst		Prototyping plan, prototype modification	Requirements specification			Appendix A	R & R	
		19.3.1			Build low fidelity prototype	Prototypes (paper and pencil) are drawn to illustrate objectives of the system	Prototypes, prototype building effort, context of prototype, low fidelity prototype, objective of prototype	Requirements analyst	Users	Prototyping plan, prototype modification	Requirements specification	Detailed event/use case model, scenario model event/use case, entity/state diagram, context diagram, sketch of screen layout		Appendix A	R & R	
		19.3.2			Build high fidelity prototype	Prototypes (software tools) are drawn to give a taste of how end product feels like	Prototypes, prototype building effort, context of prototype, low fidelity prototype, objective of prototype	Requirements analyst	Users, designers	Prototyping plan, prototype modification	Requirements specification	Simulation of user interface, simulation of the system's behavior for a given event/use case, simulation of the system's behavior for a combination of events/use cases		Appendix A	R & R	
	19.4				Evaluate the prototype		Potential requirements, prototyping metrics	Requirements analyst		Prototype modification, context of prototype, objective of prototype, low fidelity prototype, high fidelity prototype, prototype building effort	Prototypes, requirements specification, product scope			Appendix A	R & R	
		19.4.1			Test high fidelity prototype with users	Prototypes are experimented by users on their own to see if it meets the Objective of the Prototype	Prototype modifications (used until objective is satisfied), usage feedback new requirements, requirements changes due to prototypes	Requirements analyst	Users	High fidelity prototype, objective of prototype, context of prototype	Prototype is modified until it satisfies the Objective of the Prototype			Appendix A	R & R	
		19.4.2			Test low fidelity prototype with users	Prototypes are experimented casually and interactively	Prototype modifications (used until objective is satisfied), usage feedback new requirements, requirements changes due to prototypes	Requirements analyst	Users	Low fidelity prototype, context of prototype, objective of prototype	Prototype is modified until it satisfies the Objective of the Prototype			Appendix A	R & R	
		19.4.3			Get feedback on user-interface prototype			User-interface designer							RUP	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
		19.4.4			Identify new and changed requirements	Usage feedback is reviewed to discover new requirements	Potential requirements that needs to be passed through Quality Gateway	Requirements analyst		Usage feedback	Product scope, requirements specification			Appendix A	R & R	
		19.4.5			Evaluate prototyping effort	Evaluation is done on the prototyping effort. This can be used to define Prototyping Metrics	Prototyping metrics	Requirements analyst		Prototype building effort	Requirements specification, prototypes			Appendix A	R & R	
	19.5				Implement user-interface prototype			User-interface designer							RUP	
20					Structure use case model		Refined use case, new use case, refined use case model, refined use case package (optional)	System analyst			Use case modeling guidelines, glossary, use case model, use cases, supplementary specifications, use-case packages (optional)		RationalRose		RUP	
	20.1				Establish include-relationships between use cases										RUP	
	20.2				Establish extend-relationships between use cases										RUP	
	20.3				Establish generalizations between use cases										RUP	
	20.4				Establish generalizations between actors										RUP	
	20.5				Evaluate results										RUP	
21					Do requirements post mortem									Appendix A	R & R	
	21.1				Gather input for review		Quantified findings	Facilitator(s)		Individual comments, group comments, project history				Appendix A	R & R	
		21.1.1			Conduct private individual reviews	Individual reviews are conducted based on questionnaires or taped interviews to provide issues of the project	Points for clarification, input from individuals	Facilitator	Each project member	Individual comments		Sample questions on page 322 of R & R		Appendix A	R & R	
		21.1.2			Conduct separate meetings with groups	Group's experience are collected	Input from groups	Facilitator(s)	Working groups	Points for clarification, group comments				Appendix A	R & R	
		21.1.3			Facilitator reviews facts	The findings from individual reviews and group meetings are grouped and quantified and compared with actual history of the project	Quantified findings	Facilitator(s)		Input from individuals, project history, input from groups				Appendix A	R & R	
	21.2				Do post mortem		Post mortem report	Facilitator(s)		Quantified findings, project participants comments				Appendix A	R & R	
		21.2.1			Hold post mortem review meeting	Summary of findings are delivered to all involved in the project	Post mortem findings			Quantified findings, project participants comments				Appendix A	R & R	
		21.2.2			Produce post mortem report	The post mortem report is circulated among project members	Post mortem report			Post mortem findings		Sample of contents can be found on page 327 of R & R		Appendix A	R & R	
	21.3				Build a requirements filter		Post mortem report, requirements filter	Requirements analyst		System experience	Requirements filter, requirements specification, requirements template			Appendix A	R & R	
		21.3.1			Identify filtration criteria	The industry type for which the requirements filter is identified along with definition of the organizational environment and applicable technology	Industry type, organizational environment, technological environment	Requirements analyst		System experience				Appendix A	R & R	
		21.3.2			Select relevant requirement types	Each requirement is evaluated if it apply to the industry type or organizational environment or technological environment for which the project is built	Selected requirement types	Requirements analyst		Industry type, organizational environment, technological environment	Requirements template			Appendix A	R & R	
		21.3.3			Add new filtration criteria	Additions are evaluated frequently for future purposes	Requirements filter	Requirements analyst		Selected requirement types, post mortem report	Requirements filter, requirements specification			Appendix A	R & R	

