REQUIREMENTS MANAGEMENT TOOLS

AMERAN JAFFAR

FACULTY OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2004

REQUIREMENTS MANAGEMENT TOOLS

Ameran Jaffar (WGC97008)

This is submitted to
Faculty of Computer Science and Information Technology
University of Malaya
In partial fulfillment of requirement for
Degree of Master of Software Engineering
Session 2001/2002

6th March 2004

Abstract

This dissertation is a research on project management tools specifically in the area of requirements management. The dissertation begins with an introduction on requirements engineering phase in a typical software development project. Subsequently the problems pertaining to requirements engineering are discussed. The reasons for failures on software development project due to lack of requirements are also discussed. We had conducted a questionnaire session to system analysts from fifteen local organizations. We found that even most of them recognize the problems on requirements but none are practising known requirements management methodology. We had reviewed a few requirements management methodologies and one method has been selected for a case study. A case study was conducted to find that methodology effectively improve the quality of the requirements specification. We also conduct a study on requirements management tools. This is to find out a model for more applicable tools, hence a system analyst or software engineer can practically use it. The enhanced model for the requirements methodology has been proposed. The proposed tools shall be simpler to that available in the industries and it offers interfacing with the tools normally used by our analyst such as Microsoft Word and Visual Modeler for Unified Modeling Language (UML). Hopefully, once the tool is completed, it will help our software industry. A prototype has been developed and demonstrated to the system analysts who participated in the previous case study. They have been given one week to test and give feedback to further enhanced the prototype. Some of the enhancements have been immediately modified but others may need more time.

Acknowledgement

I would like to especially thank my supervisor Puan Rodina Ahmad and Prof Madya Raja Ainon of Faculty of Computer Science and Information Technology, University of Malaya, who keep giving me guidance and good advice for the past year. In completing my questionnaire I would also like to thank all the contributors listed below :

1. En. Mustafa Amiruddin, Manager of Statistical Division, Information Services Department, Bank Negara Malaysia.

4. Syamsul Kamal Ahmad, Assistant Manager of Unit Sistem, Yayasan Tekun Nasional.

5. En. Razak Mohd Mazlan. Assistant Manager of Silverlake System(Brunei) Sdn. Bhd.

6. .Norhisham Jaafar , Analyst Programmer of Sepadu Komputer Sdn. Bhd.

7. En. Aisyam Mohamed, Senior System Analyst of Kementerian Kerja Raya, Kuala Lumpur.

8. En. Khairuddin Mohd Mor, Senior System Analyst of Kementerian Pembangunan Usahawan.

9. En. Rokman Mat Adam, MIS Manager of Metacorp Corporation Bhd.

10. Puan Siti Zubaidah, Senior System Analyst of Kementerian Perpaduan Negara dan Pembangunan Masyarakat.

11. En. Abd. Rahim Ramli, MIS Manager of Telekom Malaysia.

12. Mr. Lee Nan Phin, CEO of Info One Services Sdn Bhd.

13. Freedom Sham Hanafi, Analyst Programmer from Sepadu Komputer Sdn Bhd.

14. Azlina bt Azman, Institut Tadbiran Awam Negara

15. Wan Roshidah, Jabatan Perkhidmatan Awam Malaysia

LIST OF TABLES	Page

LIST OF DIAGRAMS	Page

Chapter 1

Introduction

1.1 Introduction to Requirements Engineering phase

The typical phases in software project management is normally divided into two main phases which are Analysis and Design phase and Implementation phase. This is illustrated in figure 1.

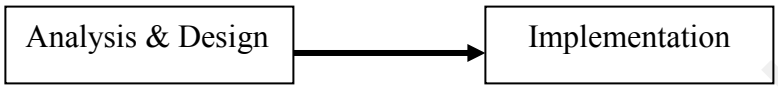


Figure 1: Analysis and design, and implementation phase

In figure 1, the analysis phase begins with project initiation or "kick-off", feasibility study and finally to perform analysis on the identified project scope. The typical design phase is to design input, process and output. This is inclusive of designing user interface, reports, database structure and also the program algorithm for the identified function as described in analysis report. For the software engineering method the approaches in design include architectural design, detailed component design and also logical and physical design. (Wilkie G, 1997 p 44). The implementation phase covers the coding and testing of the programs.

The project scope is defined from information gathered from current and potential user requirement. This is done through series of questioning and meeting sessions with user. The finalized project scope will be used by a system analyst to write a user requirements documentation. The user requirements

documentation is in the language of the user. This will act as an agreement between the user and the software developer. Both parties need to clearly understand the user requirements documentation. Hence, it may need to be revised multiple times in order to make it clear to both parties.

The user requirements documentation is used by the system analyst to write a software requirements specification. The software requirements specification is written in the language of the software developer. The software developer uses this document to develop the software system. (Wilkie G, 1997, p. 4)

Figure 2 illustrates the analysis and design phase, whereby the analysis phase is to deliver user requirements and software requirements documentation. The analysis and design phase is broken down into four more phases.

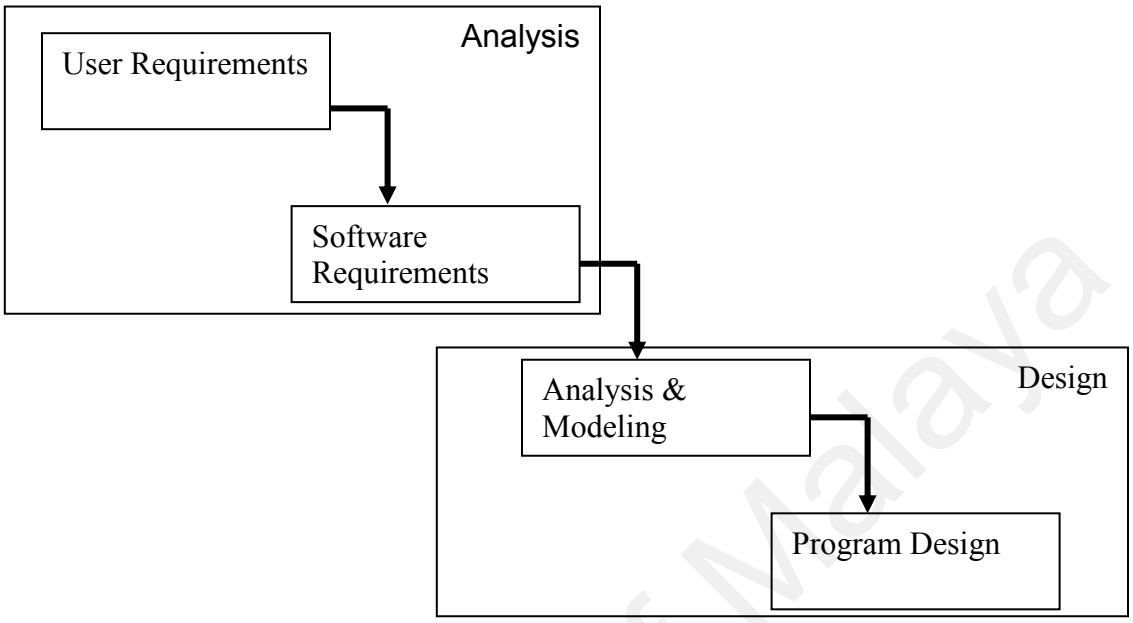


Figure 2 : Sub phases in analysis and design

From the illustration given above we can see clearly that, the user requirements and software requirements phases occur at the very beginning in the stages of software project management.

The user requirements consist of non-functional requirements and functional requirements. The non-functional requirements involve the statement of defining the environment of the system. This will include the definitions of system performance and constraints such as security, operations and physical. During this stage the definition of system development standards and the structure of the project working committee team and maintenance will be clearly defined.

The functional requirement is where the actual software function and services are defined.

The initial part of the requirements process is to capture the requirements from various sources. Therefore the 'Kick-Off' of the project shall start with the identification of the project working committee. The project working committee shall consist of both the potential users, the customer and the developer.

In a small scale project, it is normal to have a project manager, who also performs the role of system analyst whereby the system analyst will execute the process of requirements analysis. The structure of the committee can be very simple. On the other hand, in a large scale project development, the project working committee may consist of both the management and the operational level from all departments involved in the development process. The process of capturing user requirements usually has to involve meetings and discussions. These sessions may include meetings, interviews, questionnaires and any currently available documentation related to the current system. In a large scale system the requirements captured may change even after the final user requirements document has been accepted. Therefore managing requirements in a large scale project is quite a challenging task. The Project manager shall have in depth experience in managing a software development project. In a large scale project, the members may consist of many system analysts.

The project manager has to play the role of a requirements analyst, and he has to consolidate all the initial requirements documentation (sometimes called initial requirements artifact) and come out with the User Requirements Document.

The user requirements document of the system shall include the statement of definitions, which are properly grouped by its category.

Once the user requirement document have been endorsed by the project working committee, the user requirements document is considered final and from here the system analyst can produce the software requirements specifications before the analysis and modeling take place.

In software engineering the process of requirements analysis is sometimes being referred to as Requirements Engineering. Thus, in this thesis both requirements analysis and Requirements Engineering are used interchangeably.

1.2 Major Problems to Requirements Engineering.

The software projects, which are out of schedule, over budget and do not meet user expectations, are very high in percentage. According to a report by the Standish Group in 1997, Fortune's leading 1,000 firms spent, in 1996, 53% of applications development resources on projects that were technical failures or that required many more resources than initially provided. Other research shows that problems in requirements are not only more expensive, but more frequent as well. To remove an error committed in the software requirements phase can cost up to 20 times more than removing one made in the implementation phase. The reworking needed to offset imprecise user requirements documentation can take up more than 40% of a project's budget. (Meli R, 1999, p. 2)

Hence, from the above studies we can suggest that **by improving requirements management process, we probably can reduce failure, as well as the costs associated with software development projects**.

In order for one project manager to improve requirements management process, he in fact has to identify problems related to requirements management itself.

The problems related with requirements management include problems in defining the system scope, problems in fostering understanding among the different groups of people affected by the development of a system, and

problems in dealing with the volatile nature of requirements. These problems may lead to poor user requirements documentation and the cancellation of system development, or else the development of a system that is later judged unsatisfactory or unacceptable or has high maintenance costs due to frequent changes. (Christel M. G. et al, 1992, p 11)

According to Yeh R. T. et al (1984, p. 519-543), we can categorize the problems related to requirements management in two main categories which are “process and methodology related problems” and “documents or artifacts related problems”.

List of some “process and methodology” related problems are as below:

1. Lack of good requirements methodology or modeling. (Yeh R. T. et al, 1984)
2. Inadequate approach to Requirements Engineering. Most of the available techniques concentrate on functional requirements and provide relatively weak structures for expressing them. (Yeh R. T. et al, 1984)

List of some “documents or artifacts” related problems are as below :

1. The requirements document does not serve a role as communicator among customers, users, designers and implementers of the system. (Yeh R. T. et al, 1984)

2. The requirements document does not carry the weight of agreement or contractual relationships between parties. (No validation against it). (Yeh R. T. et al, 1984)
3. The system analyst or system engineer, do not have writing skills. (INCOSE, 1996, p. 1)
4. In the large scale project development, the process of developing any product requirements keep on evolving and changing. (Yeh R. T. et al, 1984)

1.3 Project Scope and Objectives

There were two known Software Project Management Methodology as quoted by Royce Walker (1999, p 6), first is the conventional Waterfall model and second is the Spiral model. The need for the second model is mainly due to the problems face during software development. It has been realized that, the requirements may change throughout the development cycle.

Among the main problems related to Requirements Engineering are lack of good requirements methodology, inadequate approach to Requirements Engineering, no validation against user requirements document, no reference to the good user requirements document and no traceability if the requirements change during product development. Therefore, the scope of this project will be, to address the above

mentioned issues in relations to the Software Industry in Malaysia for both private and government sector. We do not intend to include the established international companies due to most of them were not ready to open their internal business process.

The objectives of the project shall be as below.

Project Objectives :

a. To survey the existing Requirements Engineering Methodologies to support the Requirements Engineering phase.

b. To survey the existing practice in Malaysian Software Industry, both government and private sector.

c. To study available Requirements Engineering model and tools to support the Requirements Engineering phase.

d. To propose a model of the Requirements Engineering which is applicable to Malaysian Software's Industries.

e. To develop a working prototype for the proposed model.

1.4 Contents of the Report

Chapter 1 gives an introduction to this project. It describes the problem domain, objectives of the project and tasks carried out to complete this project.

In chapter 2 we explain the literature review done on various types of models pertaining to Requirements Engineering phase and make comparison among them. In this chapter, the functional aspect of Requirements Engineering together with tools and models supporting them are also been identified. In this chapter we also compare on various requirements tools available according to its functionality.

In chapter 3 we explain why the research required to produces more suitable and practical model for the industry in our country and how to conduct the research. The questionnaire has been designed and conducted to get the expert's view from industry pertaining to their practice to requirements analysis. A case study has also been designed to identify the best model for the industry.

In chapter 4 we describe the existing selected model and the reason why it has been chosen. It also proposes the enhancements and describes the justification for the enhancements.

Chapter 5 is a documentation on the prototyping tools developed for the new enhanced model.

Chapter 6 is an overll finding of the research. We discuss on the limitations of the research, their proposed future improvement, our contribution and also the overall conclusion of the project.

1.5 Research Methodology

The reasearch methodology in this project is according to the list stated below :

i. Literature Review.

   a. To study on the Software Project Management model, 'Waterfall' and 'Spiral' in conjucntion with Requirements Engineering.

   b. To study on the Software Development Methodologies such as Structured Analysis, Object Oriented Analysis and Problems Frame; specifically during Requirements Engineering phase.

   c. To study on the Requirements Engineering models derived from various methodologies.

   d. To study on the history of Requirements Engineering models, the common Requirements Engineering tasks or business processes.

   e. To analyse strength and weaknesses of the Requirements Enginering models.

f. To analyze four available models and recommend the models that support the project objectives d. (refer to page 10).

ii. Industrial Experts review.

a. To gather information on industrial experts on :

- Their current Requirements Engineering practice.
- Their current Software Development model in practice.
- Tools being used for Software Development and Requirements Engineering.

b. To identify the effectiveness of using a known standard Requirements Engineering methodology as compared to not using a standard method.

iii. To propose a new model that support project objectives d. (page 10).

iv. To review available tools for Requirements Engineering and propose a new tools prototype to be develop to achieve project objectives d and e. (page 10)

v. To develop a Requirements Engineering tools prototype.

Chapter 2

Literature Review

2.0 Requirements Engineering Methodology

Conventional software project management, is normally presented by the Waterfall Model, in the Waterfall model, each phase in software development project must be delivered first before one can move to the next phase. Winston Royce as quoted by Royce Walker (1999, p.6) , in his paper based on managing large software projects, using the Waterfall Model, suggests that software development methodology for large scale project that follow s waterfall model are subject to high failure risk. This is due to difficulty of getting right requirements from user in the initial stage of software development cycle. The user requirements in large scale project, normally keep on changing due to its volatile nature. The testing phase, happens at the end of development cycle is the first event for which real integrated system is verified. If there are changes in design it is very disruptive when there is possibility that, the whole system need to be modified. Furthermore the agreement between user and software developer which is in the form of user

requirements and software requirements documentation, upon which the design is based, is likely to be violated.

Winston Royce, introduced the backward steps to allow any changes made is immediately incorporated when it is identified. Figure 3, is the illustration of Waterfall Model for the large-scale system development project. The dotted arrow is the newly introduced backward step.

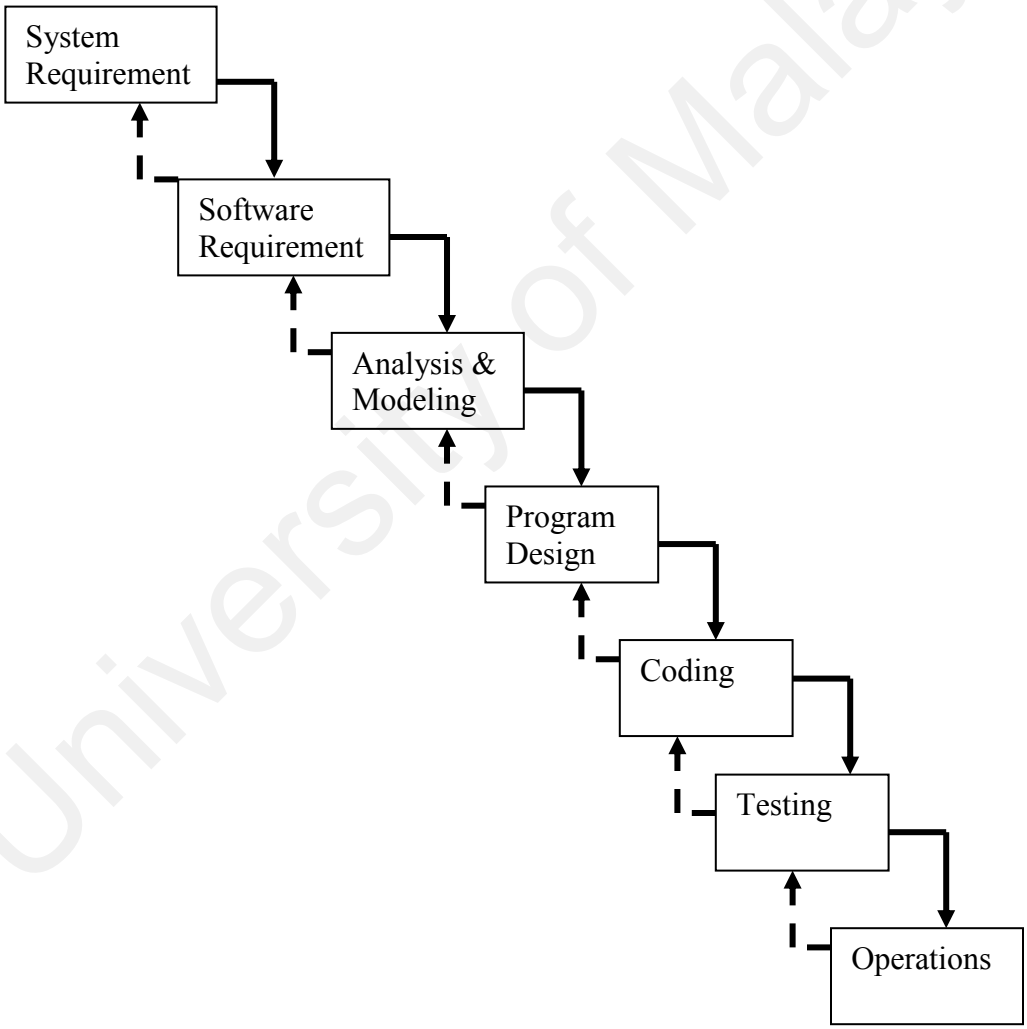


Figure 3. Waterfall Model in the large scale system.

(Source : Royce Walker p. 7)

Realizing the problems faced by Waterfall Model, Royce Walker (1999) suggested a framework for modern software management to replace the Waterfall Model. His book is more on managing software development project from the software engineering methodology rather than the old structured method. He has divided the Software Project Management stages into two stages, which are engineering and production stages. The engineering stage consists of the inception of the idea and also the elaboration of the architecture, whereby the production stage consists of the construction of beta releases and also the transition period to the final product delivery. This is actually synonym to the analysis and design phase and also implementation phase, which has been elaborated in chapter 1.

However, the typical approach to the product design, starts with the establishment of Work Breakdown Structure (WBS) which is a composition of product and activity hierarchies. Hantos P(1999), introduce Product Specification Tree(PST) which is an extended version of product hierarchies in WBS. PST indicates special characteristics of the subsystem. PST are not static and its support iterative life-cycle model. During system decomposition the tree is traversed up and down and it will change the allocations. Transition zones are defined that mark the boundaries where during decomposition, the change of technical disciplines is needed. The technical disciplines can be the knowledge in

software development process, hardware specifications and also writing skills. The ability of system engineers to manage transition zone will lead to the success of the requirements engineering.

According to Royce Walker(1999), each stage, in the software development process shall be supported by a set of artifacts. The set of artifacts can be in the form of documentation or model diagrams. The requirements artifacts comprise of an Initial requirements specification (vision document) and an analysis model (requirements model). Again, this is synonym to user requirements and software requirements documentation, which has been described in chapter 1.

Life-cycle phase or spiral model focuses on artifact sets. Each artifact set is a predominant development of one phase in the life-cycle that has been described in the previous paragraph.

Each stage of development represents a certain amount of precision in the final system description. Early in the life-cycle, precision is low and the representation is generally high. Eventually, the precision of representation is high and everything is specified in detail.

The inception phase focuses mainly on critical requirements, usually with a secondary focus on an initial deployment view. Little focus is given to implementation except perhaps choice of language and commercial components,

it is also possibly that some high level focus is given to the design architecture but not on design detail.

During the elaboration phase, there is much greater depth in requirements, much more breadth in the design set, and further work on implementation and deployment issues. The elaboration phase includes the generation of an executable prototype.

As development proceeds, each of the parts evolves in more detail. When the system is complete, all sets of artifacts are fully elaborated and consistent with one another.

In order to capture the evolution of the requirement, Wayne Woodruff (1997) suggests the requirements to be managed in hierarchy. The hierarchy is for both requirements and documentation (artifacts).

As we had previously discussed, the Requirements Engineering phase is divided into two main sub tasks which are, (i) preparing the user requirements document which can be used as an agreement between developer and the user, and (ii) preparing software requirements specification which will be later be used by the system analyst to design the system. Both documents are in fact artifacts which will probably be changed over time. Most of the Requirements Engineering methodology today incorporate what is called life-cycle or spiral approach of software development methodology. The methodology allow artifacts to be

changed many times, hence improvement in managing traceability and changes become one of the major tasks.

According to Lanman J. T. (2002), the software requirements methodology is the subset of the whole software developments methodology. According to him, there are three methodologies that can be used in software requirements : Structured Analysis, Object Oriented Analysis and Problem Frames. The first two method are the most common and the third method is a less known and used methodologies. Structured Analysis is a method that produced functional specifications as an artifact which is represented by data flow diagram (DFD), process specifications and a data dictionary. The Object Oriented Analysis (OOA) supported by the artifacts such as Software Engineering Requirements and Specifications. The OOA artifacts expressed by a System's Object model which is composed of a population of interacting objects, as opposed to the traditional data and functional views in Structured Analysis. The Problem Frames is a way of structuring a problem by partitioning it into principal parts. Patterns and architect is produced as an inventory of problem frames.

In this chapter, the comparison between the common tasks in Requirements Enginnering is made against four Requirements Engineering models. The models, has been selected based on its availability from various type of organization. The models that will be discussed here are :

1. Texas Instruments System Level Automation Tools for engineer (SLATE)

models as described by Goodwin S. and Nellon J. (1999).

2. US Naval Air System Command Requirements Management as described by US Naval Requirements Management Working Group –RMWG (1999).

3. The INCOSE Requirements Management Process as described by INCOSE requirements working group (May 1998).

4. The Rational Unified Process (RUP) by Rational Software Corporation (2001).

INCOSE (1997) had claimed to be the first to analysze the common Requirements Engineering Tasks (or business processes or activities). The processes was taken from various software engineering papers and books in particular, Oliver and Stainer (1996), Rechtin(1991), EIA IS-632 Interim Standard Systems Engineering (1994), IEEE std 1220 Trial Use Standard for Application and management of the Systems Engineering Process (1994).

From 1998 to 2001, we can see many newly introduced Requirements Engineering models, but the underlying process was not different from what has been introduced by INCOSE (1997). The processes was further improved and the recently introduced model is to detail down the underlying process.

SLATE is a Requirements Management tool developed by Texas Instruments. It was developed to provide a step towards fulfilling the total needs of systems engineering with the concentration on the early phases of the software product

development process. In order to develop a tool, Goodwin S. et al (1999) has performed research, design and develop the methodology of the Requirements Management and Technical Performance Management Subsystems.

US Naval Air System Commands through the Computer Resources Group's (CRG) Requirements Management Working Group (RMWG) has developed a framework and methodology for establishing and maintaining a requirements management discipline for software engineering activities and provides assistance in tailoring the process to meet project-specific goals. The Requirements Management process contained in the document (US Naval RMWG 1999) meets the objectives of the Software Engineering Institute's (SEI) Key Process Area (SEI February, 1993) for Requirements Management. In addition, the process has been expanded to allow application to systems engineering activities. Apart from publishing the framework, US Naval RMWG does not develop any tools to support the implementation of the framework.

International Council on Systems Engineering or INCOSE also established a group call requirements working group. The requirements working group has developed INCOSE Executable Requirements Model (INCOSE, May 1998) based on the model which was proposed earlier (INCOSE, 1997). INCOSE does not develop any tools, but SLATE has been recommended as one of the tools to implement INCOSE Executable Requirements Model.

Rational Software Corporation, a company that is popular with research on methodology in software engineering has developed a Rational Unified Process or RUP, which is a comprehensive set of methods and guidelines for developing software applications. RUP is an Object Oriented software development method. Development phase in RUP is an iterative development process, which is following the method proposed by Royce Walker (1999). According to RUP method in requirements management apply use cases and other requirements techniques. The RUP is designed to incorporate a software lifecycle method that is iterative and incremental, as this method helps teams address the risk inherent in a new development effort more effectively than the rigid "waterfall" model. Leffingwell D. et al (1999) described many requirements methods and some of it adopted by RUP. During planning a team required to choose a requirements method. He proposed the method to reflect the types of risks inherent in the environment since the purpose for requirements management is to reduce risk. More interesting Leffingwell also discussed on the Agile Software Development (Cockburn,2002) which is a method to develop a new class of products for which no current customers exist.

The remaining of this chapter is going to compare the four methodologies and where it fits into the first two phases of sub phases in analysis and design (Figure 2) which are identifying user requirements and software requirements.

2.1. SLATE Requirements Management methodology by Texas Instruments, Goodwin S. and Nellon J. (1999)

2.1.1 Identifying User Requirements

According to SLATE, the Requirements Management process begins with the elicitation and development of a set of user or customer needs and objectives (See figure 4 on capturing the requirement). Systems engineers must translate these needs and objectives into a complete and understandable set of requirements. The systems engineers then analyze and allocate these requirements to the appropriate levels of the system design. Identifying user requirements in SLATE involves steps as listed below :

- Establish and maintain communications with customer.
- Identify customer needs, mission objectives, environmental conditions and operational scenarios.

- Elicit and identify the requirements.
- Analyze the requirements.

2.1.2 Identifying Software Requirements Specification

According to SLATE, the requirements, architecture and analysis results are captured in specifications that can be represented in a specification tree that identifies all requirements documentation for a program. A specification tree and a system hierarchy are two mechanisms used to illustrate the dependencies of the design elements and the documentation of the requirements.

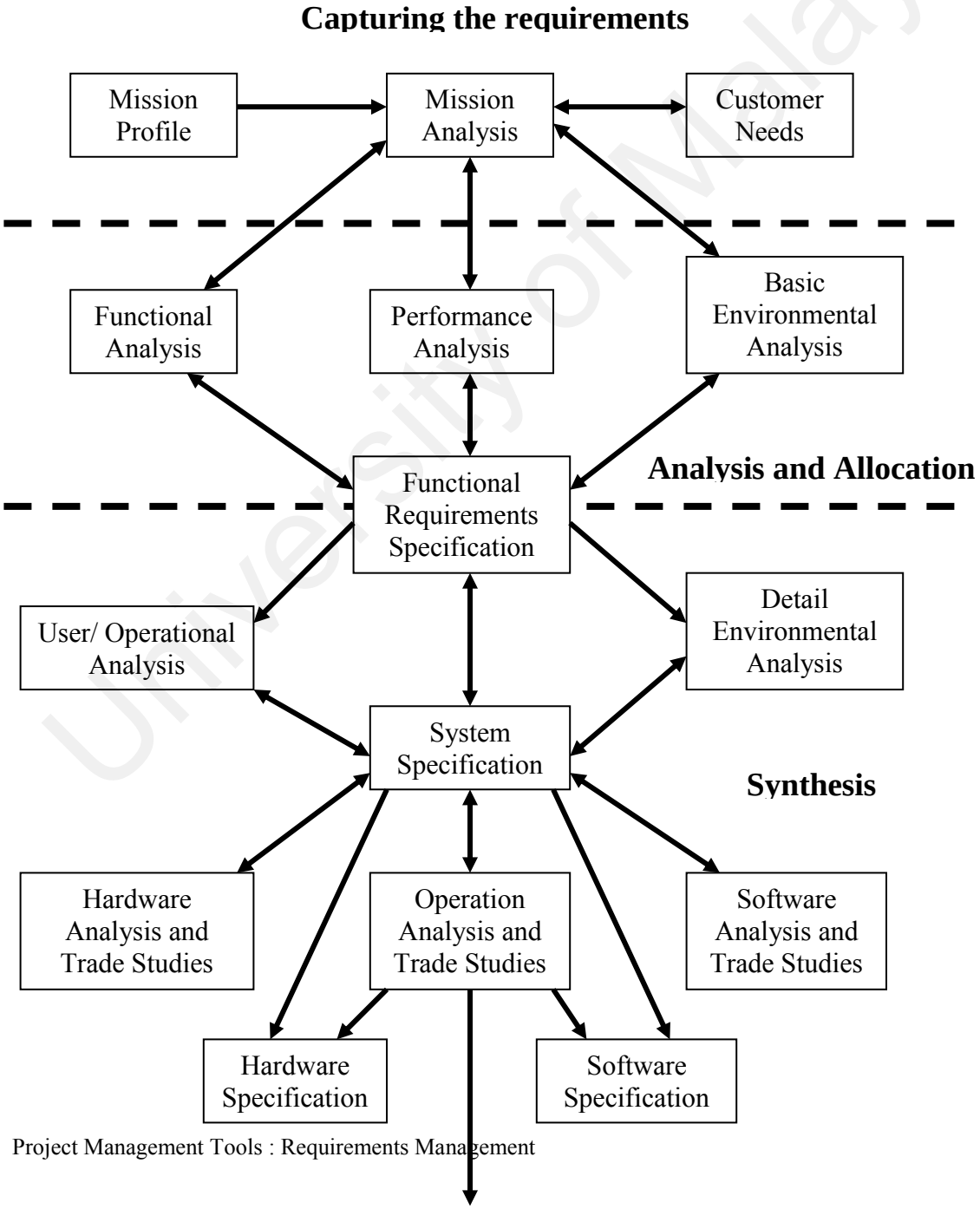
Systems engineering identifies the system requirements, collects them by function, derives and allocates them to the appropriate system segment(s). To identify software requirements specification in SLATE will involve steps as listed below :

- Analyze the requirements.
- Translate identified needs into design requirements.
- Allocate the requirements to lower level documents and system elements.
- Establish and maintain traceability.
- Analyze and monitor quantitative requirements.
- Generate all top level system documents and oversee all lower level documents.

- Verify that the design meets the requirements.
- Ensure that all development, coding, verification, deployment, operations, support, training and disposal issues are addressed.

2.1.3 Managing requirements hierarchy, change and traceability

According to SLATE, the allocation and flow down process is iterated from the highest level of the hierarchy, to lower and lower levels of detail until the lowest configuration item is identified in the hierarchical tree (Figure 5). The systems engineers allocate requirements from the top node of the system hierarchy to the lowest design element by developing, deriving, allocating, and validating the requirements one hierarchy level at a time. The derived and allocated requirements are documented in the appropriate configuration item specification as defined in a system specification tree and Work Breakdown Structure (WBS). The specification tree will essentially be synonymous with the system hierarchy in that all functional elements of the system hierarchy typically have a corresponding document to capture the requirements for the design element.



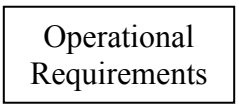


Figure 4. Capturing the Requirements according to SLATE.

2.1.4 Strength and weakness

SLATE addresses the software requirements specification and managing the requirements hierarchy, change and traceability in detail, especially on how to capture the requirements. Even though SLATE describes the requirements processes in detail, it does not describe on how to deliver the user requirements documentation. Since the user requirements will act as an agreement between the customer and the developer, we believe this cannot be excluded.

(Source : Goodwin S. and Nellon J. (1999))

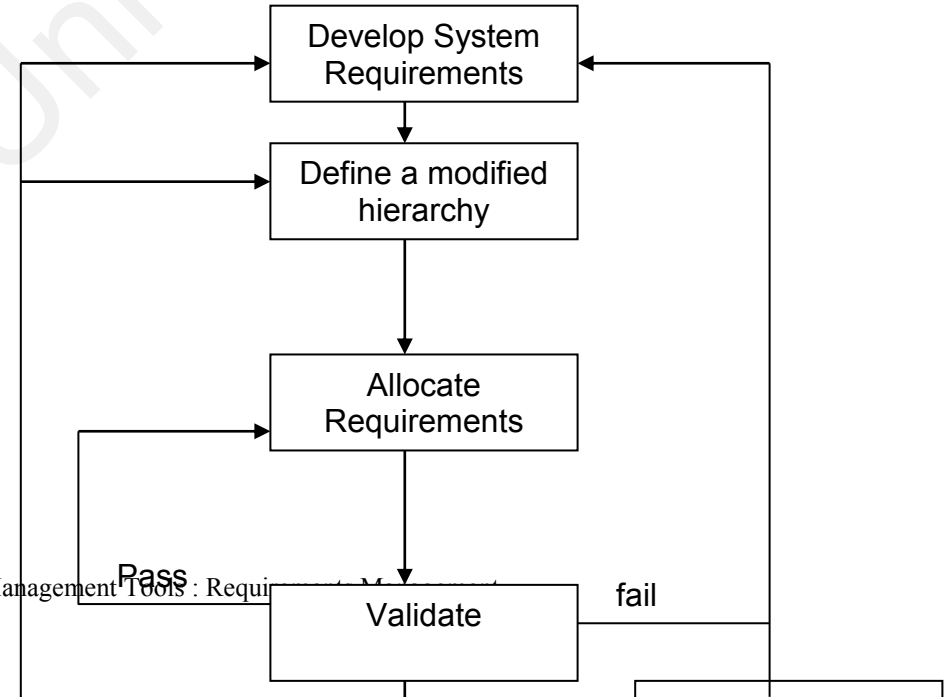


Figure 5. SLATE Requirement Process Flow

(Source: By Goodwin S. and Nellon J. (1999))

2.2 US Naval Requirements Management methodology

Figure 6 and Table 1 below presents the overall flow of activities and participation for the Requirements Management process in USNAVAL.

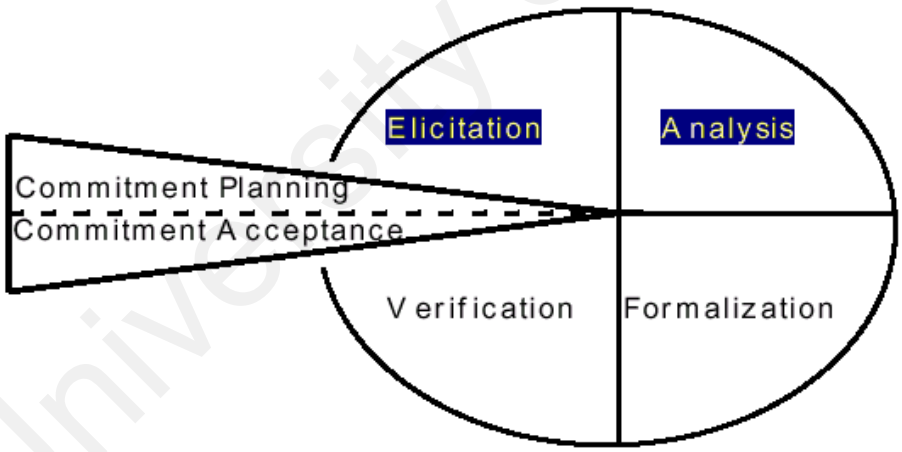


Figure 6. Requirements Management Sector

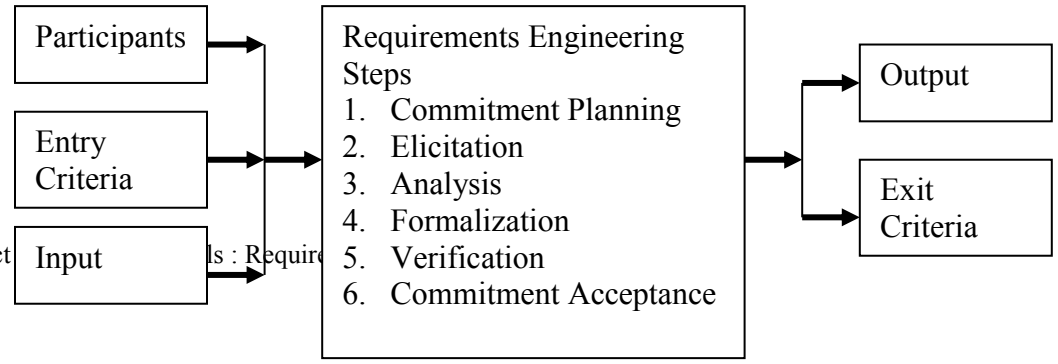


Figure 7 : Requirements Management Process Flow

Table 1: Requirements Management Process Flow

Inputs	**Processing**	**Outputs**
Mission Need Operational Scenario Requirement Change or Deficiency	1.Commitment/Planning 2. Elicitation 3. Analysis 4. Formalization 5. Verification 6. Commitment/ Acceptance	Authenticated Requirements Approval to Proceed

2.2.1 Identifying User Requirements

According to USNAVAL, the user requirements is identified when the need for planning and agreement of new or changed requirements has been identified from the **participants** such as Senior Management, Customer, Project Management, Systems Engineering, Software Engineering, CM/QA, Functional Testing, Operational Testing, End User. The initial inputs are documents related to Mission Need, Operational Scenario, Requirements, Change or Deficiency Report.

Identifying user requirements involve steps below, which is part of steps in the USNAVAL requirements engineering process. (Refer to figure 6 & 7 and also table 1) :

- **Commitment Planning** - During this activity the goals, constraints and acceptance criteria for this spiral cycle are agreed upon by the stakeholders.
- **Elicitation** - An **Elicitation** of customer requirements is performed to obtain an understanding of the operational scenario, mission needs, or changes documented in a change or deficiency report.
- **Analysis** - An **Analysis** of the requirements is performed to evaluate the quality of the requirements.
- **Verification** - The **Verification** activity is used to review, modify (if necessary) and baseline the documented requirements.
- **Commitment Acceptance** - The **Commitment Acceptance** activity provides stakeholders visibility into the current state of the process and products. Approval to proceed to the next Requirements Management

spiral cycle or next process is obtained from the customer. Authenticated Requirements are produced.

The **outputs** shall be the Authenticated User Requirements documentation with an Approval to Proceed. The customer must have approved the baselined requirements before meeting the acceptance criteria for the cycle. Approval to proceed to the next process is obtained.

2.2.2 Identifying Software Requirements Specification

According to SLATE, identifying software requirements specification will get input from user requirement documentation and its involve steps as listed below, which are part of USNAVAL requirements engineering process. (Refer to figure 6 & 7 and also table 1) :

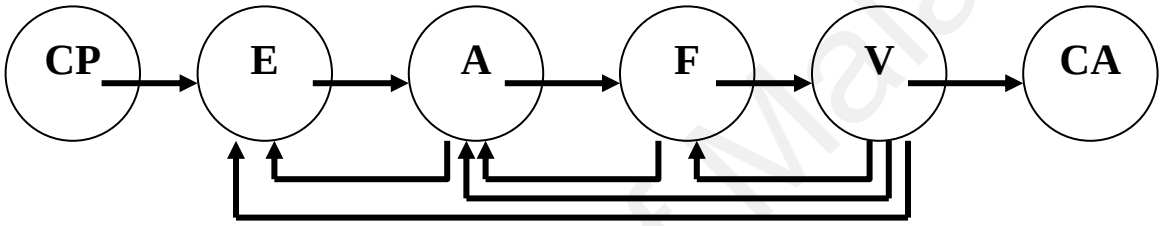
- **Analysis** - An **Analysis** of the requirements is performed to evaluate the quality of the requirements.
- **Formalization** - A team understanding of the requirements, their traceability and supporting rationale are recorded or documented in the **Formalization** activity.
- **Verification** - The **Verification** activity is used to review, modify (if necessary) and baseline the documented requirements.
- **Commitment Acceptance** - The **Commitment Acceptance** activity provides customer view into the current state of the process and products. Approval to proceed to the next Requirements Management

spiral cycle or next process is obtained from the customer. Authenticated Requirements are produced.

The **outputs** shall be the Authenticated Software Requirements Specification with an Approval to Proceed

2.2.3 Managing requirements hierarchy, change and traceability

USNAVAL, manage requirements hierarchy, change and document traceability via an activity interactions. Interactions are present between neighboring activities during progression through a cycle. In addition, iterations that address preceding functional activities may be required to complete the current cycle. The interactions are determined by the nature of the defect or clarification that is required. For example, figure 8 depicts elements discovered during the **Verification** that could be addressed in the **Elicitation**, **Analysis** and/or **Formalization** activities. In addition, the figure 8 suggests iterations between the **Analysis** and **Elicitation** activities, as well as between the **Formalization** and **Analysis** activities.



CP – Commitment Planning

E – Elicitation

A – Analysis

F – Formalization

V – Verification

CA – Commitment Analysis

Figure 8. Activity Interactions between process according to US Naval. (Adapted from US Naval RMWG (1999))

2.2.4 Strength and weakness

USNAVAL address the user requirements and software requirements specification in detail. Even though USNAVAL described the requirements

processes in detail, the method to manage requirements hierarchy, change and traceability has not been discussed in the model itself. The information on how to manage requirements hierarchy, change and traceability are available in its sample and tailoring guidance.

2.3 INCOSE Requirements Management Methodology

The Requirements Management process as published by INCOSE (1997); has given the first look into the processes, which are some common underlying features that relate to the application of software tools which was studied by INCOSE. The underlying features are :

- There is a resulting hierarchy of requirements composed of collections of requirements that correspond to the various parts of the system being engineered and the people who work on them. The hierarchical arrangement of the requirements occurs because normally System development is composed of sub-tasks, such as subsystem development, which are the responsibility of particular groups of people such as a company or a team. The arrangement repeats at lower levels; and each level is responsible for all lower level requirements.

- To derived requirements some analysis was involved. Any particular requirement in a requirements collection in the hierarchy comes either from another collection, from an analysis, or an external source variously called the "customer" or "users / buyer / suppliers". Requirements from an external source may be in non-standard, or informal form, such as telephone conversations or meeting records. Requirements that are put into collections are in a formal standard form.

- As requirements are generated or revised the flow of requirements to collections must be done in an orderly, gated manner. Generally the people responsible must agree to accept the requirements involved. Some requirements in collections do not pass to other collections or analyses but instead are implemented.

2.3.1 Identifying User Requirements

INCOSE user requirements begin with identifying “needs and desire” from the customer or “users/buyer/supplier” (Refer to figure 9). User requirements are in the customer's terms and may not be well stated or easily measurable. The user requirements will then be analyzed and stored into requirements collections. Figure 9 depict two levels of requirement analysis, the first level is for identifying the user requirements. Second level is for the identification of system requirements and its repeated to the next level for the sub-system

requirements. The “requirements collections” block for the first level is actually a collection of user requirements. This is validated to the user via “validation loop” block.

2.3.2 Identifying Software Requirements Specification

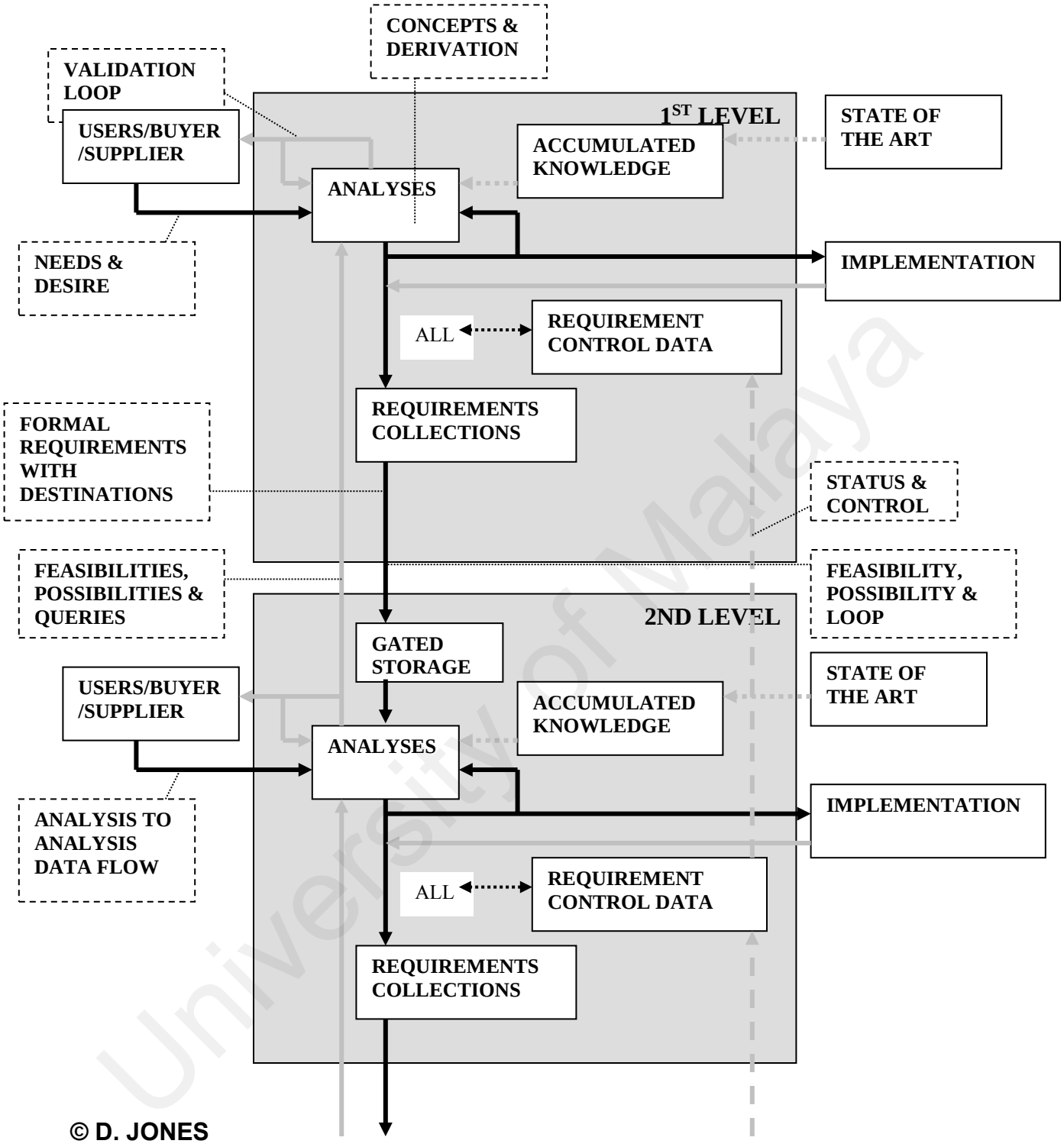
INCOSE model identify system requirements on the “second level” block and the “requirements collection” for the second level is a collection of system requirements. Software Requirements Specification or System requirements, which are derived by analysis of the customer requirements, are in system terms and should be well stated and measurable. In addition to requirements themselves, the requirements deals with a special attributes - a hierarchy of feasibilities, possibilities, and queries that are associated with the requirements. If the requirements flow goes downward then the attributes flow goes upward. Refer to figure 9, the heavy downward directed lines represent the requirement flow and the lighter upward lines are the special attributes – the feasibility, possibility and query data that were generated from the requirements being placed in the collections.

2.3.3 Managing requirements hierarchy, change and traceability

Apart from special attributes the INCOSE requirements also have the control data which is a record of a requirements in the hierarchy and their revision, source and destination. The control data enable the traceability of the requirements change. The control data represented by the dotted upward direct lines (refer to figure 9). The control data typically consist of that shown in table 2.

2.3.4 Strength and weakness

INCOSE address the software requirements specification and managing the requirements hierarchy, change and traceability in detail. Comparing to SLATE, INCOSE does not discuss on the methods of capturing requirements but INCOSE model clearly shown the next level of iteration. Comparing to the other two models, INCOSE model is more comprehensive in elaborating on requirements attribute and control data but lack in requirements planning and elicitation. INCOSE treat elicitation as part of analysis. Like SLATE, INCOSE does not clearly define the process of delivering user requirements.



(Pattern Repeats)

Figure 9. INCOSE Requirement Data Flow

(Adapted from INCOSE, 1997)

Table 2. Requirement Control Data

(Adapted from INCOSE, 1997)

Group	**Item**
Index key	Identifier and revision designator
Traceability items	Source requirement ID
	Destination(s)
	Owner
Source status items	Certainty / validation
	Questions to source
	Importance / criticality
Destination status items	Risk / feasibility
	Suggested alternatives
Verification items	Verification method
	Verification level
Transverse reference	Relationship
items	Destination

2.4 Rational Unified Process (RUP) Requirements Management Methodology

2.4.1 Identifying User Requirements.

RUP adapt the development method as proposed by Royce Walker (1999) and the requirements identification is part of engineering stages which is involved inception and elaboration. The user requirements is identified during inception phase. The user requirements document, will involve steps and consist of artifacts as listed below :

- Overriding goal is obtaining buy-in from all interested parties
- Initial requirements capture
- Cost Benefit Analysis
- Initial Risk Analysis
- Project scope definition
- Defining a candidate architecture
- Development of a disposable prototype
- Initial Use Case Model (10% - 20% complete)
- First pass at a Domain Model

2.4.2 Identifying Software Requirements Specification

The software requirements specification is identified during elaboration phase. The specification required requirements analysis and capture the artifacts as listed below (Rational Software Corporation, 2001) :

i. Scenarios – to describe the scenarios of the project.

ii. Use Case Analysis - to perform Use Case Analysis and deliver the Use Case model. The analysis is done many times and at the end of requirements analysis phase, the status of artifacts is as below :

- Use Case (80% written and reviewed by end of phase)
- Use Case Model (80% done)
- Sequence and Collaboration Diagrams
- Class, Activity, Component, State Diagrams
- Glossary (so users and developers can speak common vocabulary)
- Domain Model
- to understand the problem: the system's requirements as they exist within the context of the problem domain
- Risk Assessment Plan revised
- Architecture Document

2.4.3 Managing requirements hierarchy, change and traceability

Since RUP designed base on iterative development model and its focuses on artifacts set. Each artifact set is a predominant development of one phase in the life-cycle that has been described previously. Each stage of development represents certain amount of precision. Early in the life-cycle, precision is low and the representation is high. Eventually, the precision of representation is high and everything is specified in detail. The requirements hierarchy is managed using a technique named “Parent-Child Requirements”. RUP “Parent-Child Requirements” is to increase specificity and its elaborate on how to organize the “Parent-child Requirements”. (Leffingwell D et al, 1999)

2.4.4 Strength and weakness

RUP methodology is a complete software engineering methodology, which is an object oriented software development methodology. Hence the strength of RUP is on managing object oriented software development. RUP is specially designed for software engineers that are familiar with Rational Software Corporation’s software engineering methodologies. The RUP’s approach to requirements management involved the usage of use cases and model the business domain using Unified Modeling Language or UML (Rational Software Corporation, 1997). Leffingwell D. et al (1999) from Rational Software Corporation discussed thoroughly the process on managing software requirements, which they called a unified approach. The unified

approach discussed many requirements management approach in different software development environment but does not really introduce any new method. RUP adapt some of the method discussed by Leffingwell D. Comparing to SLATE, USNAVAL and INCOSE, RUP together with all materials develop by Rational Software Corporation's, RUP probably are the most comprehensive model available. The problem is when the model has to be applied together with all other methodologies set by Rational Software Corporation. Many companies, especially those involved in developing legacy application are not interested in Rational's methodology. Even though the RUP are very detail in most of the requirements management process they are not easy to implement.

2.5 The Common Tasks to Requirements Engineering

The table below gives a summary on comparison on tasks involved during Requirements Engineering phase for one complete cycle according to the methodologies previously mentioned, SLATE. (1999), US NAVAL (1999) , INCOSE (1997) and RUP (2001). Generally, we had rectified the requirements is hierarchical in nature and the processes or tasks in acquiring requirements may need to be repeated many times. Each cycle consists of several tasks and in general for one complete cycle we can categorize the requirements tasks into 5 main tasks.