					What			Who		When		How (mechanism)			Source	Notes
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
22a					Review requirements	Review meetings are conducted	Review record	Requirements reviewer	Customer, end user, and stakeholder		Vision, glossary, use case model, use case supplementary specifications, us case package (optional), software requirements specifications, use case modeling guidelines, iteration plan, change requests, user-interface prototype	Checkpoints: vision, stakeholder requests, us case model, actors, use case, supplementary specifications, software requirements specifications, glossary, requirements attributes	RequisitePro		RUP	
22b					Taking stock of the specification									Appendix A	R & R	
	22.1				Review specification content		Requirement interaction summary, missing requirements, contradictory requirements, prototyping opportunity	Requirements analyst		Strategic plan for product	Requirements specification, requirements filter, requirements template			Appendix A	R & R	
		22.1.1			Identify missing requirements	Requirements are cross-checked for requirements that might have been missed	Missing requirements	Requirements analyst		Strategic plan for product	Requirements filter or requirements template, requirements specification			Appendix A	R & R	
		22.1.2			Identify customer value ratings	Requirements are rated for customer satisfaction and customer dissatisfaction	Rated requirements (satisfied or dissatisfied)	Requirements analyst	Stakeholders	Strategic plan for product, requirement interaction summary	Requirements specification			Appendix A	R & R	
		22.1.3			Identify requirement interaction	Requirements that interact with one another (one design solution makes it easier or harder for the other) are identified	Contradictory requirements, requirement interaction summary	Requirements analyst		Requirements	Requirements specification	Interaction exist when there is a common policy, data, contradictory measurements, or when one has an effect on the solution to the other		Appendix A	R & R	
		22.1.4			Identify prototyping opportunity	Requirements which will benefit most from prototyping are identified	Prototyping opportunity	Requirements analyst		Strategic plan for product	Requirements specification	Questions on page 333 of R & R		Appendix A	R & R	
		22.1.5			Find missing custodial requirements	Requirements that change from time to time are checked to make sure that they are indeed changeable	Potential requirements	Requirements analyst		System terminology + requirement		Maintenance requirements for each item of stored data are checked. Context model for data flow are examined. External entities for system are checked. Storage of data items are inspected. Maintenance requirement is determined to be separate requirement or included as fundamental requirements		Appendix A	R & R	
	22.2				Evaluate requirements risk		Risk analysis, missing requirements	Requirements analyst		Requirement interaction summary, missing requirements, risk checklist	Requirements specification			Appendix A	R & R	Risks are okay so long as it is defined and monitored
		22.2.1			Look for likely risks	Requirements specification is reviewed for likely risks	Likely risks	Requirements analyst		Risk checklist and requirement interaction summary	Requirements specification	Unspecified requirement measurement is an indication of likely risk. Possible errors due to analyzing, designing and/or designing solution to the requirements indicate a likely risk.		Appendix A	R & R	
		22.2.2			Quantify each risk	Detailed assessment is performed on each risks	Risk analysis	Requirements analyst		Likely risks, missing requirements		Risk elements defined by Tim Lister and Tom DeMarco		Appendix A	R & R	
	22.3				Estimate effort		Event effort estimates, requirement effort estimates	Requirements analyst		Prototyping metrics, system experience, requirement interaction summary	Requirements specification			Appendix A	R & R	
		22.3.1			Identify estimation input	Events or use cases are used as inputs to the effort estimation	Event/use case models, functional requirements + non-functional requirements	Requirements analyst		Requirements specification				Appendix A	R & R	
		22.3.2			Identify efforts for events	Effort for events are estimated using Albrecht function points	Event effort estimates	Requirements analyst		Event/use case models, system experience, prototyping metrics		Event effort estimates = [event name + estimated function points] + total estimated function points for all events + estimate of what effort a function point means in this environment		Appendix A	R & R	

					What			**Who**		**When**		**How (mechanism)**			**Source**	**Notes**
Use case no.					Name	Description	Results (output)	Primary	Support	Input	Control	Guidelines	Tools	Templates		
		22.3.3			Estimate requirements effort	Effort is estimated using Albrecht function points (this is only suitable if event-related clusters are no identified)	Requirement effort estimates	Requirements analyst		Functional requirements + non-functional requirements, prototyping metrics, system experience, requirement interaction summary		Requirement effort estimates = {requirement ID + estimated points}+total estimated function points for all requirements + estimat of what effort a function point means in this environment		Appendix A	R & R	
	22.4				Publish reviewed specification		Reviewed specification	Requirements analyst		Event effort estimates, requirement effort estimates, risk analysis	Requirements specification, requirements template			Appendix A	R & R	
	22.4											Section 7.4 of IEEE Std 1233, 1998 edition (IEEE Guide for Developing System Requirements Specifications			IEEE	
		22.4.1			Design form of specification	Considerations are made on the design form of the specification	Form of specification	Requirements analyst			Requirements specification			Appendix A	R & R	
		22.4.1										IEEE Std 830-1998 (IEEE Recommended Practice for Software Requirements Specifications)		Annex A	IEEE	
		22.4.2			Assemble the specification	Specification is arranged for easy navigation	Reviewed specification	Requirements analyst		Event effort estimates, form of specification, risk analysis, requirement effort estimates	Requirements specification, requirements template			Appendix A	R & R	