Table 3 : Tasks involved during Requirements Engineering – comparison by methodology

SLATE By Goodwin S. and Nellon J. (1999)	US NAVAL by US Naval RMWG (1999)	RUP by Rational Software Corporation (2001)	INCOSE by INCOSE (1997)
1. Planning			
♦ Establish and maintain communication with customer ♦ Identify customer needs, mission objectives, environmental conditions and operational scenarios	Commitment/ Planning – During this activity the goals, constraints and acceptance criteria for this spiral cycle are agreed upon by the stakeholders.	♦ Overriding goal is obtaining buy-in from all interested parties ♦ Initial requirements capture	"Needs and desire" of User/Buyer/ Suplier
2. Elicitation			
Elicit and Identify the requirements	Elicitation – to obtain an understanding of	♦ Cost Benefit Analysis ♦ Initial Risk	Formal Requirements & Environments.

	the operational scenario, mission needs, or changes documented in a change or deficiency report.	Analysis ♦ Project scope definition ♦ Defining a candidate architecture ♦ Development of a disposable prototype ♦ Initial Use Case Model (10% - 20% complete) ♦ First pass at a Domain Model	
3. Analysis			
Analyze the requirements	Analysis – to evaluate the quality of the requirements.	♦ Use Case Analysis ♦ Use Case (written and reviewed by end of phase) ♦ Use Case Model	Analysis & Requirement control data

		♦ Scenarios ♦ Sequence and Collaboration Diagrams ♦ Class, Activity, Component, State Diagrams ♦ Glossary (so users and developers can speak common vocabulary) ♦ Domain Model ♦ to understand the problem: the system's requirements as they exist within the context of the problem domain ♦ Risk Assessment	

		Plan revised ♦ Architecture Document	
4. Formalization			
♦ Translate identified needs into design requirements ♦ Allocate the requirements to lower level documents and system elements ♦ Establish and maintain traceability ♦ Analyze and monitor quantitative requirements ♦ Generate all top level system	Formalization – to review, modify (if necessary) and baseline the documented requirements.	Formalize documents as artifacts for this cycle.	Requirements collections & identify special attributes

documents and oversee all lower level documents.			
5. Verification			
♦ Verify that the design meets the requirements. ♦ Ensure that all issues are addressed.	Verification Commitment/ Acceptance – provides stakeholders visibility into the current state of the process and products. Approval to proceed to the next Requirements Management spiral cycle or next process is obtained from the customer. Authenticated Requirements are produced.	Verify the artifacts against the stakeholders.	Validation loop

The common tasks to Requirements Engineering shall be illustrated as below :

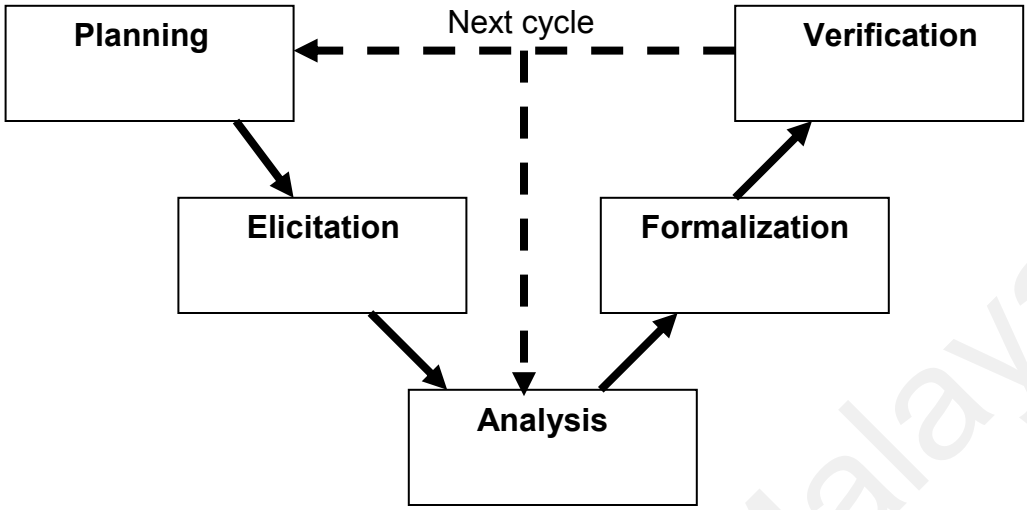


Figure 10 : Requirements Engineering Common Tasks

(1) **Planning** - During this activity the goals, constraints and acceptance criteria are agreed upon by the User. The Project Working Committee has to be identified.

(2) **Elicitation** - An **Elicitation** of User Requirements is performed to obtain an understanding of the operational scenario, mission needs, or changes (traceability) documented in a change or deficiency report.

(3) **Analysis** - An **Analysis** of the requirements is performed to evaluate the quality of the requirements.

(4) **Formalization** – The Project Working Committee understanding of the requirements, their traceability and supporting rationale are recorded or documented in the **Formalization** activity.

(5) **Verification** - The **Verification** activity is used to review with user or working committee depend on the product of the current cycle. Verify will modify (if

necessary) and endorsed the documents. Verify will also identify the proceeding to the next cycle.

(6) **Next Cycle** – depends on the outcome of the verification, the **Next Cycle** may start with the **planning** again or direct to the **analysis** for the next level of requirements hierarchy.

2.6 Summary Analysis

2.6.1 Planning phase

All the four methodologies agreed on "Planning phase" is to define higher level definition of the project scope, objectives, mission, needs or desire. SLATE (1999) and RUP (2001) specify "Planning task" is only for the first spiral cycle and the next-cycle is to include the flexibility of accepting requirement change through spiral development methodology. USNAVAL (1999) described in detail, steps involved during planning, but have not discussed much on development methodology. RUP (2001) does not discuss much on steps since Rational Software Corporation's treated this subject as a separate subject.

USNAVAL (1999) and SLATE (1999) was developed and established from within their organization, hence the model describe in detail on the planning.

INCOSE (1997) and Rational Software Corporation as an independent research institute does not discuss so much on planning. This is probably due to the planning phase in Requirements Engineering is more organization dependent. The rigid model will not satisfy all type of organizations.

2.6.2 Elicitation phase

Again we can see USNAVAL (1999) and RUP (2001) had described the "elicitation task" in detail whereby SLATE (1999) and INCOSE (1997) does not describe so much on the elicitation task. Elicitation task in INCOSE, RUP and SLATE is part of the first level analysis which is to deliver the **user requirements documentation**. In SLATE the elicitation task was not appear in the model but it is mentioned in the steps acquired to produce user requirements documentation (refer to figure 5).

2.6.3 Analysis phase

All methodologies, SLATE (1999), USNAVAL (1999), INCOSE (1997) and RUP(2001) had thoroughly defined the steps involved during analysis, but USNAVAL seems to define more comprehensive steps. RUP (2001) analyze the requirements using various methods and model such as use case analysis and UML. INCOSE (1997) elaborate more detail on

requirements attribute measurement. INCOSE (1997) combined the elicitation task together with analysis, which is the first level analysis for the highest level requirement.

2.6.4 Formalization phase

SLATE (1999), INCOSE (1997) and RUP (2001) does not describe steps involved in formalization. All three methodologies describe generally the formalization tasks and both addresses the nature of hierarchical level of the requirements. USNAVAL (1999), describe clearly steps involved in the task.

2.6.5 Verification phase

Again the USNAVAL (1999) described the steps involve during verification task. SLATE (1999) and RUP (2001) does not discuss so much on verification, both of them only described validation as a step before documents are produced. INCOSE (1997) fairly discusses on verification tasks but it does not describe it, in step by step. In general all models agreed into, in order to produce the product for any cycle, it must be verified. Verification is also used in all models as the step compulsory before moving to the first tasks in the next cycle.

2.7 Review on the Requirements Engineering tools

2.7.1 Computer Aided Software Engineering (CASE) Tools

For the past few chapters we have been discussing about various types of Requirements Engineering methodologies, however this project is about to develop a prototype for Requirements Engineering tools. Hence, in order to develop the prototype we need to identify and analyze any tools available in the market and suggest a model for tools prototype to be developed.

In the tremendous growth of software industries, tools have been developed in order to assist developers to properly manage the various stages in software development. Currently, there are many CASE tools in the market and they can be categorized into Analysis & Design; Programming & Debugging; Project Management; Graphical User Interface Builder; Quality, metrics and testing and Documentation tools.

Analysis and Design tools have been used for quite sometime to help developers to present and model the Process and Data Flow Diagram together with Database Entity Relationship modeling (Hernandez M. J.,1996);. Some advance methodology for Software Project Management Tools has been used together with the Analysis and Design tools. Some typical examples are SSADM developed by British Computer Society

(Ashworth C. et al., March 1993) and also METHOD/1 as developed by Andersen Consulting (1990).

Project Management tools have been used by the best-in-class organizations. According to Jones P. (1978), the best-in-class organizations may have more than 10 times the quality tool capacities and 30 times the project management tool capacities than the organizations that fail with software. The difference between companies that succeed and those that do not is that the former employ effective project management tool suites whereas the latter generally do not.

The usage of Function Point metric has becoming popular to the new software project management tools. The Function Point is used to produce cost estimation of the project and evaluate the completeness of various kinds of software tool suites. This approach can clearly reveal some of the critical differences between successful software projects, average projects, and total failures. (Jones P., 1978)

2.7.2 Requirements Engineering and Design Tools

2.7.2.1 Definition

According to INCOSE (1997), a Requirements Management toolset is one that facilitates Requirements Management. The following basic features are considered to be significant for tools :

- Identification of "individual" requirements
- Assignment to a destination and sorting of requirements
- Requirement group (collection) revision identification / base lining
- Providing a basic data interface.

The tools must be able to store the information electronically with capability to print. Such a tool set may be composed of one or several tools, and any of these tools may perform functions in addition to Requirements Management.

2.7.2.2 Tool Categories

INCOSE (1997) categorized the Requirements Management tool sets as follow:

1. Document-based: Traces requirements by adding tables / links to pre-existing documents.
2. Requirement tracing: Traces and often edits requirements between imported requirements collections.
3. Requirement generating: Generates requirements collections from within the tool. Such tool sets generally can also be used in a similar way to those in the preceding category.
4. Built-in modeling and simulation.

2.7.2.3 Tool Functionality.

According to INCOSE (1997) an automated Requirements Management tool can support both the engineering discipline and the management discipline to manage requirements. Such a tool must be able to collect and manage both technical and programmatic requirements.

Refer to Requirements Management Tool Survey Summary from INCOSE. (Appendix A.). INCOSE had divided the Requirements Management Tool into functions which we had summarized into statistical table as below .

Table 10. Requirements Management Tools Functional Summary

Requirements Engineering functions	Percentage of tools supported
1. Capturing Requirements/Identification (Functional Requirements)	70%
2. Capturing System Element Structure (Non-Functional Requirements)	70%
3. Requirement Flow Down (Sub-system requirements)	70%
4. Traceability Analysis (Version Control)	70%
5. Configuration Management	70%
6. Document & Other Output Media	70%
7. Groupware	70%
8. Interface to other tools	60%
9. System Environment	10%
10. User Interfaces	60%
11. Support & Maintenance	50%

The technical functions required by the Requirements Engineering tools, according to INCOSE (1997) are :

1. Requirements identification, viewing and editing

2. Tracing requirements to their origin, and report generation.

3. Change impact analysis. When a requirement is change, all affected requirements must be identified and verified to ensure their integrity after the change.

4. Completeness and consistency checking.

The management functions required by the Requirements Engineering tools, according to INCOSE (1997) are :

1. Metrics collection and monitoring requirement stability through change control.

2. Change control consists of keeping track of any adds, deletes, or changes to existing requirements.

3. Tracking changes and the reasons for the changes can lead to improved processes to reduce changes, and hence costs, on future programs. With this data collection, continuous improvement can be applied to reduce the instabilities in the future.

2.7.2.4 Auxiliary Functions

A function that INCOSE(1997) has not included in their definition of a Requirements Management tool is analysis or design, including system design. A number of commercial tools however include such auxiliary functions as functional analysis and simulation capabilities, either built into the tool or in the form of a closely integrated companion product with which the tool interfaces.

When auxiliary functions are included in a tool the associated interfaces are generally hidden. On one hand the integrity of the interface is likely to be good because all aspects of it are controlled by one vendor, but on the other hand one of the advantages of a tool interface is lost - the ability to use alternative tools for a given function.

2.7.2.5 Data Interfaces

According to INCOSE(1997), the major types of interfaces are

- Document import / parsing tool
- Analysis tools (including functional model, physical model, simulation, schedule, and risk management)
- Publication tool (electronic and / or paper)
- Database / data files
- Engineering design tools
- Companion programs (e.g. drawing editors, etc.)
- Other Requirements Management tools

2.7.2.6 Tools and organization maturity level

As we had been discussed in the previous chapter, the INCOSE (1997) model are complex. It is a reason why INCOSE (1998) had introduced an executable model which is a simpler version. Similar goes to the tools suggested by INCOSE, it is again have complex functionality and it may not feasible for the new and small organizations.

SEI's has introduced a field guide to effective requirements management under SEI's Capability Maturity Model which is adapting the requirements management for small organizations from Rational Software Corporation.

According to Woodruff W. (1997), in many smaller and immature organizations, requirements management is a difficult challenge. Smaller groups mean fewer resources, and many organizations focus their efforts on design, development and testing not on managing requirements. Some small organizations may perceive requirements management as an activity only for large organizations that have complex products and large staffs to support the effort.

Woodruff W. (1997) addressed three issues to manage requirements in a smaller organizations :

- Establishing a requirements and document hierarchy

- "Decomposing" requirements documents
- Maintaining requirements and traceability

According to Woodruff W. (1997), the Rational Software Corporation's RequisitePro is one of the tools that support the above issues.

2.7.2.7 Summary Analysis

In order for us to develop a new Requirements Management tool, It is important for us to understand the existing Requirements Management tools functionality. INCOSE (1997) provide thorough understanding on the existing tools and how the tools has been categorized and position into the bigger scope of Project Management tools and CASE tools.

INCOSE (1997) suggest the implementation of iterative development using “spiral model” through change impact analysis. Iterative development, emphasizes on artifact management. INCOSE (1997) suggest a data interface to be developed to ensure the artifact’s application, to be integrated to the tools. The framework on project management about handling artifact as proposed by Royce Walker (1999), probably is one of the most comprehensive materials regarding iterative development. In order

for today's Project Manager to be able to use the tools it must be able to support artifacts management. The Requisite Pro from Rational Software Corporations is probably the only tools support Royce Walker (1999) project management framework.

In order to be more realistic to the Malaysian software industry, the tools for small organizations are more applicable. We found not many simple tools that enable small scale developer to use it but the SEI's article (Woodruff W, 1997) on the Rational Software Corporation's RequisitePro is one of a kind. The article can be used as a baseline for development on any new simpler requirements engineering tools to be used by the small or immature organizations.

2.8 The existing practice in Requirements Engineering

The Software Engineering Institute(SEI) at the Carnegie-Mellon University developed framework for Process Capability Maturity Model (CMM) for software development (Stephen H. K., 1995, p. 40). In most of the mature organizations, the process of Requirements Engineering is normally documented as a set of standards to be followed by their software engineers. Even though the usage of Requirements Management tools is not yet established, some mature organization which are according to CMM belong to Level 3 and above, have

developed tools which conform to some functionality of Requirements Engineering. Matured organizations probably at least have a guideline for one to follow. The problems happen normally to immature organization when they do not have any set of documented guidelines or procedures.

The immature organization which are according to CMM belong to Level 1 or 2, will probably leave it to their software engineers or system analysts to come out with the user documentation and specifications. Often this document does not go through the proper process of **Planning, Elicitation, Analysis, Formalization and Verification**. Managing requirements may not be easy in this situation whereby, when the **requirements change**, there will be no traceability to the previous document. From the research done (refer to chapter 3), most of the organizations today, are managing documents artifacts manually. Some of the organizations, use Electronic Documents Management System as tools to manage documents, but most of them still are using word processors and standard windows folders to keep and organize their documents.

We had conducted a survey (refer to chapter 3 for survey detail), on 10 software houses or software development departments in Malaysia. We had grouped them into three categories, where category A is for companies established for more than 15 years and their CMM Level are 3 and above, category B is for companies established within 7-15 years which is belong to CMM Level 2 and category C is for companies established less than 7 years which their CMM is at

Level 1. Our assumption here is the CMM level is normally grow parallel with the establishment of the organization.

For the survey conducted, three of them are belong to category A, two of them belong to category B and five companies belong to category C.

Most of the companies in category A have documented guidelines on software development practice but the emphasis on Requirements Engineering phase is minimal. For category B, these companies have documented software development practices but compared to category A, the documentation is very brief. For category C, the companies do not have development guidelines documented and most of its software engineers or system analysts do not even have a proper filing system, whether electronically or manually.

When we asked about Requirements Management using computerized tools, the finding shows that most of the software houses in Malaysia today do not have any practice in place.

2.9 Summary

The Requirements Engineering or Requirements Management processes are very important to enable successful delivery of the software product.

A unified framework in software project management as published by Walker Royce (Royce Walker, 1999) can be a very good framework for those who are interested in life-cycle model. Rational Software Corporation had develop a methodology to manage a unified framework which is called Rational Unified Process or RUP (2001). The artifacts base management tools, has been developed based on this framework which is called Requisite Pro. The tools even though work fine with Rational's methodologies but it does not applicable for the small software companies which normally do not have capacity to follow rigid methodologies.

Requirements Management must maintain control data for the requirement hierarchy and facilitate operator control of the process. Therefore comparing to SLATE (1999) and USNAVAL (1999), the methodology as proposed by INCOSE (1997) can be accepted as a more complete Requirements Management process methodology except it does not discuss process steps in details. Even though USNAVAL (1999) describe more detail on steps involved in each Requirements Engineering task, the flow of INCOSE (1997) model cover almost all aspect mentioned in the other two. INCOSE (1997) also introduced the simplified executable model (INCOSE, May 1998) and also published the list of tools that confirm to INCOSE. INCOSE (May 1998) does not recommend new tools but proposes developers to use many types of different tools that are available in the market.

The Requirements Engineering processes are part of their main business processes to the software organization. Software Engineering Institute's (SEI) Capability Maturity Model (CMM) classify and stated the practice for process improvements for the organisations but this may not be applicable to the small business. (Judith G. B. , 1994). Hence, it is impossible to have tools that are applicable to all CMM's levels and also for all sizes of business in each level. We no doubt need to have a Requirements Tool for Malaysian Software Industry both private and government which most of them are small in size and immature.

The INCOSE (May 1998) simplified executable model is more practical, at least a PC base model shall be developed based on INCOSE (May 1998) requirements to integrate tools, together with the capability in managing artifacts as what proposed by Royce Walker (1999) and Requirements Management process steps as detailed by US NAVAL (1999). Later the data interface capability shall be added to improve the tools.

Chapter 3

Industrial Experts Review

3.0 Introduction and Objectives

In order to come out with a Requirements Engineering methodology that is more suitable and practical, research needs to be carried out. The research is divided into two phases which are :

a. Gathering information on industrial experts practice : The questionnaire has been sent and several interview sessions have been conducted with experts from various organizations. The purpose of the questionnaire is to understand the current practice in the Requirements Engineering, hence the objectives of the questionnaire are :
   i. To gather information from industry experts on their current practices in Requirements Engineering Methodology and to find relationships between the age of the organization's establishment and their documents management.
   ii. To find if they are practicing 'Life Cycle or spiral' development model (Royce Walker, 1999).
   iii. To find if they are using tools in software development especially during Requirements Engineering phase.

b. A case study has to be designed to identify the effectiveness of using known standard methods compared to without using standard method during Requirements Engineering. We had selected INCOSE

Executable Requirement Model (May, 1998), because of its simplicity. The objectives of the Case Study is to find out :

i. How INCOSE model can enhance the characteristics of **good requirements** as described under "well defined" requirements characteristics by INCOSE (1996), by comparing INCOSE model with the implementation without model ?.

ii. How INCOSE model can enhance **requirements manageability – Traceability and Accessibility** by comparing INCOSE model with the implementation without model ?.

Each characteristic of the requirements specification generated following INCOSE model and the requirements specification generated without model is given mark ranging from 1 to 10. 1 is for the worst and 10 is for the best. The evaluation has been carried out by two independent persons. (Refer to Acknowledgement for personal details).

3.1 Gathering information on industrial expert practice

Most of the software development related personnel is familiar with the term – **Software Requirements Specification**. The **Software Requirements Specification** is produced from sets of **User Requirements Documentation,** which are the results of the Requirements Engineering phase in software development. It is common for the Requirements Engineering methodology to be the subset of the standards set in the bigger scope of Software Development

methodology. Computerized Tools has been developed to support some of the methodologies. Therefore the Requirements Engineering tools are also a subset to one complete set of Computer Aided Software Engineering (CASE) tools. There are many types of models established for Requirements Engineering and also tools to help on delivering the model.

As previously mentioned, the Software Development methodology, also contributes to the development of Requirements Engineering methodology. Even though the 'Waterfall Model' has been in practice for so long, some adjustments have been made by some organization to make it capable of handling requirements change (INCOSE, 1996). The research in software project management (Royce Walker, 1999) has shown the best approach in software development model is 'Life-Cycle or spiral Model'. In this model all typical stages in software development will be done more than once.

The Requirements Management in 'Waterfall Model' normally endorsed once the Analysis and Design Model has been agreed upon, which means no more changes can be done until the whole package is delivered. In 'Life Cycle or spiral Model' the requirements can be changed in between and normally the very basic functions will be delivered first and the rest will evolve as the system is used by the user.(Royce Walker, 1999)

Questionnaires on “Requirements Engineering Practice” and how marks is given to the respondent.

1. a. How do you normally organize **User Requirements documentation** and **Software Requirements Specification** ?

   (Mark 0 – for no systems at all, 5 – for the response which is using manual or partly computerized system, 10 – for the response using fully computerized system)

   b. Do you follow any standard in preparing your **User Requirements Documentation** and **Software Requirements Specification**, if Yes, please specify ?

   (Mark 0 – for the response which is definitely NO, 5 – for the response which is following the standard set by the management, 10 – for the response which is following the known requirements management standard endorsed by the management)

2. a. Do you use any Requirements Management tools to help you to prepare **User Requirements Documentation** and **Software Requirements Specification ?**

   (Marks 0 – for the response which is definitely NO, 5 – for the response which is using tools other than specifically designed for requirements management tools, 10 – for the response which is using known requirements management tools)

   b. if Yes please specify ?

(Marks 0 – if it is not specified , 5 – for the response which is not an established tools or indirect tools such as CASE tools or simple application developed without proper Requirements Management analysis , 10 – if the tools is recognized and it is clearly specified)

a. Do you use **'Life Cycle or spiral Model'** .(Royce Walker, 1999) in Software Development in your organization ?

(Marks 0 – if the response is NO, 5 – for the response which is mixed or modified version of "waterfall model", 10 – if the response YES)

b. If no, what do you think the **problems** that encourage you to implement the **'Life Cycle or spiral Model'** model (Royce Walker, 1999) ?

(Marks 0 – for no problems identified, 5 – for the correct response but not related in managing requirements hierarchy, change and traceability, 10 – for the response which is related to managing requirements hierarchy, change and traceability)

3. How do you manage **changes** in requirement and **traceability** to the previous version of requirements?

(Marks 0 – if it is no response , 5 – for the response which is not an established tools or indirect tools such as CASE tools, office applications or simple application developed without proper managing requirements change

and traceability analysis , 10 – if the tools is in used are recognized and it is clearly specified)

4. Do you think **tools** will help you manage requirements, if YES how ?

   (Marks 0 – if it is no response or do not know how , 5 – for the response which is YES but the purpose of the tools is not clearly described , 10 – if the tools is to solve the problems of requirements management is described)

5. Have you ever used the **CASE tools for Analysis & Design**, if yes state what tools, when and where ?

   (Marks 0 – if the response is NO , 5 – for the response which is YES but the purpose of the tools is not clearly described , 10 – if the tools is clearly described)

6. Do you think tools will help you to produce **User Requirements Documentation** and **Software Requirements Specification**, how ?

   (Marks 0 – if it is no response or do not know how , 5 – for the response which is YES but the purpose of the tools is not clearly described , 10 – if the tools is to solve the problems of requirements management in managing documents/artifact is described)

3.1.1 Organizing Survey and Questionnaire

The questionnaire had been given to 15 industrial experts from different types of organizations representing a sample of the Malaysian Industry. They were grouped into three categories, category A for the organizations whereby their MIS department was established more than 15 years, category B for organizations established within 7-15 years and category C for organizations established less than 7 years.

Category A organizations

1. Bank Negara Malaysia.
2. Telekom Malaysia.
3. Silverlake System Sdn Bhd.
4. Institut Tadbiran Awam Negara (INTAN).
5. Jabatan Perkhidmatan Awam (JPA).

Category B organizations

6. Sepadu Komputer Sdn. Bhd.
7. Kementerian Kerja Raya.
8. Kementerian Pembangunan Usahawan Nasional.
9. Metacorp Corp.
10. Kementerian Perpaduan Negara dan Pembangunan Masyarakat.

11. Jabatan Perikanan, Kementerian Pertanian Malaysia.
12. Institut Latihan Kehakiman dan Perundangan (ILKAP).

Category C organizations

13. Yayasan Tekun Nasional.
14. Info One services Sdn. Bhd.
15. Advanced Altimas Sdn. Bhd.

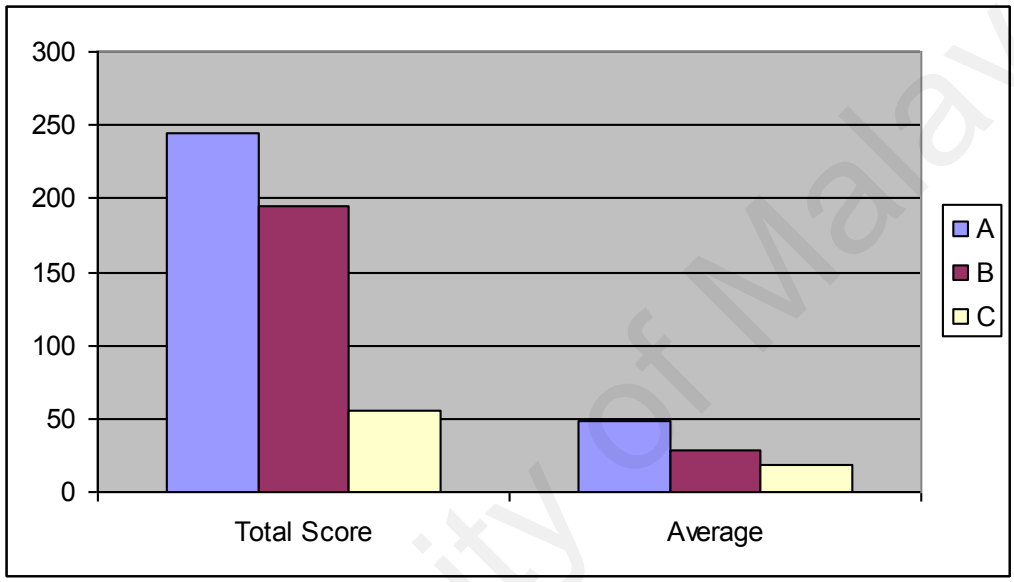
Each of the experts had been approached individually and all of them had been given a short briefing on the objectives of the questionnaire and also the terminology used especially on various type of development models and Requirements Engineering methodology. Except for those from international base organizations, many responded positively to the questionnaire. We had difficulty to get response from the international base organizations and also some local organizations due to their unwillingness to expose the methodologies practiced by them. Hence, our samples may not represent the real picture of the industry.

Table 4 : Questionnaire Results Weighting Summary

Questionnaire \ Organizations	Category	1a	1b	2a	2b	3a	3b	4	5	6	7
1	A	5	5	0	0	5	10	10	5	5	5
2	A	5	5	5	0	5	10	5	10	10	10
3	A	5	5	5	0	5	10	10	5	10	10
4	A	5	5	5	0	5	5	5	5	5	5
5	A	5	0	0	0	5	10	5	5	10	5
6	B	5	0	0	0	5	5	5	5	5	5
7	B	0	5	0	0	5	10	0	0	5	5
8	B	0	0	0	0	5	0	0	0	0	5
9	B	0	5	0	0	5	0	5	10	5	10
10	B	0	5	0	0	5	10	0	5	5	5
11	B	0	0	0	0	5	10	5	5	5	10
12	B	0	0	0	0	5	5	0	0	5	5
13	C	0	0	0	0	5	0	0	0	0	5
14	C	0	5	0	0	5	0	5	5	5	0
15	C	0	0	0	0	5	5	0	5	5	0

Table 5 : Questionnaire Results Average by Category

Category	Total Score	No of Samples	Average
A	270	5	54
B	210	7	30
C	55	3	18.33



3.1.2 Discussion on findings

Different organization may practice different methods during Requirements Engineering phase. All category A with one category B that we survey have a system to organize User Requirements documentation and System Requirements Specifications. The system is a combination of a manual and a computerize system. Most of category A which is four out of five, three out of seven category B and one out of two category C

organizations, have at least a guideline set by their management on Requirements Engineering methodology. We can draw the conclusion that managing requirements is more organized in the mature organizations. It is disappointing to note that **none** of the organization that we survey is following any known requirements management standard. This means the requirements management standard, even it is available is not popular among the local developer.

The result of questionnaire no 2 is even more disappointing when only three of the category A organizations in the sample are using tools during Requirements Engineering phase. The tools being used here is either the internally develop application to manage the storage and retrieval of the documents or the CASE tools that is having some level of requirements management. Again **none** of the organization that we survey is following any known requirements management tools and again this means the requirements management tools, even it is available is not popular among the local developer. The sample is probably not enough for us to make a conclusion but it is believed that it's generally to show the actual scenario.

All organizations are using "waterfall model" with slight modifications. This is probably due to most of the organizations begin with the typical "waterfall model" methodology. In fact majority are using SSADM. Except four, the rest of the respondents realized on the situation faced by

"waterfall mode" which are incapable of managing requirements hierarchy, change and traceability. The modification to the "waterfall model" is made unofficially to allow changes to user requirements and system specifications. Again **none** of them is following the known standard 'Spiral/ Life Cycle Model' methodology (Royce Walker, 1999). This briefly shows that, even though most of the organizations are initially using "waterfall model", they have to modify it to cater for the requirements change during development time. Hence indirectly the 'Life Cycle Model' (Royce Walker, 1999) is in fact in used by most of the organizations.

In managing change, nine out of fifteen respondent responses that, they have a mechanism to manage change. Only two out of nine organizations, use known documents management tools that support version control. The other seven organizations are implementing proper filing system which some of them partly computerized. We may conclude that version control, be it manually or computerized to manage change has been recognized as part of general administration task to many organizations especially the matured organizations (category A and B).

Regarding to having tools to manage requirements, four respondent seems do not have any idea at all of the purpose of requirements engineering tools. Another eleven have some ideas on the purpose of the tools but incapable to describe clearly the overall picture of the tools. Only

two respondent, address the problems of requirements management as the purpose of having tools. We can conclude that, not many organizations or systems engineers really know how to address the problems of requirements engineering. It seems like the issue of requirements management is not yet popular to the local software development industry.

About the CASE tools, thirteen out of fifteen respondents have an experienced using CASE tools during analysis and design phase, but only two of them manage to described properly the purpose of using tools. Again base on our sample we can conclude that, “CASE tools” has already been popular to the local software development industry.

Regarding to using tools to help producing **User Requirements Documentation** and **Software Requirements Specification** more than 90% of the respondents agreed that tools will help. Four of the respondents manage to address the real issue of requirements management. We think the positive feedback from the respondent on the given question, is due to their realization on the subject after series of discussion.

3.2 Case Study on the Requirements Engineering Methodology.

3.2.1 Case Study Background

Based on the conclusions made from the previous phase of research – "Gathering information on industrial experts practice", which was conducted to get some information on the industry practice in Requirements Engineering, it was found that none of them are using any established methodology or tools during Requirements Engineering phase. 40% of the samples have standard guidelines but it is part of their general documentation standards.

Until now, we cannot find any study to show that using an established methodology will result in better quality Requirements Specification. The case study is designed to identify the effectiveness of an established method in Requirements Engineering to deliver quality Requirements Specification.

After referring and comparing various models in Requirements Engineering methodology as described in Chapter 2 Literature Review, all models – SLATE, USNAVAL, INCOSE and RUP at least agree to the common processes listed below to enable successful implementation of the Requirements Management process :

- Planning
- Elicitation
- Analysis
- Formalization
- Verification

As INCOSE methodology is comprehensive enough yet simple to implement especially for the analysis stage, it has been selected as a methodology to be applied for Requirements Engineering in the case study.

The above mentioned processes are the processes to achieve 'well defined' requirements characteristic. The case study will adapt definition of 'Well defined' requirements from INCOSE(1996). INCOSE(1996), describes 'well defined' requirements characteristics in three areas :

i. Characteristics of individual requirements. (Refer to Appendix)
ii. Characteristics of aggregate requirements. (Refer to appendix)
iii. Supporting characteristics to individual requirements. (Refer to Appendix)

Based on the response from the industrial expert's questionnaire we can confirm the above mentioned processes is required for a Requirements Management process and the management capability criteria like traceability support should be added to the case study objectives.

Referring to **INCOSE (May 1998)** which describes executable Requirements Management model, it provides the simplest reasonably complete model possible which from this can be implemented with or without tools. This Model, has been chosen for the case study. The case study also adapted the

comprehensive requirements classification according to Yeh R. T. et al (1984).

3.2.2 The preparation of the Case Study.

The case study was conducted at one of the established software development company by interviewing two system analysts. The system analysts were given a 15 minutes briefing session on the objectives of the case study.

They were required to prepare Requirements Specification according to IEEE Std 1220-1994. IEEE defined that, the Requirement is a statement identifying a capability, physical characteristic, or quality factor that bounds a product or process need for which a solution will be pursued.

The measurement of the Case Study Result will be as given below :

Good Requirements Characteristics - will be as specified by INCOSE "well defined" requirements characteristics (refer to Appendix), and weight will be given from 1 to 10, whereby 1 is the worst and 10 indicate the best.

Traceability & Accessibility– will be tested on both System Analysts, the response time to trace the requirements changes and to produce the

requirements, and again weight will be given from 1 to 10, whereby 1 is the worst and 10 indicate the best.

3.2.3 The Execution of the Case Study

The case study was conducted using two different projects. The first Project was the development of Premise Online system at Kementerian Pembangunan Usahawan and the second project was the development of Complaints Information Database System at Bank Negara Malaysia. Both projects are similar in terms of number of functions to be delivered which is about 10 functions, their allocated development of six man months and also their development cost which is RM 50,000.

Project 1 :

Managed without using known methods whereby analyst was given their own initiatives to manage their own project to achieve the Case Study's objectives.

Project 2:

Managed with the proper control on documentation handling which was previously set according to INCOSE Requirements Engineering methodology.

Below are the requirements engineering common tasks tailored to INCOSE executable model, which had been given to the analyst who managed Project 2. The execution of the project 2 will follow the tasks according to the requirements engineering common tasks as previously discussed in chapter 2, item 2.5.

Task 1 : Planning.

The analyst is requested to file (The file will be labeled as INITIAL REQUIREMENT) all earlier requirements from various sources (user/buyer) and identify :

i. their owner

ii. Revision as A (The first revision is labeled as A, second revision labeled as B …)

iii. Date

Task 2 : Elicitation

Classify the requirements and its artifacts according to classification by RT Yeh et al (1984) :

a. Conceptual_Model

b. Data_Model

c. Data_Flow_Processing_Model

d. NonFunctional:Process

e. NonFunctional:Constraints

f. NonFunctional:Performance

g. NonFunctional:Economic

h. NonFunctional:Maintenance

.

Task 3 : Analysis.

Identify Requirements attributes as described by INCOSE Executable Requirements Management Model Interim Report (May 1998). The block (refer to figure 9) actually phases in Requirements Management and the INCOSE Executable model simplifies the block into INPUT, GATED STORAGE, ANALYSIS, REQUIREMENTS COLLECTION and IMPLEMENTATION OUTPUT. Due to hierarchical nature of the requirements, INCOSE Executable model also manages the sub-system requirements. This is being performed through the next block level and it is repeated to the lower level. Even the INCOSE Executable Model are the simplified version but the terminology used are more complex when compared to the others, therefore the block in INCOSE model is replaced with what identified as the common tasks to requirements engineering as discussed in chapter 2. The first level block is actually the planning block. The block which is replaced are : Formalization is to replace "Requirements Collection" and "Gated Storage" block, and Verification is to replace "validation loop block"; retaining the INPUT and ANALYSIS block from INCOSE. Each requirements for each hierarchy level will be given and attribute. The attributes is listed below :

a. Queror – Identification and revision of the block from which this requirement was sent back.
b. Destination - Identification of block where requirement is destined.
c. Ambiguity – An indication of how well stated the requirement is on a scale from 1 to 3: 1 (default) is a well stated requirement, 3 is a To Be Defined and 2 is a requirement somewhat ambiguous.
d. Criticality - Importance or certainty rating from 1 to 5: 3 (default) is normal requirement, 1 is nice to have and 5 is absolutely essential.
e. Risk – An indication of risk on a scale of 1 to 5: 3 (default) is normal risk and 5 is very high risk.
f. Cost – An indication of production cost on a scale of 1 to 5: 3 (default) is normal cost, 5 is very high cost and 1 is essentially free.
g. Immediacy – An indication of schedule on a scale of 1 to 5: 3 (default) is normal schedule,1 is due after most other requirements

   and 5 is due very early.
h. Complexity – *Design* complexity and an indication of number of

i. requirements expected to be derived from this one and the difficulty of the derivation on a scale of 1 to 5: 3 (default) is normal, 5 is very high complexity and 1 is simple.

j. Priority - A numerical rating based upon other attributes (and in real life, upon judgment) used to determine the order in which requirements will be processed by the Requirements Management system.

k. Verification – Verification method NONE, TEST, INSPECTION, ANALYSIS, HIGHER LEVEL and LOWER LEVEL.

Task 4 : Formalization.

Formalization is to transform the compiled documents in to presentable documents. The formalized document will be given to all users and management and a meeting to be conducted for document verification.

Task 5 : Verification.

The meeting was then held to verify the formalized requirement. Any changes to the requirements shall refer back to the INITIAL REQUEST and the necessary changes will be make according Task 1 – 4 again but with new file label as Version B and so on.

3.2.4 The Result of the evaluation of the Case Study.

The evaluation of the case study is performed by two independent persons.

Table 6 : Case Study Results.

	from 1st evaluator		from 2nd evaluator	
Characteristics\ Marks	**Project 1**	**Project 2**	**Project 1**	**Project 2**
Good Requirements - Standalone				
1. Necessary.	**3**	**8**	**5**	**7**
2. Concise (*minimal*, *understandable*).	**7**	**7**	**6**	**7**
3. Implementation free.	**5**	**5**	**5**	**5**
4. Attainable. (*achievable or feasible*).	**10**	**10**	**8**	**9**
5. Complete. (*standalone*)	**6**	**6**	**6**	**6**
6. Consistent.	**3**	**8**	**3**	**5**
7. Unambiguous.	**6**	**6**	**5**	**5**
8. Standard Constructs.	**5**	**5**	**5**	**5**
9. Words to avoid	**5**	**5**	**5**	**6**
10. Verifiable.	**6**	**10**	**6**	**7**
Good Requirements - Aggregate				
11. Complete.	**6**	**7**	**5**	**7**
12. Consistent.	**7**	**7**	**6**	**7**

Manageability				
1. Version Traceability	**1**	**8**	**4**	**8**
2. Supporting Document Accessibility	**2**	**8**	**4**	**7**
Objectives Achievement Marks				
Objectives 1	**57.5%**	**70%**	**60.83%**	**63.3%**
Objectives 2	**15%**	**80%**	**40%**	**75%**

Table 7 : The average Case Study Results.

The average results from both evaluator.

Objectives Achievement Marks		
Objectives 1	**59.17%**	**66.65%**
Objectives 2	**27.5%**	**77.5%**

3.2.5 Findings on the Case Study.

Based on the results, we found that :

a. Project 2 is producing better characteristics of good requirements.
b. Project 2 is producing better requirements manageability.

Hence, comparing between the project where the manager was given a brief understanding of the objectives and the criteria to achieve the objectives of “Good and Manageable Requirements”; with the project,

where the manager was requested to follow strict guidelines according to the common task of requirements management with model adapted from INCOSE, the second produced better result. We can therefore conclude that the INCOSE methodology is capable of enhancing the characteristics of good requirements and also enhancing manageability of documentation.

To achieve 'Well define' requirements, the second project manage to achieve 66.65% marks as oppose to the first project 59.17% marks.

To achieve Manageability requirements, the second project manage to achieve 77.5% marks as oppose to the first project 27.5% marks.

3.3 Summary

Base on our samples from various type of industry and various organization maturity level in Malaysia, not many organizations practice established standards in Requirements Engineering methodology. Most of the immature organizations do not have any kind of practicing standards whereas some of the mature organizations may have their own sets of standards. To have more realistic samples, It is good if we can have the international software development's companies such as IBM, Microsoft, SUN, CSA etc to be part of our samples but those companies do not want to reveal their management process to the

outsider. This is the reason why we do not have those established international companies in our sample.

In the United States, many organizations have developed their own sets of standards, to be used in their organizations. For example: SLATE by Texas Instrument and US Naval Requirements Management by US Navy. The organization later develop a tools and later on INCOSE as a research institute develop their model base on the available tools. Hence, it is explainable why INCOSE model are more complete.

The Case Study has shown that using the INCOSE methodology, the requirements produced is graded higher in terms of the mentioned characteristics.

Most of the methods in Requirements Engineering are not easy to implement without the help of tools. Most of software engineers follow the organizations standard procedures in documenting the requirements. The implementation of the methodologies are always a problem when it does not involve top level management.

In the event of immature organizations, the top level management is not really concerned about the methodology in Requirements Engineering. Many of the companies in Malaysia belong to this group and many of the software

development project's failure is due to lack of requirements (INCOSE, 1996), thus we foresee the need for a model of one standard methodology to be used for these kinds of organization.

The simple model and tools will enable analyst and software engineers to use it on a personal capacity if it is not an adapted standard in their organizations. The model shall not be too complicated in order to support individual usage, and it shall have tools to support it. The integrated tools environment has to be developed to enable tools that are currently used by the analyst and software engineers to be integrated together.

Chapter 4

Proposed Model Enhancements

4.1 Introduction and background.

The table below shows the weight given for each methodology reviewed in Chapter 2 against the tasks or processes involved in a typical Requirements Engineering methodology. The weight given is based on the availability and comprehensiveness of the documentation in discussing the Requirements Engineering processes. The weight score is from 1 to 10 whereby 1 is for the lowest score and 10 for the highest score. For each reviewed methodology we have also identified tools supporting it.

Table 8 : Requirements Engineering Processes

– level of comprehensiveness by reviewed methodology

METHODOLOGY\ Task or Process	SLATE By Goodwin S. and Nellon J. (1999)	US NAVAL by US Naval RMWG (1999)	INCOSE by INCOSE (1997)	RUP by Rational Software Corporation (2001)
1. Planning	6	8	1	3
2. Elicitation	1	1	1	8
3. Analysis	5	5	8	8
4. Formalization	1	7	1	8
5. Verification	1	5	3	1
Artifact Management	5	5	5	8
Model	Classified as simple. Refer to Figure 4.	Classified as simple. Refer to Figure 8.	Classified as comprehensive. Refer to Figure 9.	Classified as comprehensive.
Tools	SLATE	Not Available	A few tools supported, SLATE is one of it.	Requisite Pro

Hence for the purposes of the development of the new Requirements Management tools, we will combine the strength from various methodology to come out with a new enhanced methodology. The new tools shall be developed based on the enhanced methodology.

We would like to propose a new enhanced Requirements Engineering model to adapt the strength from all the previously reviewed models. We also recommend that for every stage of Requirements Engineering processes, the mechanism to manage the artifact and traceability according to the RUP (2001) framework shall be provided. Even though RUP is the most comprehensive methodology we had reviewed, it is not recommended to be used on its own, without considering using RUP as a whole. RUP is one complete package and not easily tailored (Henderson Sellers B. et al, 1999). RUP require software engineers to understand the whole RUP functionality and techniques. Hence this is impractical for the small scale organization. Since RUP address the issue brought by Royce Walker (1999) on artifacts management, we suggest that the artifacts management functionality to be adapted from RUP.

Table 9 : The proposed adaptation for new enhanced Requirements Engineering methodology.

Requirements Management Processes	Methodology to be adapted from.
1. Planning	USNAVAL
2. Elicitation	USNAVAL
3. Analysis	INCOSE
4. Formalization	USNAVAL
5. Verification	USNAVAL
Artifact Management	RUP

We would like to incorporate the artifact management mechanism and traceability into the typical Requirements Engineering methodology. Hence the methodology shall be as presented in Figure 11.

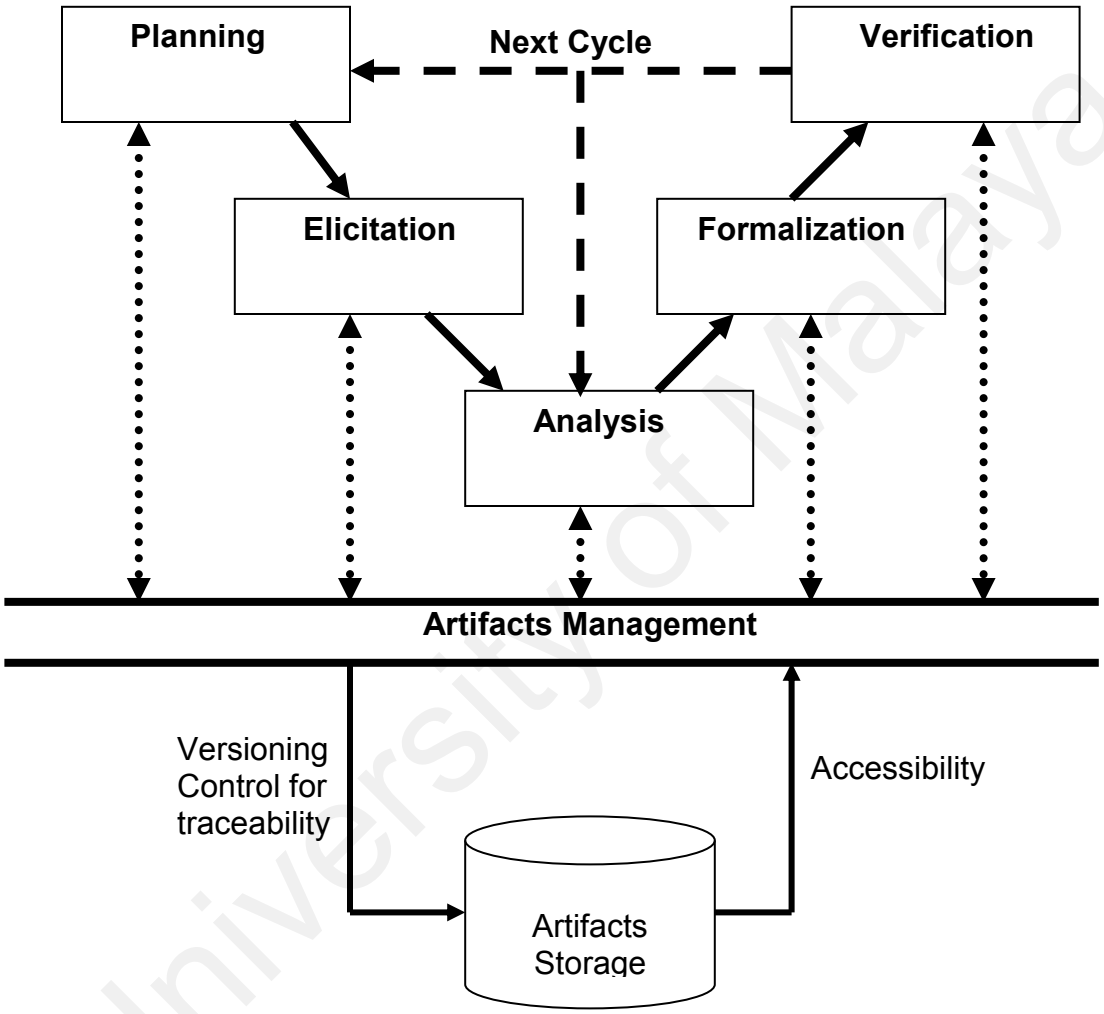


Figure 11. Requirements Engineering Methodology with Artifacts Management.

4.2 Requirements Engineering Model

The model of the Requirements Engineering Methodology, as proposed by INCOSE (1997) and RUP (2001) is a comprehensive and complete model. The INCOSE model is not much different from that proposed by SLATE, in fact the development of INCOSE model is an improvement of SLATE. RUP (2001) as we had previously discussed is not advisable to implement without adapting the whole RUP development method. The only good feature of RUP is the artifact management. In the previous chapter we had conducted a case study using INCOSE method, without tools supporting it.

INCOSE underlying idea encompassed the management of standard requirements and also covers the composition of the requirements. The main reason to maintain the requirements composition, is that most of the system are normally composed of other sub-systems.

Project managers, who are managing the Large Scale Software Project Development will find this INCOSE Requirements Engineering Methodology Analysis phase very useful, but in most immature organization especially in Malaysia it is probably too complex. From the Industrial Experts questionnaire feedback we have identified the needs to simplify the model to meet the industrial use for practicality. Fortunately INCOSE (may 1998) introduce a simplified model which has been called INCOSE executable model. Figure 13 is an INCOSE executable model.

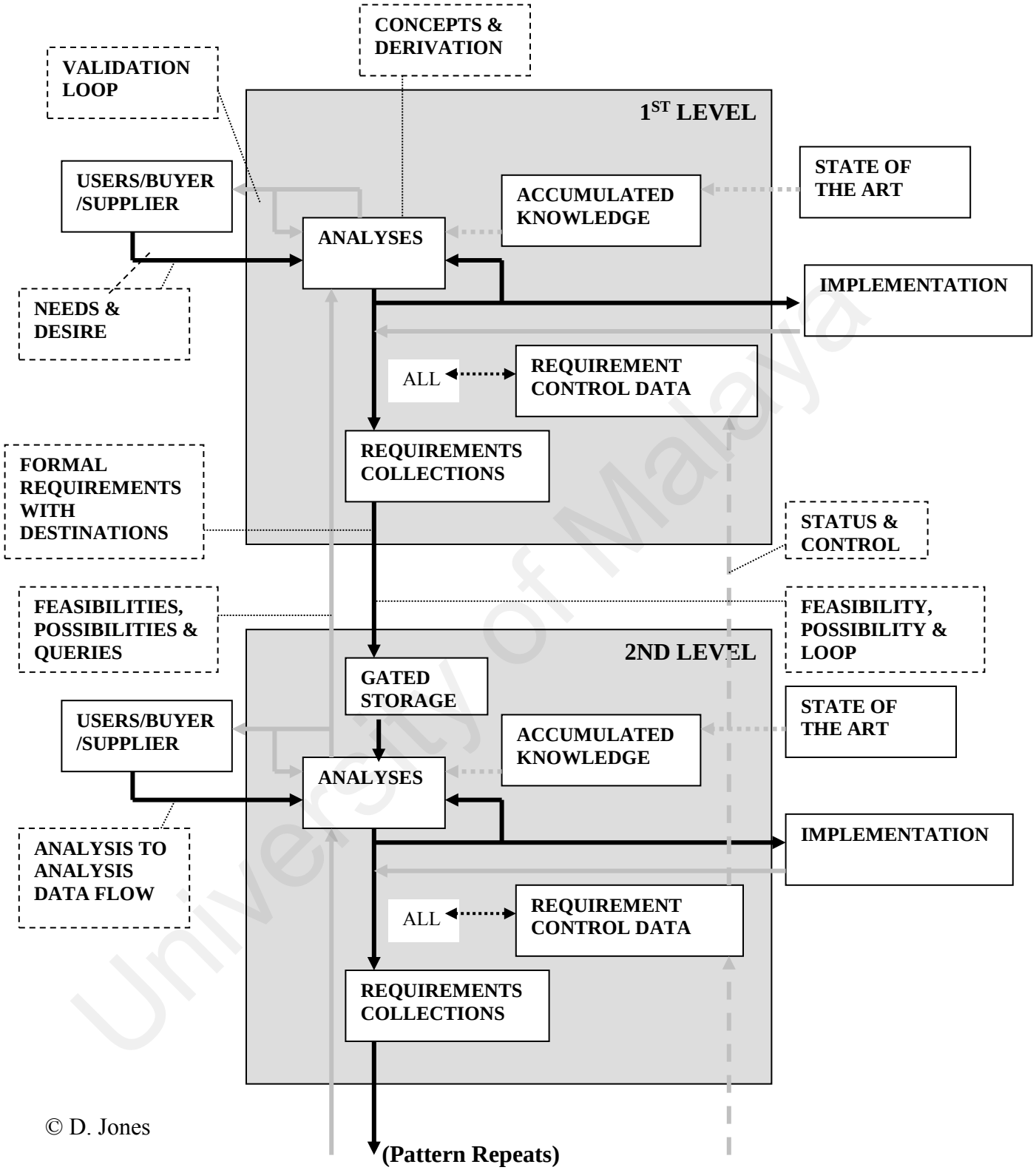


Figure 12. INCOSE complete model

4.2.1 The INCOSE Executable model (INCOSE, May 1998)

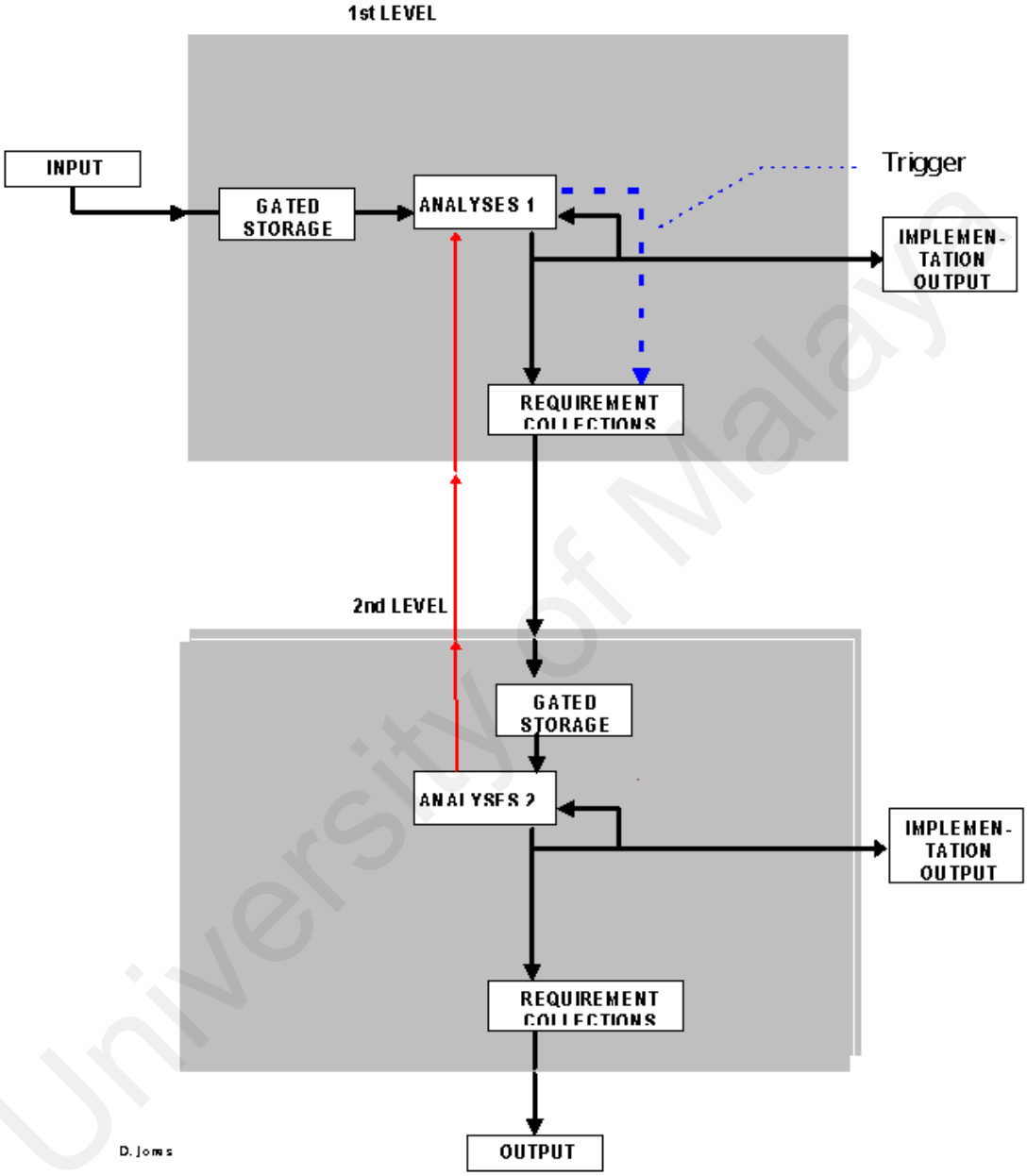


Figure 13. INCOSE Executable model

In this proposed executable model, the changes from and similarities to the previous diagram (figure 12) include:

- The Accumulated Knowledge and State of the Art blocks are gone. Their effect is incorporated in the parameters of the Analyses blocks.

- The Requirement Control Data blocks are gone. The control parameters will reside in parameters in the blocks, and in attributes attached to data packets and described in part below. Added to the previous diagram is a trigger signal indicating that the analyses for the level is sufficiently complete to update the requirements collection to a new revision.

- The feedback loops around the Analyses blocks are kept. There is provision for requirements passing from analysis to analysis on the same level.

- The multilayer structure of the second level is retained. There are multiple Analyses and Requirements collections at the second level.

- The Implementation blocks still exist, but there is no feedback of queries from them.

As previously mentioned, this model simulates the flow of requirements through an organization or process with the requirements being represented by data packets containing attribute data.

Possible data flow is:

a. From Input into the first level, an initial set of requirements. (In a later development of the model these may be subdivided into
    i. Architecture/Concept Requirements (elements),
    ii. Performance Requirements,
    iii. Constraint Requirements (cost and schedule))
b. From Analysis 1 to Requirements collections to Gated Storage to Analysis 2 to the output:
c. Derived Requirements
d. From the Analysis 2 to Analysis 1:
e. Requirements with changed Risk / Feasibility attributes.

4.2.2 The problems with the INCOSE Executable model

The following is the summary of the problems with the INCOSE Executable model as previously discussed.

a. The model does not describe clearly the common tasks in requirements management. There is no step by step guide for the INCOSE processes or tasks.
b. The model does not describe clearly on artifacts management. It does have a versioning control but it does not provide mechanism for artifact and traceability like RUP.

4.2.3 The new Enhanced Requirements Engineering model

The following is the proposed changes necessary to the INCOSE executable model in order to make it more practical for industries in Malaysia :

a. Reorganize the process block according to well known terminology as commonly used by developers. The block represents phases in Requirements Management and INCOSE Executable model block are INPUT, GATED STORAGE, ANALYSIS, REQUIREMENTS COLLECTION, IMPLEMENTATION OUTPUT. The block will be replaced by what has been identified as common task to requirements engineering (refer to 2.5). The first level ANALYSIS block will be broken down into PLANNING, ELICITATION and ANALYSIS. The FORMALIZATION block is to replace REQUIREMENTS COLLECTION and IMPLEMENTATION OUTPUT. We still maintain VERIFICATION

LOOP as what appear in the initial INCOSE model since we believe this loop is important.

b. To incorporate version control as a process to handle changes especially when the project is applying an iterative development. REVISION block is introduced to show clearly on the model on how to handle requirement revision.

c. To introduce the artifacts management block which enable each stages in Requirements Management Analysis processes to communicate (save the current document or refer back to the previous stage/version of documents).

4.2.4The proposed Requirements Engineering model

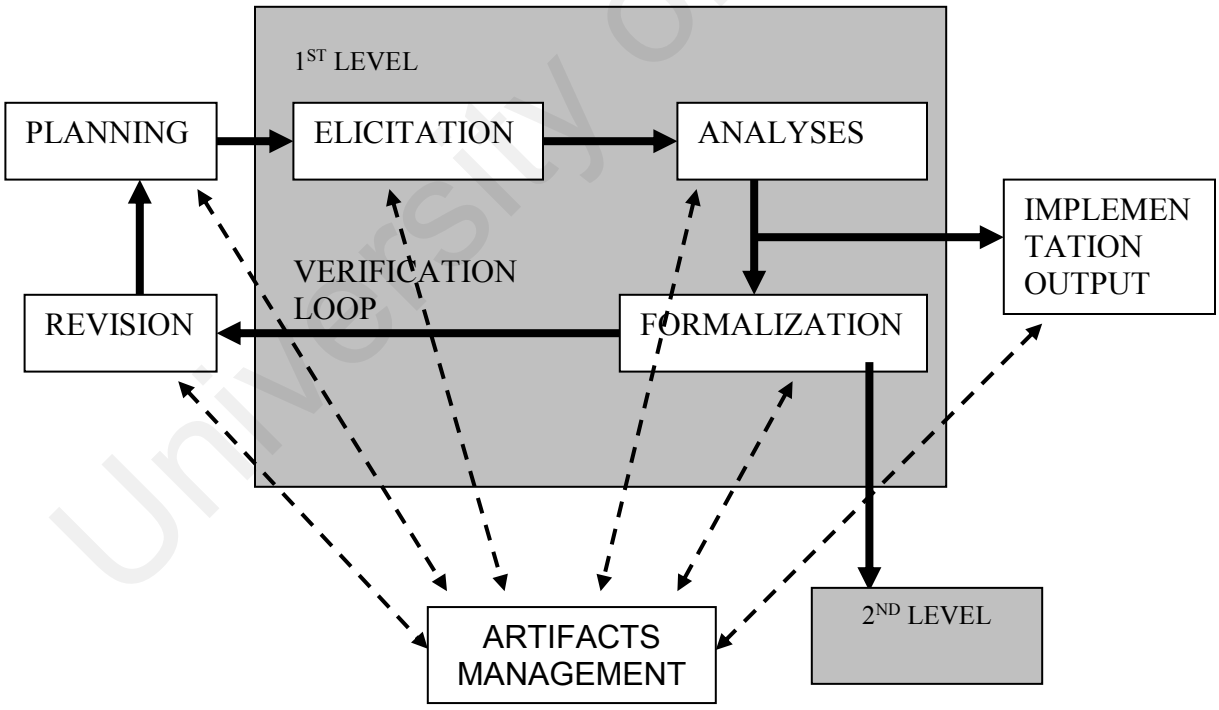


Figure 14.

The Proposed Requirements Engineering model.

4.3 Discussion on the new model

We will discuss the new model from two aspects – managing requirements tasks or processes and also managing the artifacts. On managing requirements task comparing to INCOSE model the new model is simpler and had treated "REVISION" as a different process from "PLANNING" and "ELICITATION" as a different process from "ANALYSIS". This will allow those who understand the basic of requirements management processes to understand the model easily, since those processes are the common process or task found in requirements management methodology. The new model also shows clearly on artifacts management process which most of the model we had discussed before do not have the feature. On the methodology, we recommend to adapt USNAVAL methodology. USNAVAL describes the processes in the form of tasks list, which are easy to understand and implement. RUP together with Rational's RequisitePro as a requirements management tools, probably is the most complete methodology but it is designed as an Object Oriented development methodology, hence it is not suitable for common use especially those analyst who are inexperienced in object oriented development. We will adapt only the RUP artifacts management processes since we believe the artifacts management is very important and RUP had discussed thoroughly on the subject.

We believe that, the new model is suitable for the beginners and immature organizations which they might not have a capacity to understand complex

development methodology such as RUP. This model is simple but complete. This will improve the usability of the tool which we will develop base on the new model.

4.4 Summary

The aim of our research is to develop a more practical model that can be used by any developer or project manager to produce well defined requirements and to assist in the manageability of the requirements. The new proposed model, is actually proposed based on the established INCOSE executable model with changes according to feedback from the Industrial experts questionnaire and also the findings of the case study. The Requirements Engineering model is to describe the underlying methodology but complete documentation on methodology is required before tools can be developed.

Chapter 5

Requirements Engineering

Tools Prototype Specification

5.1 Requirements Engineering Tool Development : Objectives and background

In chapter 1, among our thesis objectives is to propose an applicable model of the Requirements Engineering phase for Malaysian software industry and to develop a prototype for the proposed model.

In chapter 4, we had discussed and proposed the new enhanced Requirements Engineering methodology and model, and before that in chapter 2 we had discussed on various aspect and functionality of Requirements Management tool. The Objectives of this prototype project are :

a. To develop a Prototype for Requirements Management Tool - Upper CASE tool (INCOSE 1998) focusing on Requirements Specification and Analysis.

b. To adapt a modified version of requirements methodology and model as specified in Chapter 4.

5.2 The new Requirements Engineering methodology and model.

Figure 15 and 16 below depict the newly improved methodology and model for the Requirements Engineering as previously discussed in chapter 4.

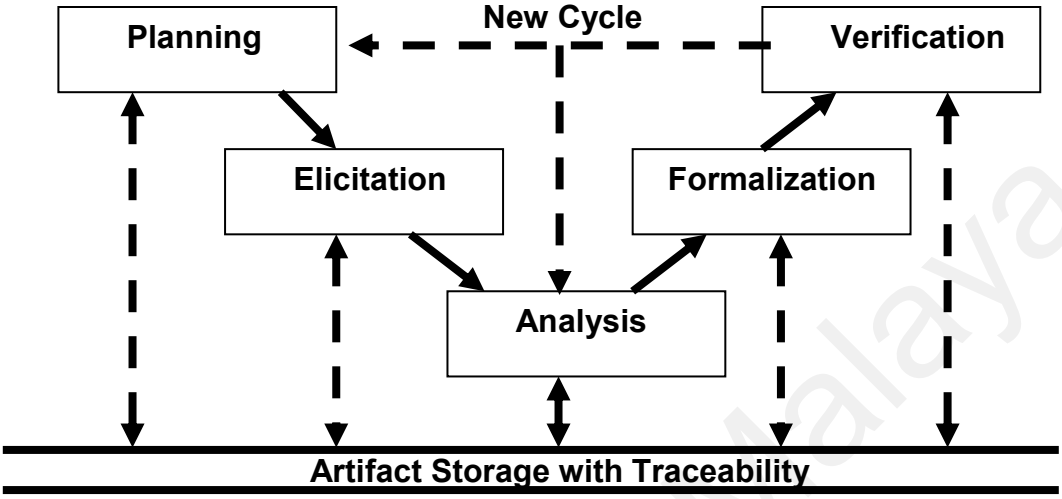


Figure 15. Requirements Engineering Methodology with Artifact Management.

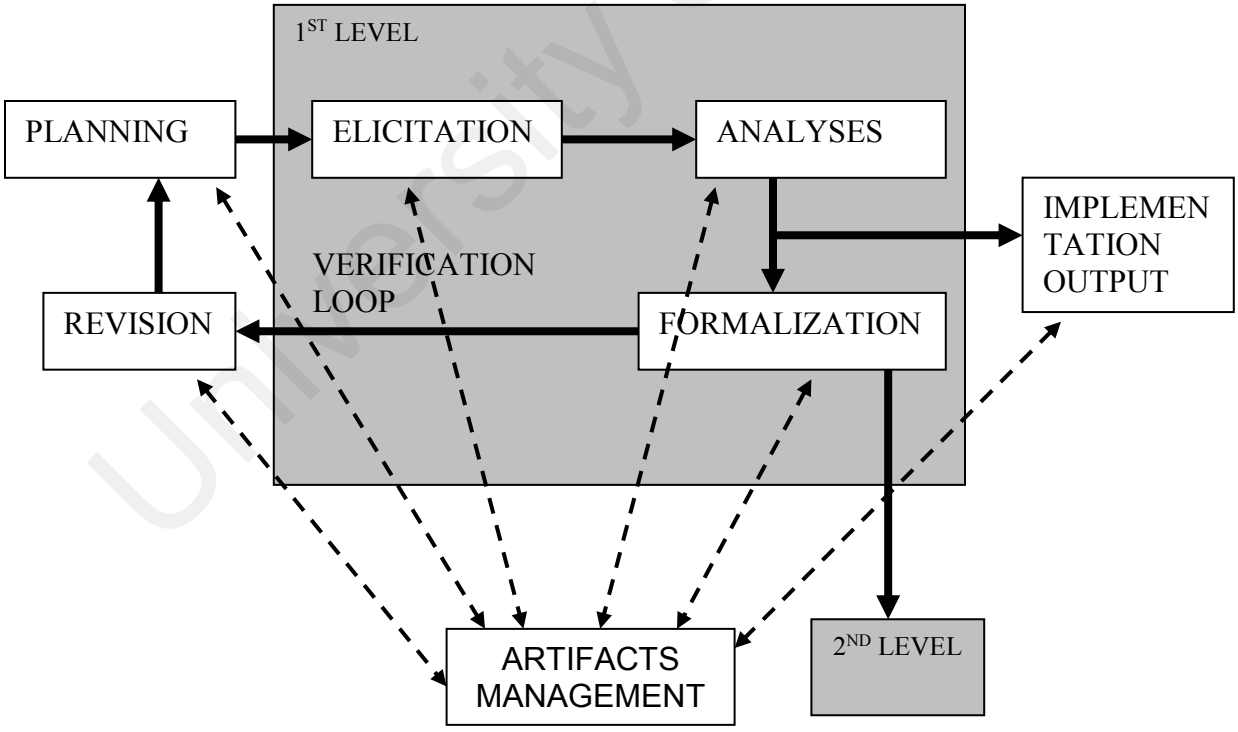


Figure 16. Propose Requirements Engineering Model with enhancement.

From the methodology and model shown in Figure 15 and 16 we can conclude that, the proposed Requirements Engineering tools shall consist at least five sub-systems which are : Planning, Elicitation, Analysis, Formalization and Revision.

Apart from building an application from the model (figure 17), we also need to know the detail processes or tasks involved in requirements management. We had adapted USNAVAL documentation on detail steps involved in each major task of requirements engineering since this is the most comprehensive material we manage to find. Below is detail step for each common task in requirements engineering according to USNAVAL.

The requirement for the planning phase is to adapt from USNAVAL. Figure 17 below visualised USNAVAL processes or tasks involved during planning.

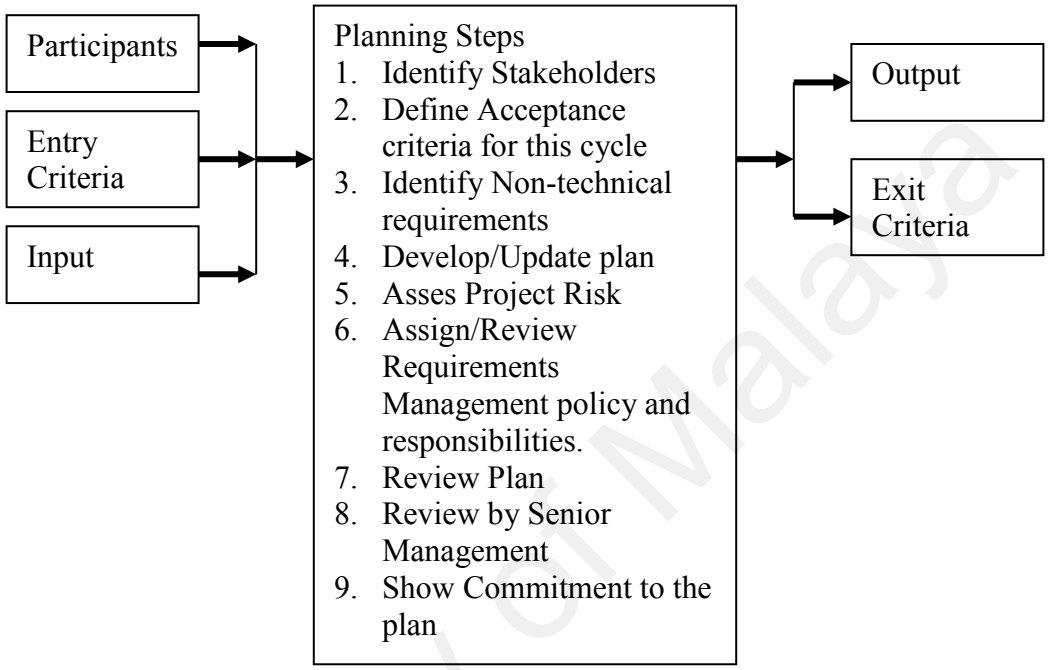


Figure 17. US NAVAL Requirements Engineering Steps

The requirement for the elicitation phase is to adapt from USNAVAL. Figure 18 below visualised USNAVAL processes or tasks involved during elicitation.

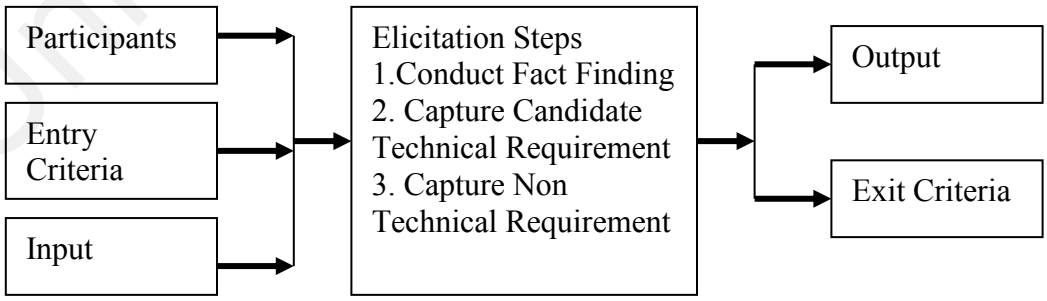


Figure 18. Elicitation Process

The requirement for the anlysis phase is to adapt from USNAVAL. Figure 19 below visualised USNAVAL processes or tasks involved during analysis.

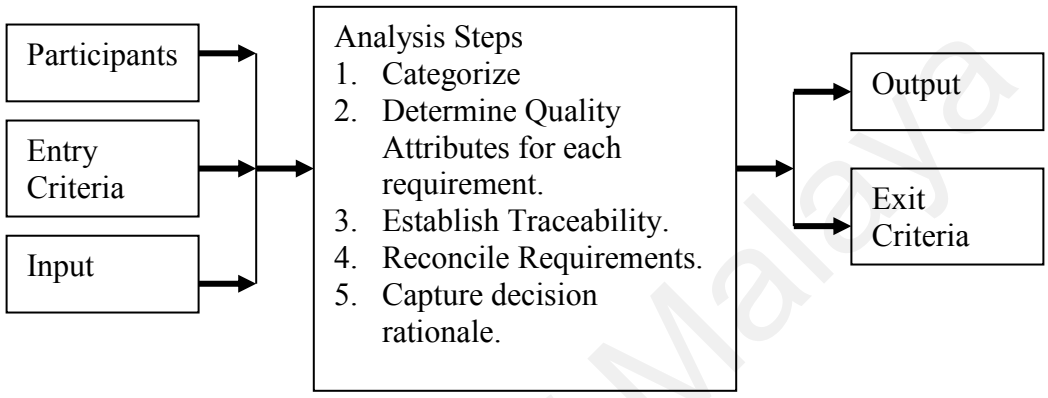


Figure 19. Analysis Process

For step no 2, to determine quality attributes for each requirement, we adapt INCOSE. INCOSE described requirements attributes more comprehensive.

According to INCOSE (1997), the attributes of the requirement are as listed below :

(1) ID

(2) REVISION

(3) QUEROR

(4) DESTINATION

(5) AMBIGUITY

(6) CRITICALITY

(7) RISK

(8) COMPLEXITY

(9) COST

(10) IMMEDIACY

(11) VERIFICATION

The first two attributes, ID and REVISION, uniquely identify a particular requirement, and when combined with the next two attributes, QUEROR and DESTINATION, locate it in the flow of requirements. The next seven attributes, AMBIGUITY, CRITICALITY, RISK, COMPLEXITY, COST, IMMEDIACY, and VERIFICATION are flags to requirements which in real life require special attention. The first four of these describe the requirements own characteristics, and the last three describe the characteristics of the related budget, schedule, and verification plan requirements. The VERIFICATION attribute is a flag indicating the verification method. The numerical values for these attributes are chosen so that higher numbers indicate a greater need for attention.

The requirement for the formalization phase is to adapt from USNAVAL. Figure 20 below visualised USNAVAL processes or tasks involved during formalization.

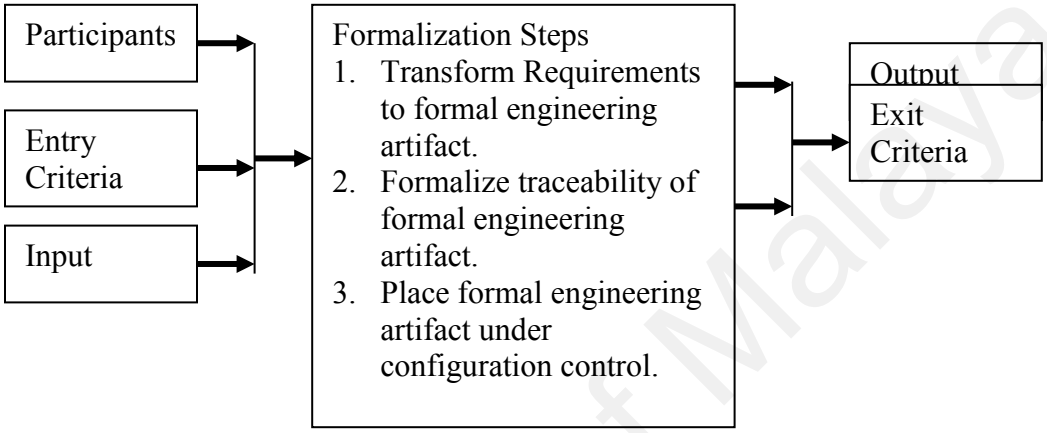


Figure 20. Formalization Steps

The requirement for the verification phase is to adapt from USNAVAL. Figure 21 below visualised USNAVAL processes or tasks involved during verification.

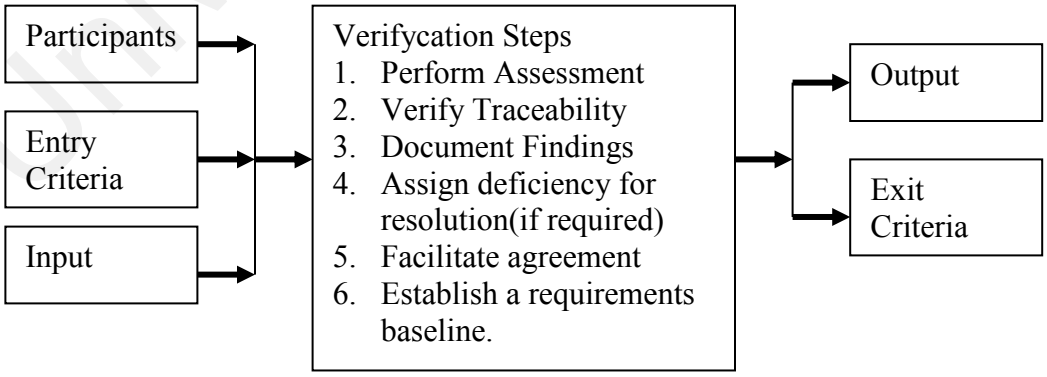


Figure 21. Verification Steps

5.3 Non Technical Requirements

This is where the non technical aspects of the requirements such as response time, overall budget, hardware, security and other is being assesed.

a. Requirement Flow down : This function is to enable managing sub-system requirement.
b. Traceability Analysis: This function will enable document versioning management.
c. Document & Other Output Media : This function will enable to generate, store and publish document.
d. Groupware : This function will enable Project Working Committee to collaborate with each other.
e. Interface to other tools : This function will enable transferring data from one function or tool to another.
f. System Environment : This function is to define System environment requirements.
g. Support & Maintenance : This function is to define Support and Maintenance structure.
h. Artifact Management. : This function is to manage requirements artifact.

5.4 Analysis and Design.

5.4.1 Methodology.

We will apply the Object Oriented Software Engineering (OOSE) methodology, which was developed by Objective System of Sweden (Wilkie G, 1997; p 120). The methods works with five different model :

a. The requirements model – which captures functional or technical requirements.

b. The analysis model – which concentrates on developing object structure.

c. The design model – which refines the object structure to suit the current implementation environment.

d. The implementation model – which is used in implementing the system.

e. The test model – to verify the system.

The model will be represented using Unified Modeling Language (UML), please refer to UML Summary (Rational Software Corporation, 1997).

5.4.2 The Requirements Model

The requirements model captures technical or functional requirements by use case modelling (Wilkie G, 1997).

5.4.2.1 Identifying Actor.

The actor will be created from the participants identified in technical requirements. (Refer 5.2.1). The participants identified as :

a. Senior Management (SM) – Those who have the authority to decide on resources especially human and fund.
b. Customer – Those who will fund the project.
c. Project Manager(PM) – Those who will manage the overall project. If PM are not capable of being as Requirements Engineer (RE), he or she must be assisted by RE.
d. Systems Engineer (SysEng) – Those who will provide the Operating System and Hardware environments.
e. Software Engineer (SoftEng) – Those who will developed the system from the Analysis and Design specification produced by RE.
f. Configuration Manager (CM/QA) – Those who monitor and manage the changes and revision.
g. Functional Testing (FT) – Those who will be involved in testing system functions.

h. Operational Testing (OT) – Those who will be involved in testing system operation.

i. End User - Those who will use the system.

5.4.2.2 Requirements Model.

The Requirements Engineering model as shown in Figure 21 describe that, the Requirements Engineering tools shall be in five subsystems, hence the model will be presented from the general model of requirements engineering and also the specific subsystems requirements :

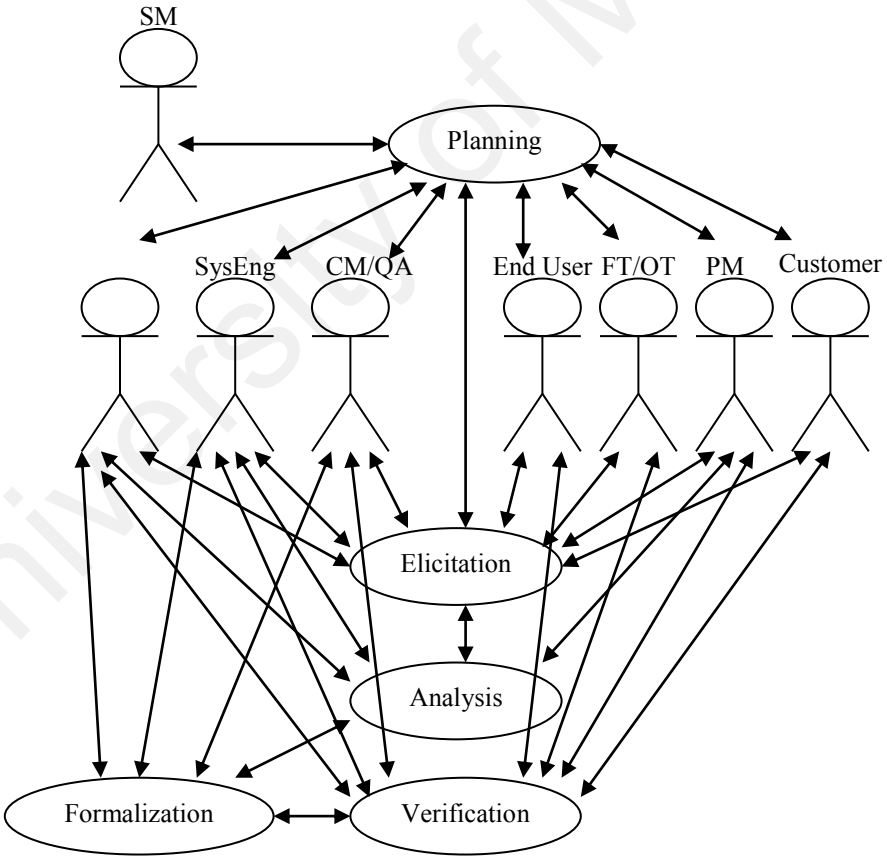


Figure 22 : Requirements Model - Requirements Engineering

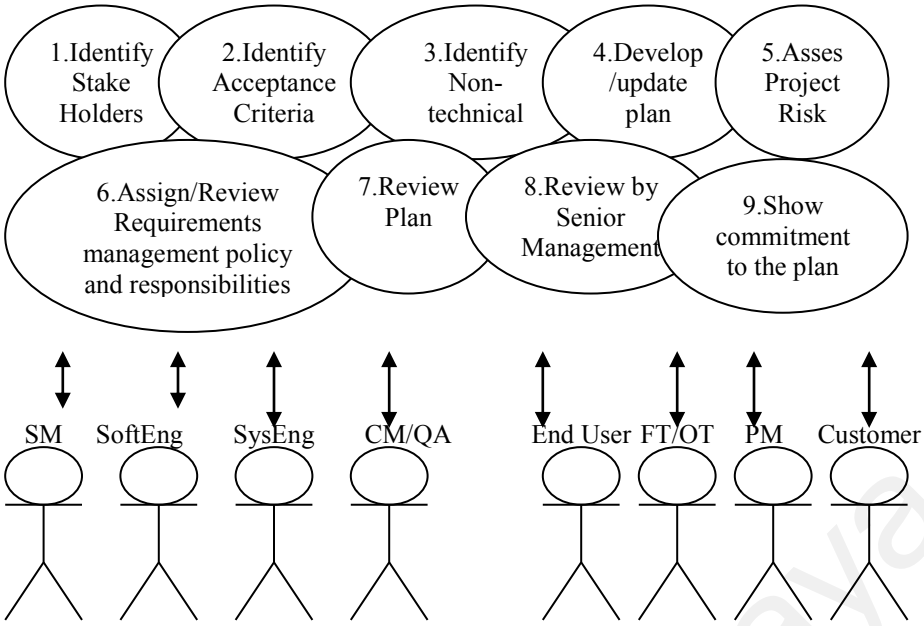


Figure 23 : Requirements Model - Requirements Engineering, Planning Subsystem.

.

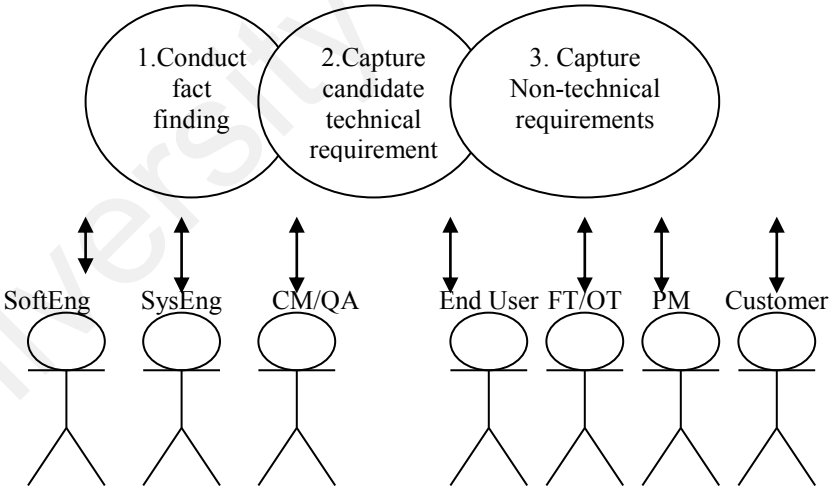


Figure 24 : Requirements Model - Requirements Engineering, Elicitation Subsystem

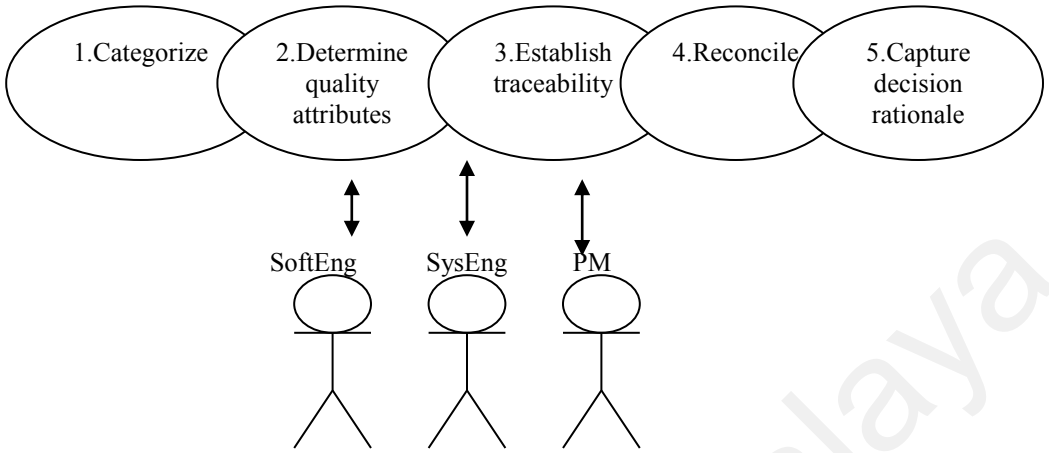


Figure 25 : Requirements Model - Requirements Engineering, Analysis Subsystem.

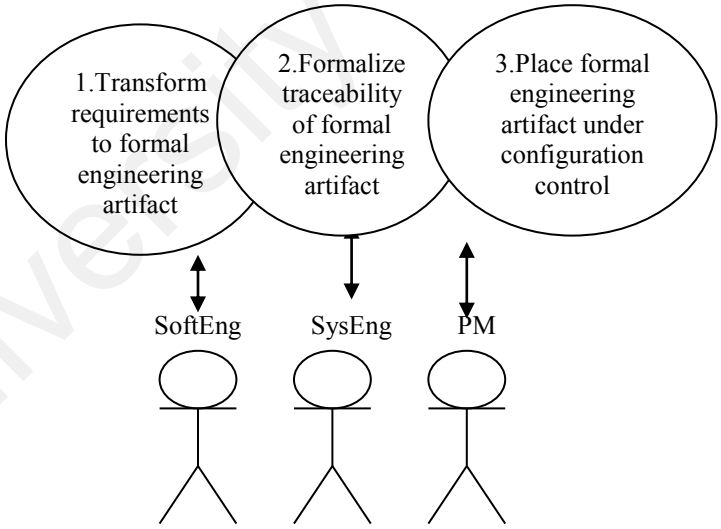


Figure 26 : Requirements Model - Requirements Engineering, Formalization Subsystem

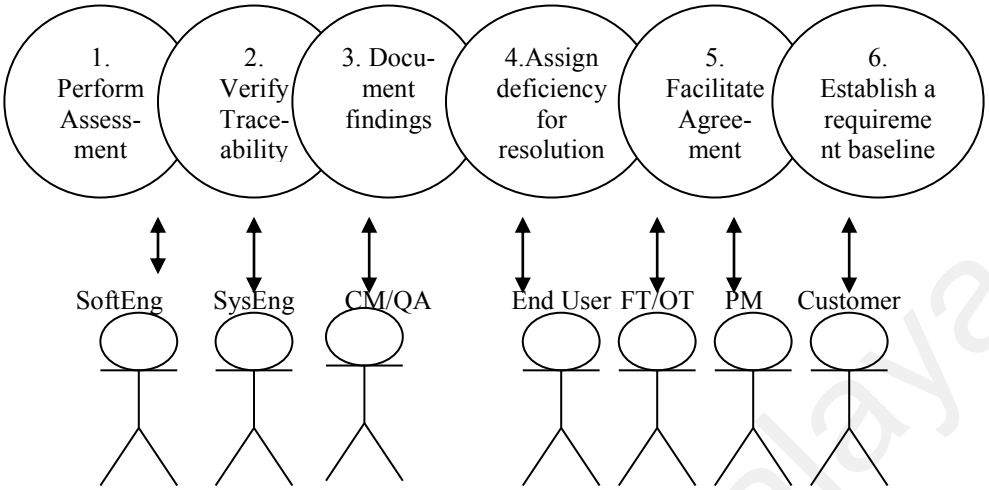


Figure 27 : Requirements Model - Requirements Engineering, Verification Subsystem.

5.4.3 The Analysis Model

In this stage the use cases are refined and made more detailed and robust.

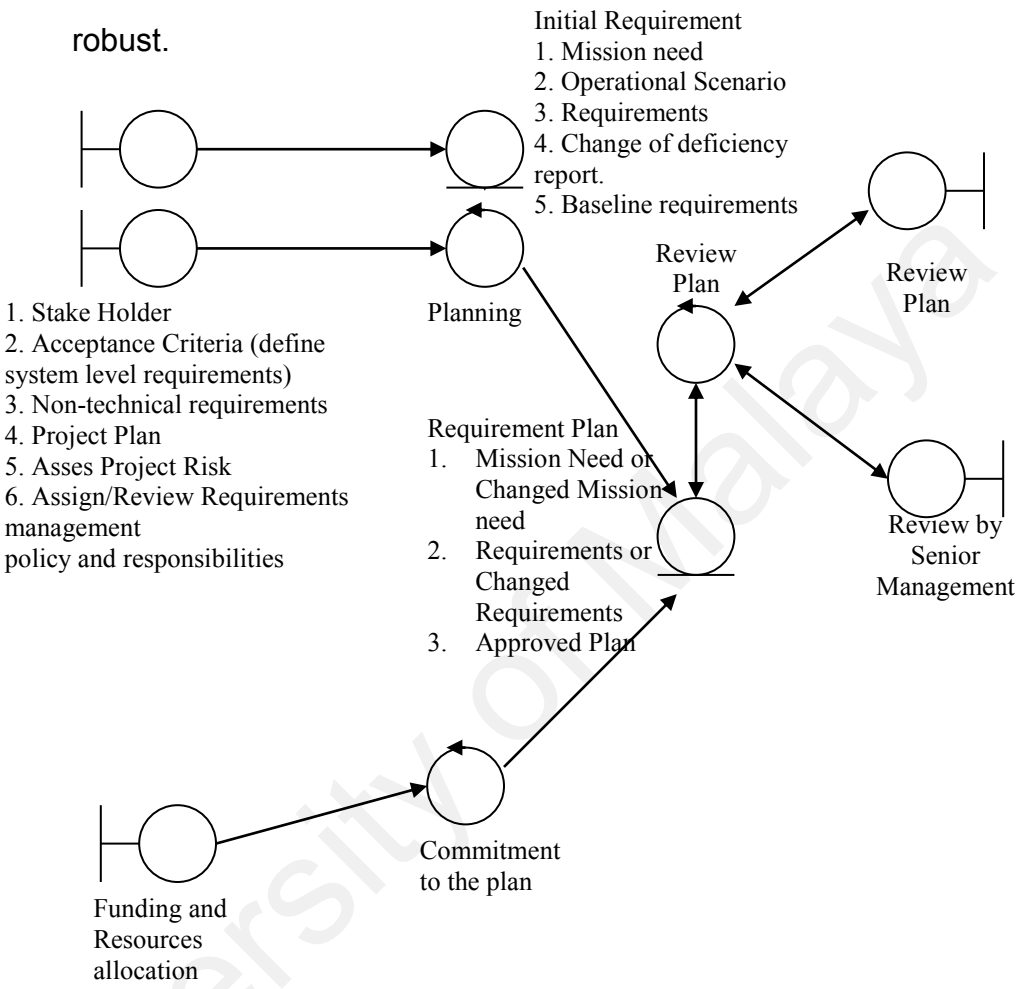


Figure 28 : Analysis Model - Requirements Engineering, Planning Subsystem

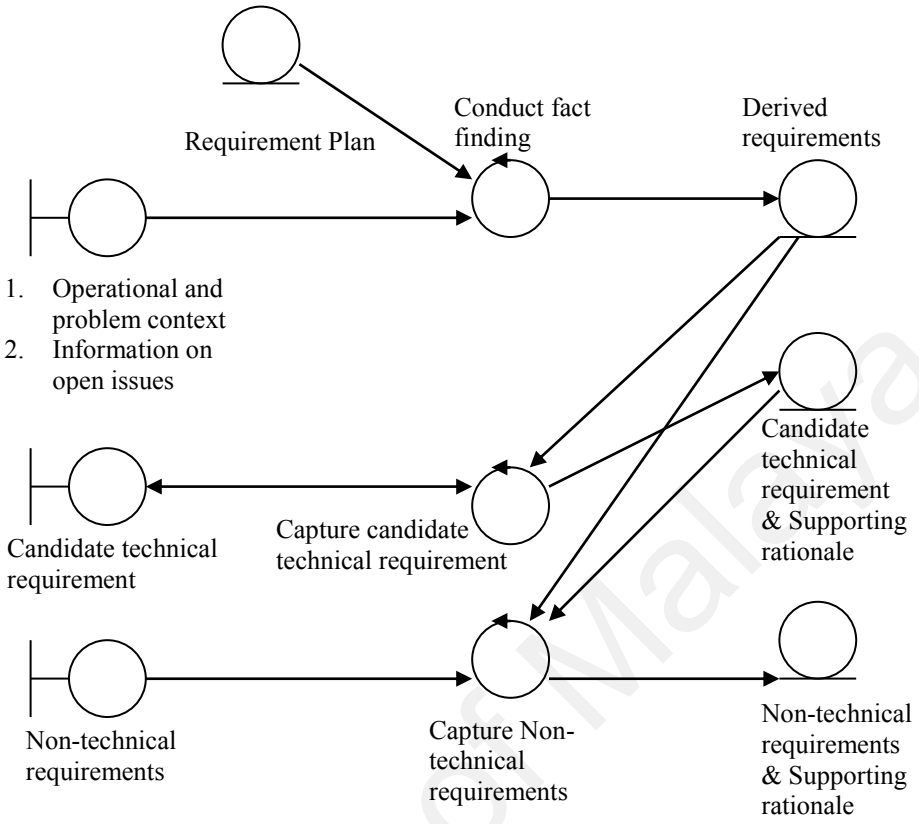


Figure 29 : Analysis Model - Requirements Engineering, Elicitation Subsystem

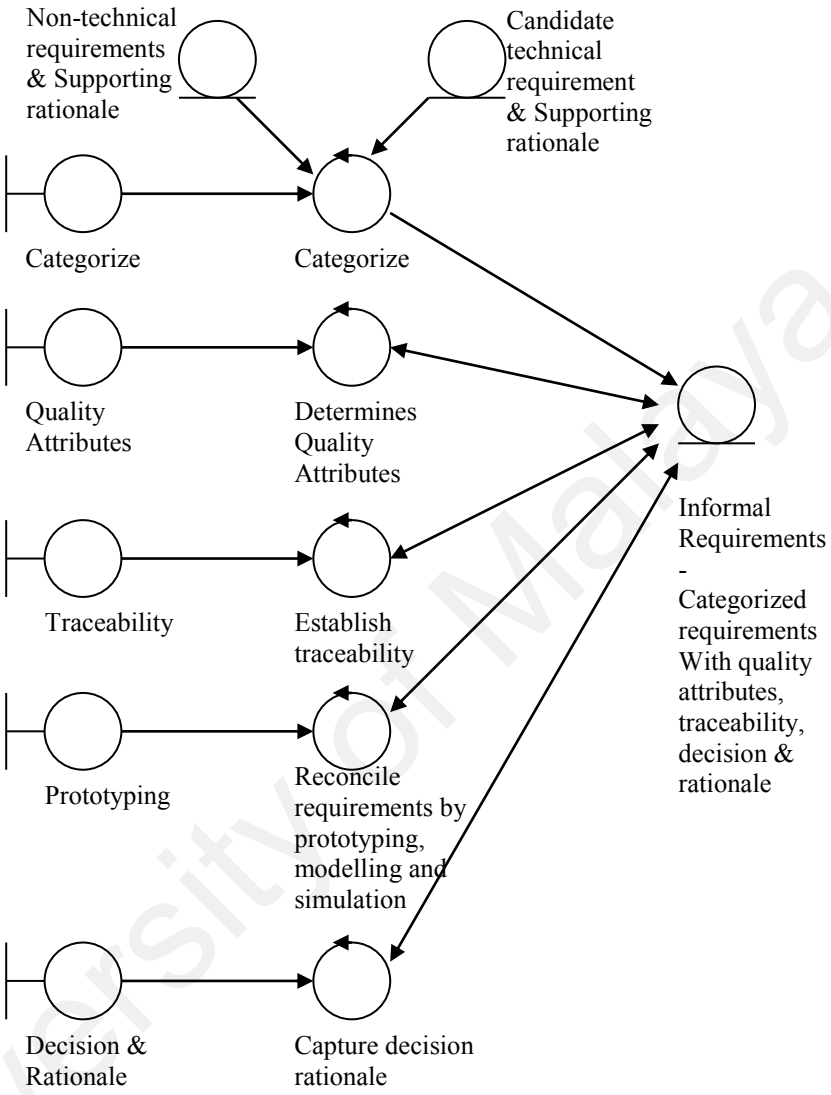


Figure 30 : Analysis Model - Requirements Engineering, Analysis Subsystem

.

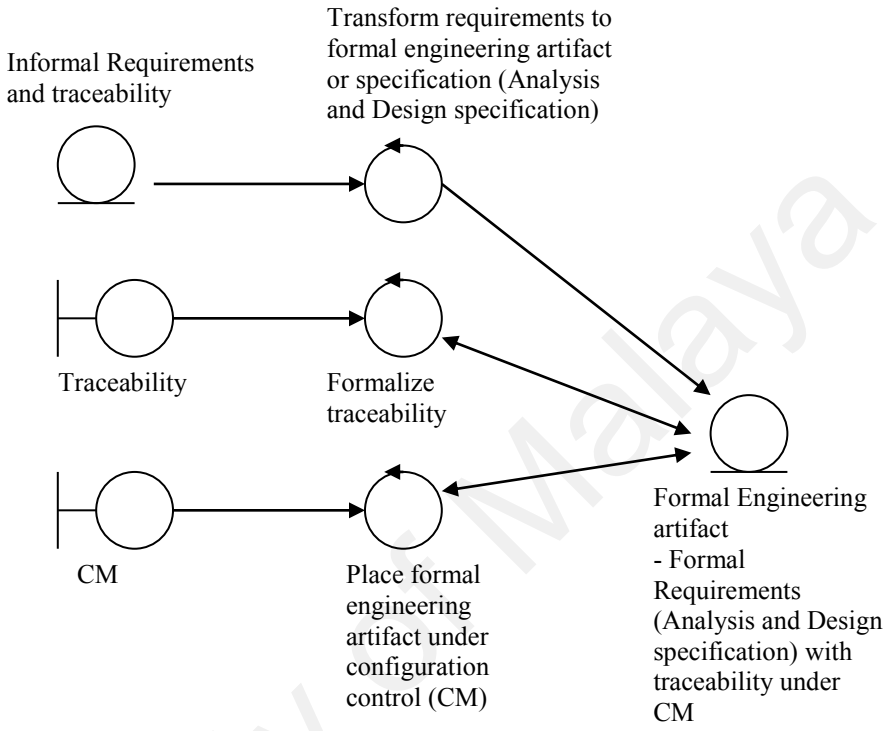


Figure 31 : Analysis Model - Requirements Engineering, Formalization Subsystem

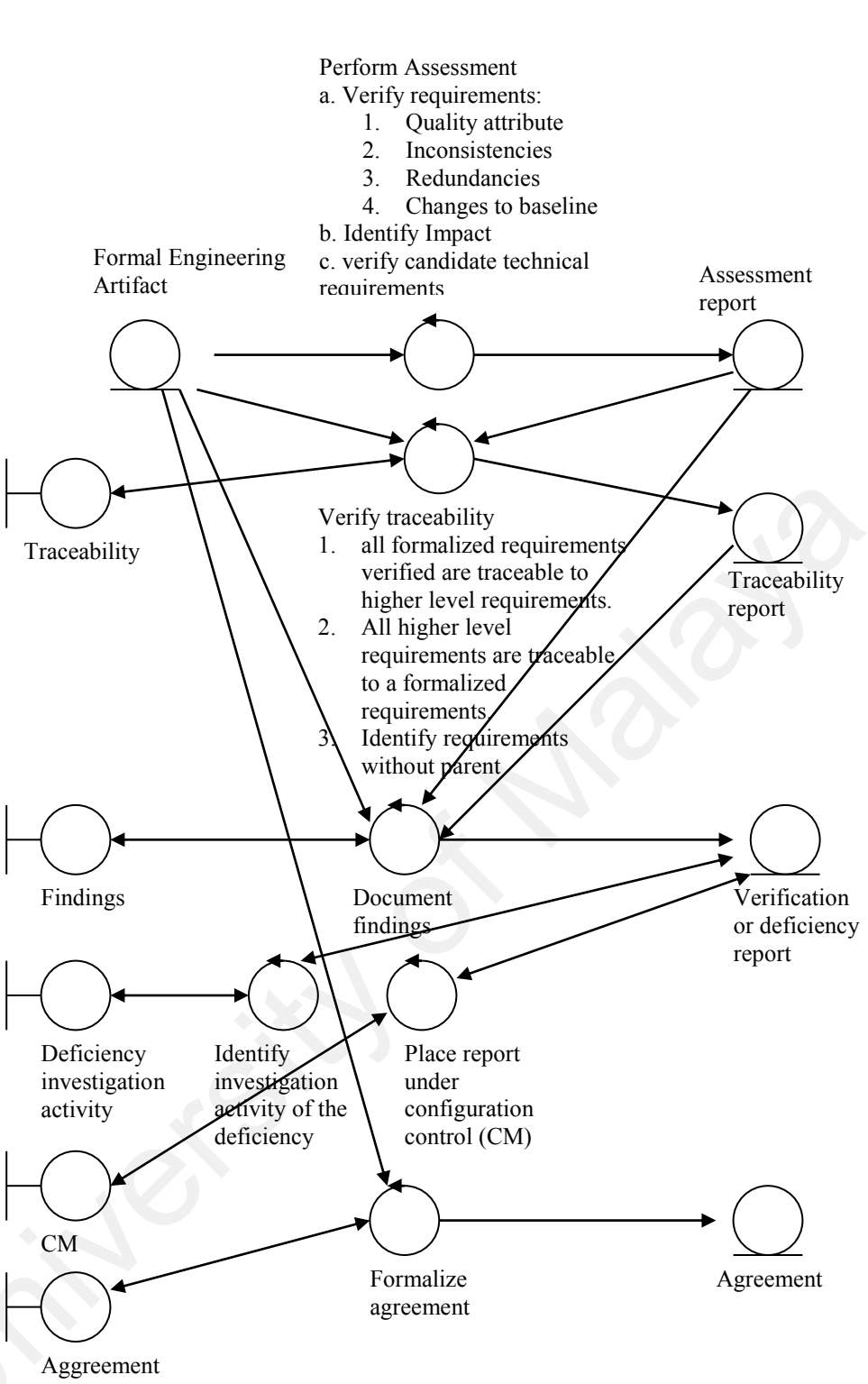


Figure 32 : Analysis Model - Requirements Engineering, Verification Subsystem.

5.4.4 The Design Model

The design model (see figure 33, 34, 35, 36 and 37) defines the implementation, it serves as a higher level view of the source code. It consists of design classes (and types) and design subsystems (Jacobson I. et al, 1997; p. 75).

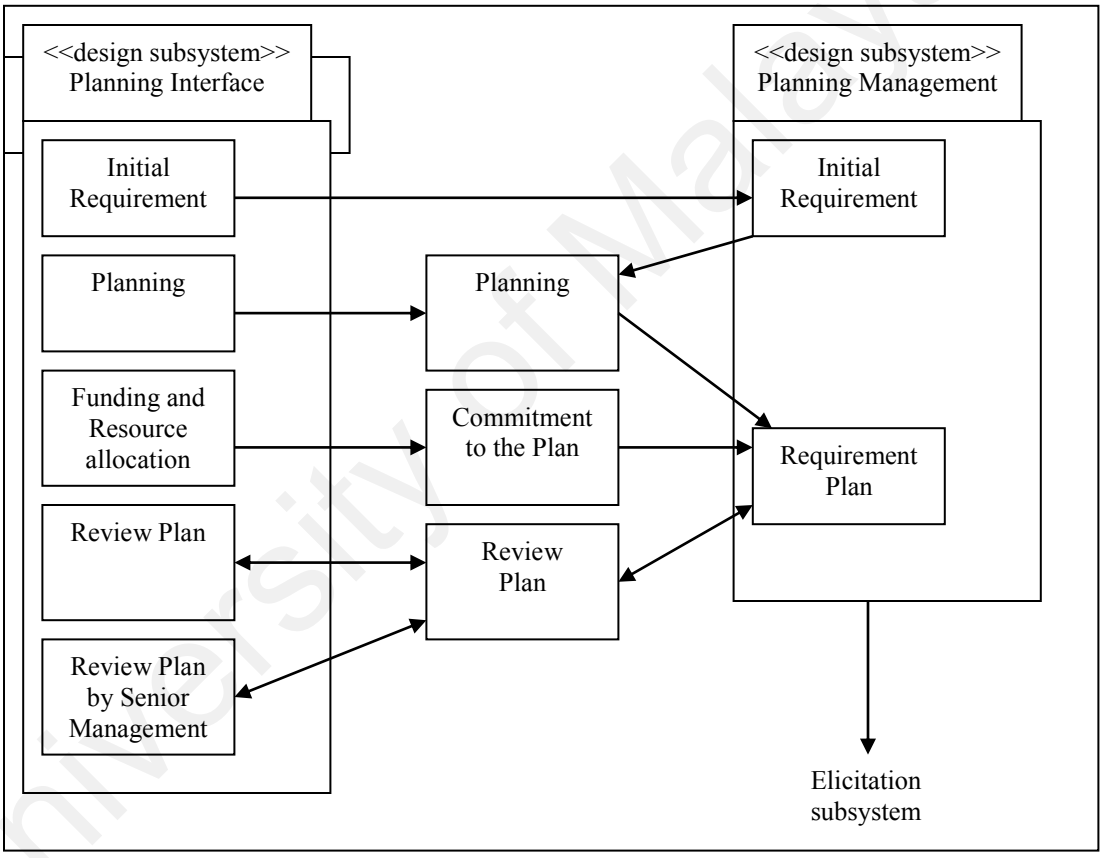


Figure 33 : Design Model - Requirements Engineering, Planning Subsystem.

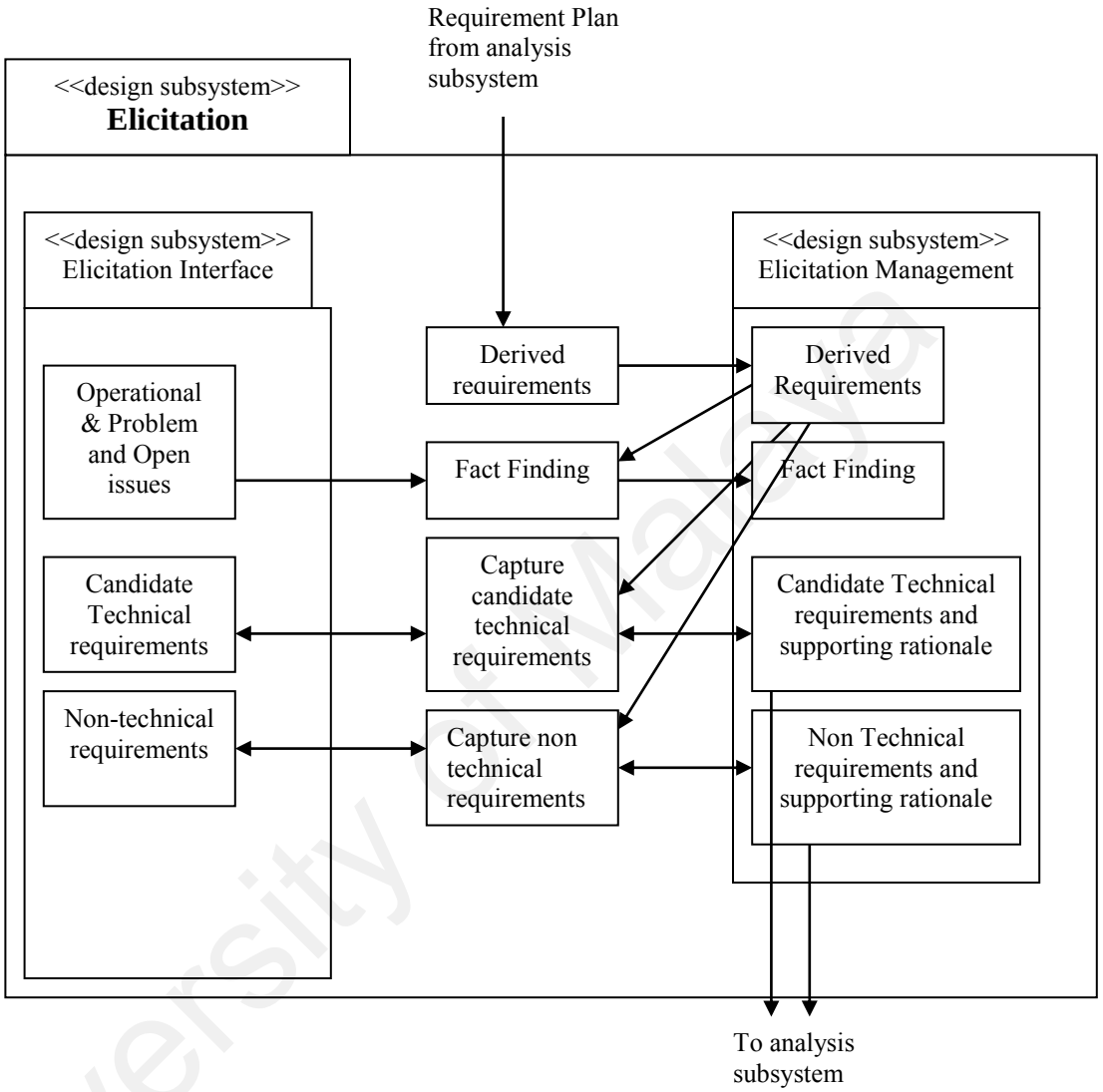


Figure 34 : Design Model - Requirements Engineering, Elicitation Subsystem.

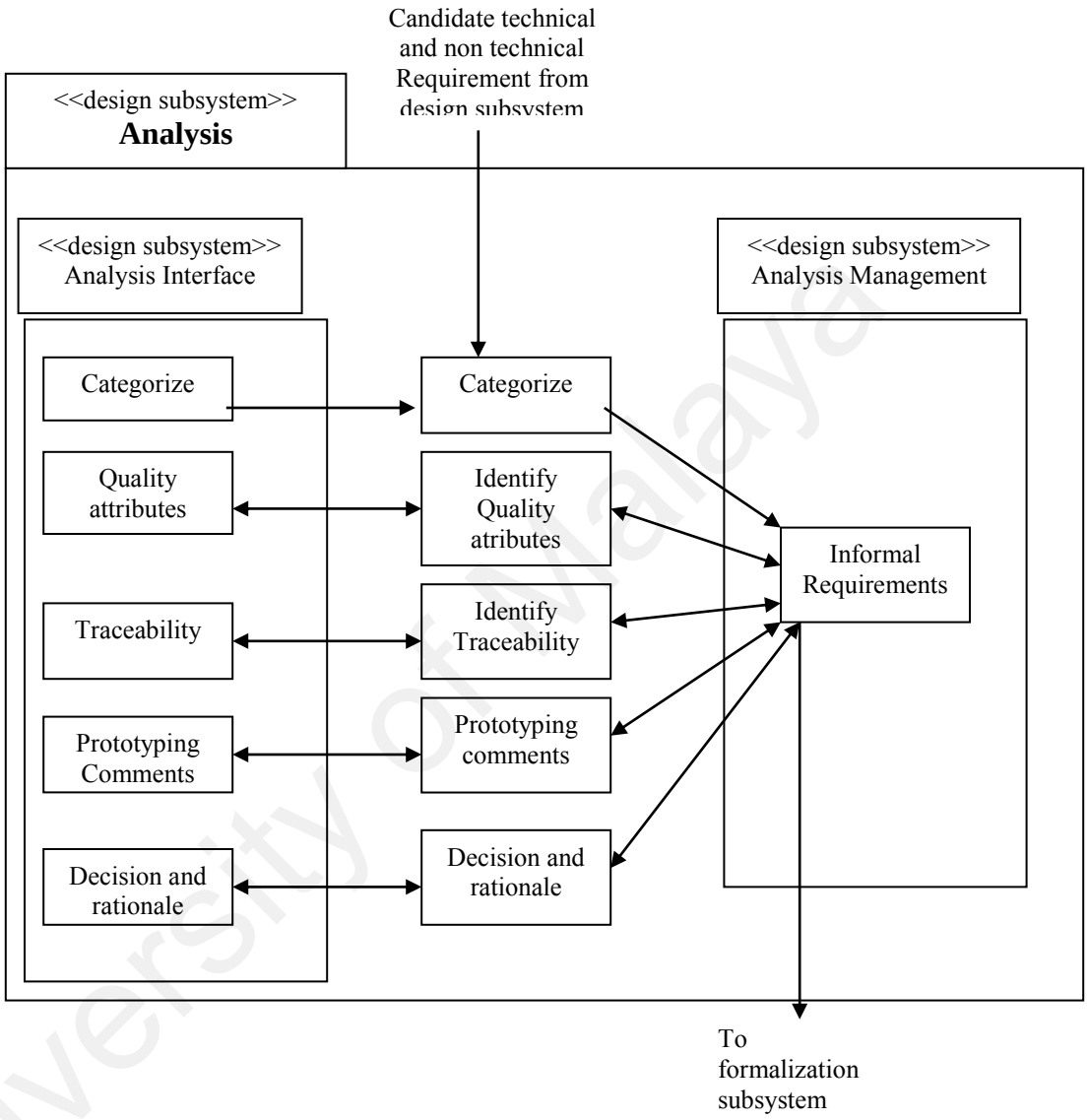


Figure 35 : Design Model - Requirements Engineering, Analysis Subsystem.

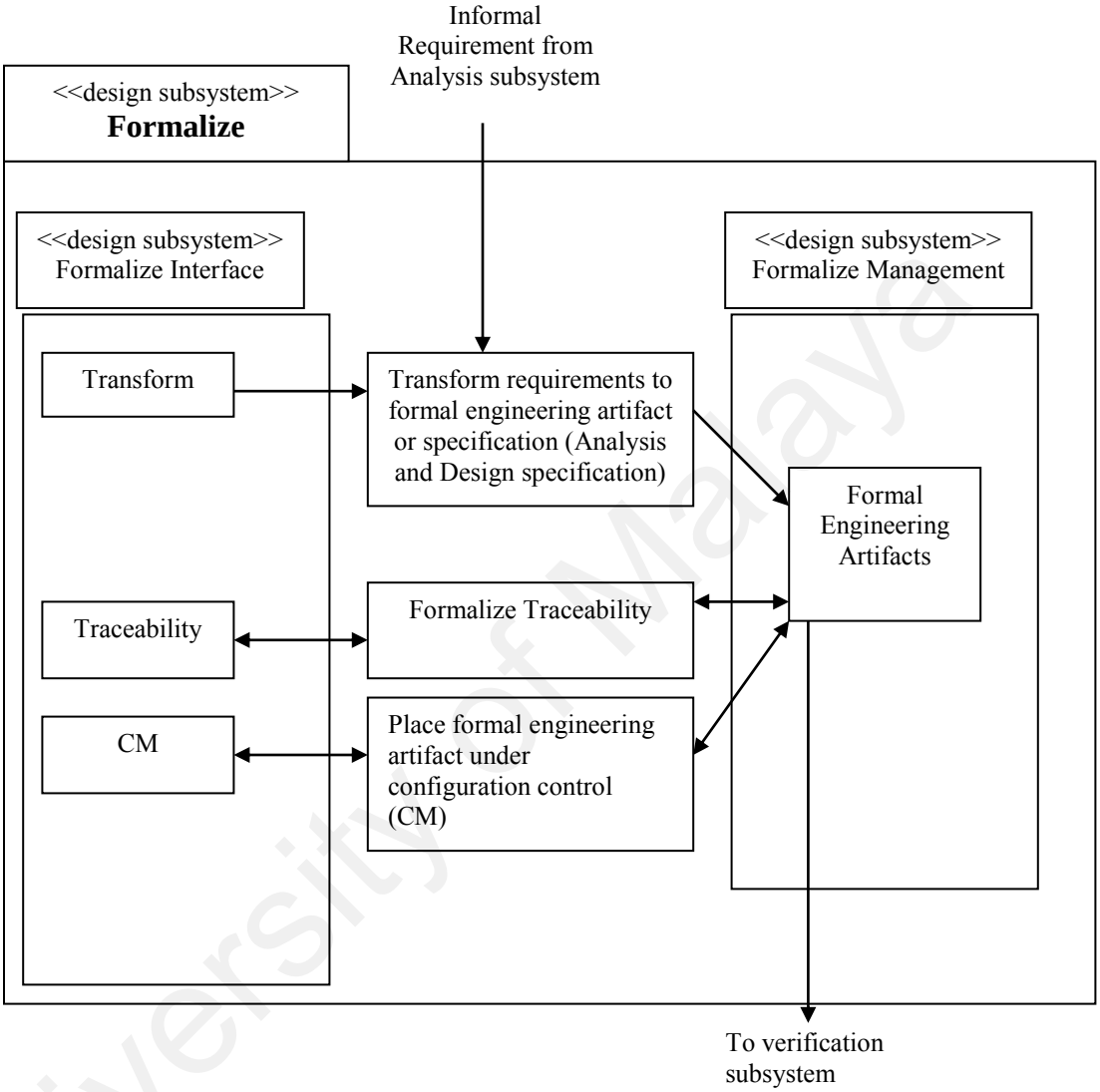


Figure 36 : Design Model - Requirements Engineering, Formalization Subsystem.

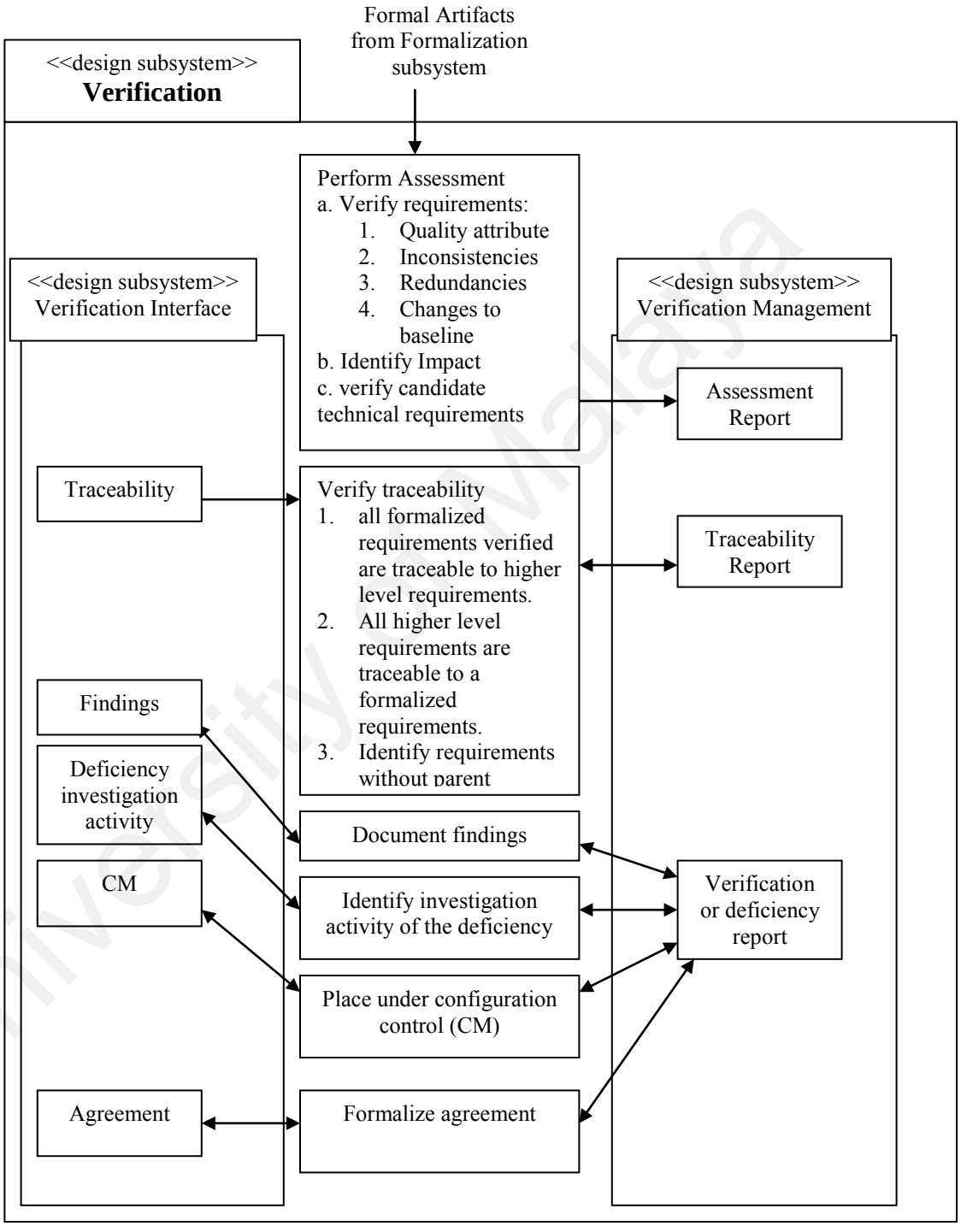


Figure 37 : Design Model - Requirements Engineering, Verification Subsystem

5.4.4.1 Design Model – Entity Relationship Diagram for Database modeling.

The development of the proposed system is base on relational database, hence the data modeling is to use entity relationship model (Hernandez M. J. ,1996).

i. List of identified entities.

- Initial Requirements
- Requirements Plan
- Derived Requirements
- Fact Finding
- Candidate Technical Requirements
- Non-technical requirements
- Informal Requirements
- Formal Engineering Artifact (formal requirement specification)
- Assessment Report
- Traceability Report
- Verification or deficiency report

ii. Identification of Entities attributes.

- Initial Requirements
  - Project ID
  - Project Description
  - Mission Need or Changed Mission need
  - Requirements or Changed Requirements
  - Operational Scenario
  - Change of deficiency report.
  - Baseline requirements
  - Requirements Plan
    - Project ID
    - Stake Holder
      - Name
      - Position
      - Organization
      - Acting
      - Email
      - User ID
      - Password
      - Group
    - Acceptance Criteria (System level requirements)
      - Requirement ID
      - Description
    - Non-technical requirements
      - Requirement ID
      - Description
    - Project Plan
      - Activities
      - Expected Start Date
      - Expected Completion Date
      - Cost
      - Human Resource
    - Asses Project Risk
    - Assign/Review Requirements management
    - Review by Stakeholders
    - Review by management
    - Funds and Human Resource allocation
    - Approval
    - Policy and responsibilities
    - Operational and problem context
    - Open issues and solution

- Requirements
  - Project ID
  - User ID
  - Requirement type
    - Technical Requirements
      - Technical Requirement ID (eg. 1.1.1)
      - Date Change
      - Requirements Description
      - REVISION
      - QUEROR
      - DESTINATION
      - AMBIGUITY
      - CRITICALITY
      - RISK
      - COMPLEXITY
      - COST
      - IMMEDIACY
      - VERIFICATION
    - Non-technical requirements
      - Non Technical Requirement ID (eg. 1.1.1)
      - Date Change
      - Requirements Description
  - Requirement Status and document artifact
    - Initial
    - Derived
    - Informal Requirements
    - Formal Engineering Artifact (formal requirement specification)
    - Assessment Report
    - Traceability Report
    - Verification or deficiency report

iii. Entities after normalization.

PK – Primary Key FK – Foreign Key

1. Initial Requirements

- Project ID (PK)
- Project Description
- Mission Need or Changed Mission need
- Requirements or Changed Requirements
- Operational Scenario
- Change of deficiency report.
- Baseline requirements

2. Requirements Plan

- Project ID (PK)
- Asses Project Risk
- Assign/Review Requirements management
- Review by Stakeholders
- Review by management
- Funds and Human Resource allocation
- Approval
- Policy and responsibilities
- Operational and problem context
- Open issues and solution

3. Project Plan

- Project ID (PK,FK)
- Activity No (PK)
- Activities
- Expected Start Date
- Expected Completion Date
- Cost
- Human Resource

4. Stake Holder

- Project ID (PK,FK)
- Name
- Position
- Organization
- Roles – SM, CUSTOMER, PM, SYSENG, SOFTENG, CM/QA, FT/OT, EU
- Email
- Telephone
- Fax
- User ID (PK)
- Password
- Group

5.Technical Requirements

- Project ID (PK,FK)
- Technical Requirement ID (eg. 1.1.1) (PK)
- Date Change
- Requirements Description
- Assessment Report
- Traceability Report
- Verification or deficiency report
- REVISION (PK)
- QUEROR
- DESTINATION
- AMBIGUITY
- CRITICALITY
- RISK
- COMPLEXITY
- COST
- IMMEDIACY
- VERIFICATION
- Prototyping Comments

6.Non-technical requirements

- Project ID (PK,FK)
- Non Technical Requirement ID (eg. 1.1.1) (PK)
- Date Change
- Requirements Description
- REVISION (PK)
- QUEROR
- DESTINATION
- AMBIGUITY
- CRITICALITY
- RISK
- COMPLEXITY
- COST
- IMMEDIACY
- VERIFICATION
- Prototyping Comments

7.Requirement Status

- Project ID (PK,FK)
- Requirement ID (PK,FK)
- Report Location
- Date Change
- Status (PK)
  - System Level
  - Derived
  - Informal Requirements
  - Formal Engineering Artifact (formal requirement specification)

.

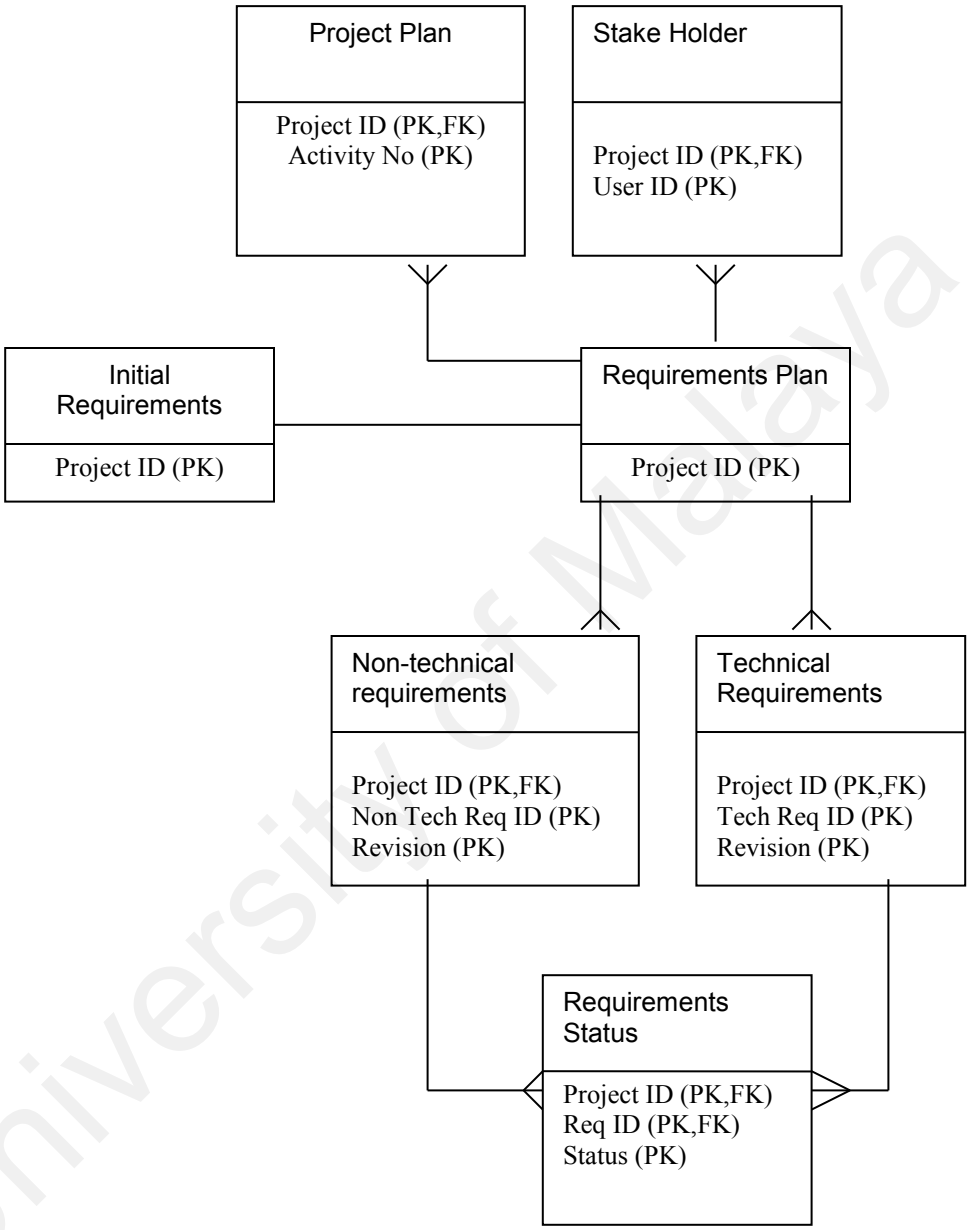


Figure 38 : Entity Relationship(ER) Diagram

5.5 The Proposed Project Development Tools.

The proposed project will be divided into components which are

i. Web Based Client and Server Component.

The system will be developed based on this architecture. The subsystem is to be used by all stakeholders. This component will be developed using Active Server Pages/ActiveX Components and run under MS-Internet Explorer environment as a client and Microsoft Web Server (PWS/IIS) as a server. For simplicity purpose, the DBMS to be used is MS-Access 97

ii. Windows Client component Subsystem.

- Some of the module probably has to be developed as a windows client in Local Area Network environment. This is due to lack of session management in a web based system. It will be developed using MS-Visual Basic. For simplicity purpose, the DBMS to be used is MS-Access 97
- Artifacts Management Subsystem. The artifact management subsystem will developed using MS-Visual Basic. For simplicity purpose, the DBMS to be used is MS-Access 97.

5.6 The Proposed component architecture diagram.

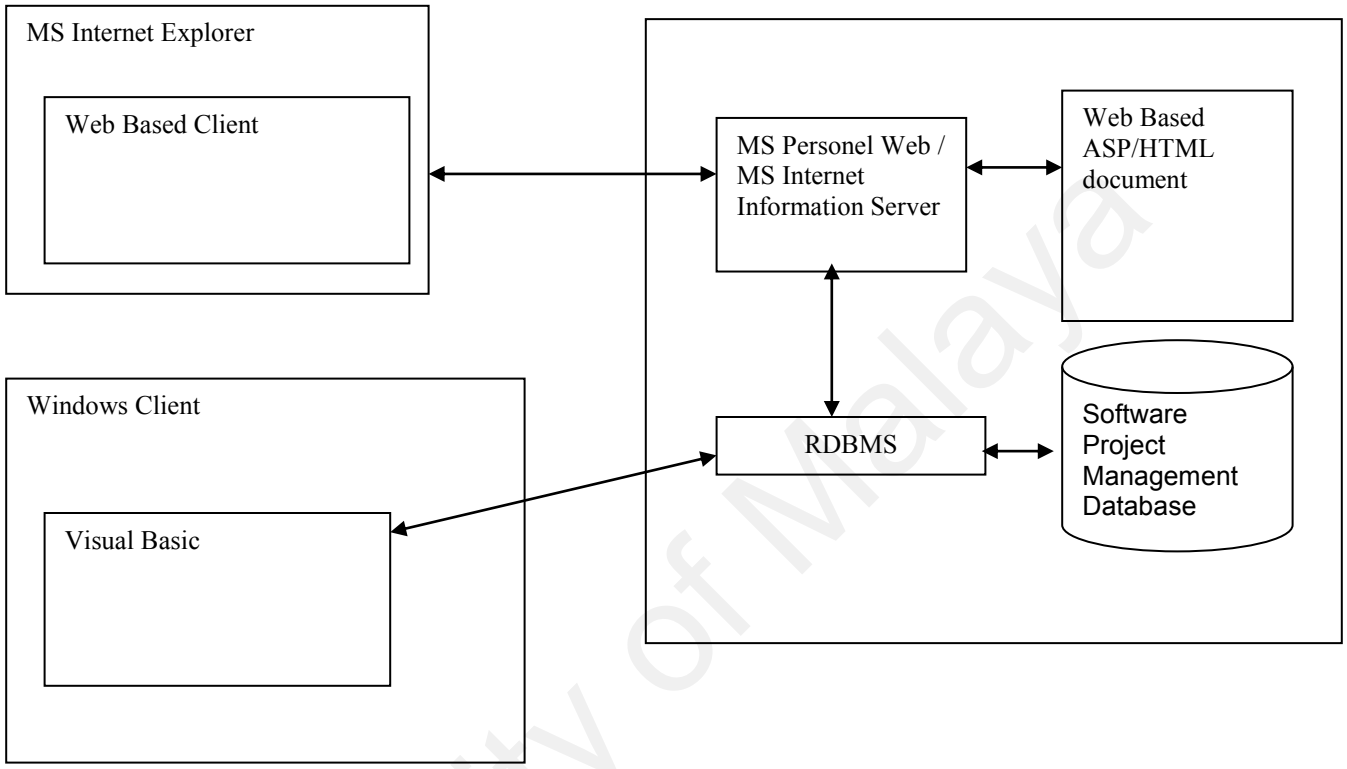


Figure 39.

Component Architecture Diagram

5.7 Project Methodology and development plan.

We will develop the prototype based on 'spiral model', but since this is only the prototype development and not working model, the cycle may not be complete when we completed this thesis.

Table 11. The development Plan and Deliveries are as below.

Activities	Duration	Start/Completion Date
1. Defining and Endorsement of the Protoype Scope	12 Weeks	1/10/2000-28/2//2001
2. Detail down prototype Scope : a. Understanding and Document prototype development objectives. b. Understanding and Document Functional requirements c. Understanding and document Model and Framework requirements	2 weeks 2 weeks	1/10/2000-15/3/2001
3. Project Development & Coding	8 weeks	1/9/2001-30/11/2001
4. Documentation & Dissertation.	36 weeks	15/4/2001-15/1/2002
5. Testing	4 weeks	1/12/2001-31/12/2001
6. Review Results from testing.	4 weeks	15/1/2002

5.8 Project Prototype Testing and Comments from tester.

The first version of project prototyped was released for comments on 30th November 2001. The system analyst involved in the case study (refer to chapter 3), was chosen to comment on the prototype. The prototype was installed into both of their PC's and both of them were given 30 minutes briefing and hands-on training on how to use the prototype. Both of them were given one week to use the prototype by entering information for the project previously managed by them which has been used for the case study (refer to chapter 3). They were also required to evaluate the functionality enhancements as compared to implementing the Requirements Engineering without tools, which had been done during the case study. The marks given shall be as below :

0 – if Requirements Engineering functionality shows no improvement.

5 – if Requirements Engineering functionality shows slight improvement.

10 – if Requirements Engineering functionality shows drastic improvement.

Both of them were also required to give comments to improve the prototype.

Table 12. Tools Prototype Functionality Evaluation.

Requirements Engineering functions	Comments by first tester	Comments by second tester	Mean
1. Capturing Requirements/Identification (Functional / Technical Requirements)	5 – OK	8 – I love the functional requirements structure. The only thing is the attribute shall not be hard coded.	6.5
2. Capturing System Element Structure (Non-Functional / Non-Technical Requirements)	3 - OK	8– I love the Non functional requirements structure. The only thing is the attribute shall not be hard coded.	5.5
3. Requirement Flow Down (Sub-system requirements)	3 - OK	2 – The flow down structure should be	2.5

		simplified	
4. Traceability Analysis (Version Control)	5 - OK	5 - OK	5
5. Configuration Management	5 - OK	2 – Should support more comprehensive configuration management.	3.5
6. Document & Other Output Media	5 – OK	5 – OK, using windows facility	5
7. Groupware	0 – No groupware ?	0 – So far, the groupware features doesnot seem to works.	0
8. Interface to other tools	0 – No Interface ?	0 – It does not support, shall have the capability of interfacing or passing data to other tools.	0
9. System Environment	5 – OK	5 – OK,	5

		windows base	
10. User Interfaces	3 – Shall improve ?	2 – very bad, it should improve	2.5
11. Support & Maintenance	NA	NA	NA
12. Overall	**This tool is very useful and I believe it is gone to help me a lot. The only thing is, it still lack of usability. User Interface need to be improved, anyway I like the tool especially its capable to manage the artifacts.**	**OK but need a lot of improvements before it is justified to be commercially used.**	
13. Average	**3.4**	**3.9**	**3.65**

5.9 Discussion on evaluation of prototype.

The functional (technical) and non-functional (non-technical) requirements capturing feature is easy to develop. It is just a screen to update the database fields. Therefore it is able to score high mean, 6.5.

The requirement flow down feature need to involve some complex process, we do not automate the function. The improvement for the requirement flow down feature can be developed base on some artificial intelligence mechanism. Probably when the requirement was initially captured in the planning phase, the "automated derivation process" during elicitation phase is capable to derive the requirements intelligently. The mean score for the feature is 2.5.

The traceability feature are part of INCOSE(1998) model requirement attribute, which we apply it here during analysis phase. The improvement required for the feature probably on traceability report. The mean score for the feature is 5.

The configuration management are normally developed as a separate tool. The prototype, only have very basic feature, to show its functionality. This was done by managing different folders in Microsoft Windows Explorer. The improvement to the feature, apart from improving the current feature, is to develop interface to the readily available configuration management tools. The mean score for the feature is 3.5.

The document and other output media feature can be further improved by developing a template to the requirements engineering artifacts. The template can be in a Microsoft Word or PDF format and it must capable to merge all the information captured in the database. The mean score for the feature is 5.

The groupware feature was not incorporated in the current system prototype. The feature can be developed if the whole system is developed base on web architecture. It is very nice if we have time to develop, the system base on web with some email and news processing capability.

The interfacing to other tools is also a very nice to have feature. The current system prototype does not have this feature. The CASE tools for analysis and design, the modeling tool such as UML are probably some of the potential candidate. INCOSE(1997) describe the interface mechanism for tools supported by the model.

The system environments feature for the current system prototype is running on Microsoft Windows environment. The improve version of the system may have the system running on web, hence it can offer multi-platform capability. The mean score for the feature is 5.

The user interface feature for the prototype was not properly designed. It is recommended for the improvement of the system user interface, some detail analysis for user interface is conducted. The mean score for the feature is 2.5.

The support and maintenance feature for the prototype are not available since the system was not developed for commercial.

5.10 Conclusion.

From the evaluation of the prototype by the two testers it can be summarized that, on overall the tool is capable of improving the requirements engineering processes, however in order to ensure usability of the tool further research and continuos improvement are needed.

Chapter 6

Overall finding of the research.

6.0 Introduction

When we started the research three years ago, not many reference materials related to the subject could be found. Today we found that the subject had become more popular and the related topics pertaining to the subject from many sources are available even online.

We found, the research is very useful, especially it is about a topic which is not used to be discussed by many analysts or software engineers but it is so significance in the sense that it contributes to the high failure risk if it is not being addressed properly. Many analysts from the industry that we approach realize the problems of requirements management but most of them accept them without really understanding the requirements engineering attributes and processes in detail.

6.1 Limitations of the research and future improvement

6.1.1 Requirements Engineering versus CMM level.

Our research is focusing on applicable requirements engineering tools for small or immature organization. It is a matter of fact until now there was no organizations working on classifying CMM level to the organizations in Malaysia.

According to Judith G. B.(1994) , the research to determine the applicability of the Software Engineering Institute's (SEI) Capability

Maturity Model (CMM) for Software to small businesses and small software organizations found that small businesses are faced not only with a lack of resources and funds required to implement many of the practices stated in the CMM, but also with the task of basing their process improvement initiatives on practices that do not apply to a small business and small software organization. He discusses why small businesses and organizations are experiencing difficulties implementing CMM-based process improvement programs and how some are tailoring their approach to meet their quality goals.

In our research we differentiate between mature and immature organization base on years of establishment, which may not represent real scenario of CMM level and the size of the organizations.

6.1.2 Usage of the Requirements Engineering tools.

Usage of the tools that was developed is based on a few users which may not show the reality. We do not develop it for a large scale software project. We develop it to be a simple tool, which is to guide the analysts or engineers to deliver proper requirements documentation. There was no intelligent analysis engine. According to Hummer T et al (1998), the nature of the Requirements Engineering artifacts is detailed, time-sensitive, highly internally dependent, and can be continuously changing. Hummer T

et al(1998) suggest that, a common failing has been the manner in which tools are used to support requirement management processes starting at the initial development phase. We agree with Hummer but our definition to his “manner in which tools are used” shall be according to the size and complexity of the project. More research can be done in this area to map the right tools functionality to the right project.

6.1.3 Managing changes and Agile processes.

Our tool is capable of capturing the requirements for the simple system in which the requirement hierarchy is two levels and the requirements has been identified at the initial stage. Apart from versioning control the tools required manual update downward or upward the requirements hierarchy to apply any requirements change. It is incapable of managing requirements change during development. But the reality about the software development is the requirements may change during development. This is being identified as agile processes. We believe this is a very brilliant idea since from our past experience it is very difficult to finalize the requirements start from the initial stage. We support the idea of agile processes but our tools for the time being is incapable of managing agile processes. Agile process is the iterative development which is in contrast with up front requirements elicitation and analysis. According to Favare J (2002), requirements may be introduced, modified, or removed in

successive iterations. Again we believe more research can be done on requirements engineering that support agile processes especially for the small scale project.

6.1.4 Cost benefit on requirements engineering.

For the small scale organizations which manage small software project, this is a dilemma which many of them tend not to apply requirements engineering as phases in software development processes. This is due to lack of resources and the fact that the management does not see the cost-benefits of the requirements management. The organizations tend to work on limited resources – time and human.

According to Favare J(2002), assessing the cost–benefit of requirements is more difficult than design or implementation, because requirements are the furthest upstream in the development process. Consequently there are more unknown factors; it can take a full development cycle before the complete economic impact of a requirement is known. This is why some organizations just skip the requirements engineering phase.

Favere J(2002) also elaborate a few techniques to increase the cost-benefit which is include the techniques for creating reusable requirements and also tools of cost–benefit analysis that apply to the agile requirements

process. Again we believe more research can be done in the area of requirements engineering cost-benefit analysis.

6.1.5 Requirements engineering training.

The requirements engineering may follow known established method and model, and due to its complexity and uncertain cost-benefit some software organizations just do not have any interest to invest. Even though they realized the risk and the importance of requirements engineering but the investment to train their team to apply the complex process of requirements engineering some time is not worthwhile.

We are convinced by a journal article on “Transition packages: an experiment in expediting the introduction of requirements management” by Fowler P et al (1998). The transition package was a password-protected web site that included about 100 documents for use in both performing and introducing requirements management practices. These documents included examples, templates, checklists, and guidance materials. The outcome of the research is convincing and capable to improve the “transition period”.

6.1.6 Collaboration and web base tools.

In our tools, we had used web base form to capture initial requirements from various stake-holders. This is to promote collaboration and the analysts or requirements engineer will elicit and analyze the requirements and transform it into the structured form – which is categorizing and building requirements hierarchy. The requirements engineer will table the initial requirements specification in the meeting for verification. Verification process can be in the form of official discussion or prototyping.

Duran A et al(2002) is developing a web base requirements management verification techniques using XSLT language. This is interesting but we are not sure on their practicality and usability. Anyway, we believe using web to improve requirements engineering tools collaboration functionality is one of the area to be explored.

6.1.7 The politics of requirements management.

Andriole S et al (1998) suggest that most of the time when we think about requirements management, we always think about its technicality. We tend to forget other very important factor to be considered – which is politics. According to him apart from the requirements and stakes holder identification, the project “discretionary” is also to be identified. The impact analysis especially on cost benefit once the project completed is to be done. On the functional requirements, the task specification has to be

identified, prioritized and analyzed risks. The “nonfunctional” requirements, like security, usability, and interoperability are to be identified, prioritized and to analyzed their risks. The trade off between functional and nonfunctional requirements must also be identified.

He also suggested that before we can starts prototyping we must understand the project well enough. And if the prototype is acceptable, everyone must willing to sign off on the prioritized functionality and non-functionality to be delivered, on the initial cost and schedule estimates, on the estimates’ inherent uncertainty, on the project’s scope, and on the management of additional requirements.

We think this is another area of research on requirements engineering tools. Probably tools that are capable of giving early warning alert when the political environment is unhealthy.

6.1.8 The interfacing to other tools.

The research can be extended to allow interfacing to other tools. The current tool, allow interfacing by executing application from within the tool via the artifacts management function. The artifacts developed from any tools can be stored in our RM tool and it is structured according to the requirements hierarchy and version. The current version of our requirements management tool does not treat the application as object component hence it does not pass any parameter. The future research can be done to allow parameter passing to the tools. INCOSE (1998) define some of the interfacing criteria which may help any tools developer to incorporate the interfacing.

6.2 Our contribution.

We are in fact trying to contribute a simple requirements management model to be used for small organization. When we develop our model we had come across one good article by Woodruff W. (1997) on Requirements Management for small organization. Woodruff W. (1997) describes a practical approach to requirements management for small organizations, shall include the following issues:

- Establishing a requirements and document hierarchy
- "Decomposing" requirements documents
- Maintaining requirements and traceability

Woodruff W. (1997) is proposing Requisite Pro, a requirements management tool by Rational Software Corporation's to manage requirements management.

Requisite Pro is a tool that was develop base on Rational Unified Process (RUP) which is a complex requirements engineering methodology which is developed for "life-cycle" development methodology. Even though it is capable of handling small organizations but it was not designed for it. Woodruff W. (1997) proposed the use of matrix to manage requirements change and traceability and this has to be done manually since the actual change management in Requisite Pro is complex.

We had developed a model based on the **common task in requirements engineering** identified during our studies (refer chapter 2)**.** Our model is complete but simple and it is designed for small organizations, when we exclude all the complications especially in managing change. We provide change management via version control and there is no such thing as automatic propagation upward or downward requirements hierarchy. Our model, include artifact management functionality and it must support version control. To simplify change management we attach version to the highest hierarchy level. When there is change, the whole requirements hierarchy will be duplicated and given the new version. Any changes has to go through the whole process cycle of Planning, Elicitation, Analysis, Formalization and Verification. Our model is applicable for both "waterfall" and "life-cycle" development methodology.

We had developed a requirements management (RM) tool prototype base on the model. An approach to use the tool has slight different compare to Woodruff W. (1997), when we suggest the issues to be address shall be :

- Establishing a requirements hierarchy
- Identifies requirements artifact (documents) for each requirements hierarchy.
- Maintaining requirements and traceability

Our RM Tools capable of registering requirements and its hierarchy. RM tools for the time being manage the requirements hierarchy through the level of reference number. For example 1, 1.1 and 1.1.1 is for three downward levels of requirements. RM tools support INCOSE (1998) requirements attribute. RM tools capable of managing the artifact documented using any type of tools available including CASE tools or MS-Office. We had made full use of Microsoft's windows technology library for file management.

In order to commercialize the tool, there are lot of improvements need to be done especially in the user interface.

6.3 Conclusion.

The research in requirements engineering tools is very important to ensure the greater success of software development project. We know the requirements engineering processes is a complex process and that is probably the reason why it is not widely used until today. The simple tools which can encourage application of requirements engineering is very important at least to the extend that is capable of encouraging the organizations to appreciate the investment in requirements engineering. We are looking forward to further improve our research. A research with the Malaysian government to identify the success rate on the government software development project; plan, train and implement the application of simplified requirements engineering processes and tool; and to prove the usage of tools will improve the project success rate. There are lists of improvements listed in 6.1 to be considered in order to ensure the tools to be really useful for the beginners.

References

Andriole, S. (1998). *The politics of requirements management* .IEEE Software , Volume: 15 Issue: 6 , Page(s): 82 –84

Ashworth C. and Slater L. (1993). *An Introduction to SSADM Version 4*. McGraw-Hill Companies

Andersen Consulting (1990). *Andersen Consulting – FOUNDATION : METHOD/1* . Andersen Consulting.

Christel M. G. & Kang K. C. (1992). *Issues on Requirements elicitation*. Carnegie Mellon University/ Software Engineering Institute (CMU/SEI).

Cockburn A (2002). *Agile Software Development*, Addison Wesley

Duran, A.. Ruiz-Cortes, A.. Corchuelo, R.. Toro, M.. (2002). Supporting requirements verification using XSLT, Requirements Engineering, 2002. Proceedings*. IEEE Joint International Conference on* , Page(s): 165 -172

Favare, J. (Mar/Apr 2002). *Managing requirements for business value*, IEEE Software , Volume: 19 Issue: 2 , Page(s): 15 -17

Fenton N. E. & Pfleeger S.L. (1997), *Software Metrics, A Rigorous Approach*, PWS Publishing Company.

Fowler, P.. Patrick, M.. Carleton, A.. Merrin, B. (1998). Transition packages: an experiment in expediting the introduction of requirements management, Requirements Engineering, 1998. Proceedings*. 1998 Third International Conference on* , 6-10 Page(s): 138 -145

Goodwin S. and Nellon J. (1999*). System Level Automation Tools for Engineer (SLATE)*, Texas Instruments.

Hammer, T.. Huffman, L. (1998). Automated requirements management-beware HOW you use tools: an experience report Requirements Engineering, 1998*. Proceedings. 1998 Third International Conference on* , 6-10 1998 Page(s): 34 -40

Henderson-Sellers B, Due R and Graham I (1999). Existing OO process-focused methods : A critique of RUP and OPEN from a PM perspective; School of Computing Science, University Of Technology, Sydney.

Hernandez M. J. (1996*). Database Design for Mere Mortals: A Hands-on Guide to Relational Database Design*. Addison Wesley.

IEEE (1999),STD 1220-1294 *definition of the Requirement*. IEEE

INCOSE (1996). *Characteristics of Good Requirement, INCOSE*. Requirements Working Group

INCOSE (1997). *Interfacing Requirements Management Tools In The Requirements Management Process - A First Look (A Requirements Working Group Information Report).,* INCOSE Requirement Working Group.

INCOSE (May 1998). *Executable Requirements Management Model, Interim Report* . INCOSE Requirements Working Group.

Jacobson I, Griss M, Jonsson P. (1997*). Software Reuse – Architecture, Process and Organization for Business Success*. Addison Wesley.

Jones C. (1978). *Project Management Tools and Software Failures and Successes* . Software Productivity Research, Inc.

Judith G. Brodman , Donna L. Johnson (1994). What small business and small organizations say about the CMM: experience*, report Proceedings of the 16th international conference on Software engineering*

Kan S.H,(1999) *Metrics and Models in Software Quality Engineering*, Addison Wesley.

Lanman J. T. (2002). A Software Requirements Engineering Methodology Comparative Analysis : Structured Analysis, Object-Oriented Analysis and Problem Frames ; Department of Computing and Methamatics, Embry-Riddle Aeronautical University.

Leffingwell D. and Widrig D.,(1999*), Managing Software Requirements: A Unified Approach*, Addison Wesley.

Meli R. (1999). *Risk, Requirements and Estimation of a software project*, European Software Institute (ESI)

Peter Hantos (May 1999). A systems engineering view of requirements management for software-intensive systems. *Proceedings of the 21st international conference on Software engineering*

Rational Software Corporation et al (1997). *Rational Unified Process (2001)* . Rational Software Corporation.

Rational Software Corporation et al (2001*). Unified Modeling Language(UML) Summary ver 1.1.* Rational Software Corporation.

Rodrigues A. G. and Williams T. M. (1996). System Dynamics in Software Project Management: towards the development of a formal integrated framework . *MIT Boston : International System Dynamics Conference.*

Royce Walker.(1999), *Software Project Management – A unified Framework*, Addison Wesley.

Sellapan, P.(2000), *Software Engineering – Management and Methods*, Sejana .

SEI (1993) *Key Practices of the Capability Maturity Model, Version 1.1*. Software Engineering Institute.

Stephen H. K. (1995), *Metrics and Models in Software Quality Engineering*. Addison Wesley

US Naval (1999). *The Requirements Management Guidebook. US Naval Air Systems command through the Computer Resources Group's (CRG),* Requirements Management Working Group –RMWG

Wilkie G. (1997), *Object Oriented Software Engineering* , Addison Wesley.

Woodruff W. (1997), *Requirements Management for Small Organization*. Rational Software Corporation.

Yeh R.T., Zave P., Conn A. P., Cole G. E.(1984). *Handbook of Software Engineering, Software Requirements : New Directions & Perspectives*. p 519-543. Van Nostrand Reinhold Company Inc .