*Master Thesis*
*Software Engineering*
*Thesis no: MSE-2007:27*
*October 2007*


# Suitability of the Requirements Abstraction Model (RAM) Requirements for High Level System Testing

**Naeem Muhammad**

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

**Contact Information:**
Author:
Naeem Muhammad
Address: Folkparksvagen 17:09
37420, Ronneby, Sweden
E-mail: naeemmuhammad@gmail.com

University advisor:
Dr. Robert Feldt
Department of Systems and Software Engineering
Blekinge Institute of Technology School of Engineering
372 25 Ronneby, Sweden

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 271 25

# ACKNOWLEDGEMENT

Completing this task was not an easy job, but it turned out to be an easy job with the support of various entities. I would like to mention some of them here.

Thanks to the lord of the lords Allah (SWT) and Prophet Muhammad (SAW) for uncountable blessings to me at each moment of my life.

I am thankful to my supervisor Dr. Robert Feldt, whose continuous guidance and encouragement made it possible for me to complete this task effectively.

I am grateful to my beloved father, what I am today is because of him. He sacrificed his life for me. I am thankful to my family and the persons very closed to me, they supported me all the way in my life, particularly during this study.

Thanks to my friends Qasim Ali and Kamran Fazal for their support during all the times. They are continuous source of encouragement for me. Thanks to the other friends in Ronneby for their support.

Sweden, people of the Sweden and the staff at the BTH are very cooperative, I would like to express thanks to them for their support during my study.

Naeem Muhammad
Ronneby, Sweden

# ABSTRACT

In market-driven requirements engineering requirements are elicited from various internal and external sources. These sources may include engineers, marketing teams, customers etc. This results in a collection of requirements at multiple levels of abstractions. The Requirements Abstraction Model (RAM) is a Market Driven Requirements Engineering (MDRE) model that helps in managing requirements by organizing them at four levels (product, feature, function and component) of abstraction. The model is adaptable and can be tailored to meet the needs of the various organizations e.g. number of abstraction levels can be changed according to the needs of the organization.

Software requirements are an important source of information when developing high-level tests (acceptance and system level tests). In order to place a requirement on a suitable level, workup activities (producing abstraction or breaking down a requirement) can be performed on the requirement. Such activities on the requirements can affect the test cases designed from them. Organizations willing to adopt the RAM need to know the suitability of the RAM requirements for designing high-level tests.

This master thesis analyzes the requirements at product, feature, function and component level to evaluate their suitability for supporting the creation of high-level system test. This analysis includes designing test cases from requirements at different levels and evaluating how much of the information needed in the test cases is available in the RAM requirements. Test cases are graded on a 5 to 1 scale according to the level of detail they contain, 5 for better detailed and 1 for very incomplete. Twenty requirements have been selected for this document analysis; twenty requirements contain five requirements from each level (product, feature, function and component). Organizations can utilize the results of this study, while making decision to adopt the RAM model.

Decomposition of the tests developed from the requirements is another area that has been explored in this study. Test decomposition involves dividing tests into sub-tests. Some benefits of the test decomposition include better resource utilization, meet time-to-market and better test prioritization. This study explores how tests designed from the RAM requirements support test decomposition, and help in utilizing above listed benefits of the test decomposition.

**Keywords:** Market-Driven Requirements Engineering, Requirements Abstraction Model, Software Testing, Test Case Decomposition.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# 1 INTRODUCTION

This chapter describes the background and purpose of this study. It also states aims, objectives, research questions and the research methodology selected to answer the research questions.

## 1.1 Background

Testing in the traditional development methodologies is considered to be the last phase. In such methodologies, testers design or at least execute tests at the last phase. Recent approaches to software testing show that testing activities can be started in parallel to the requirements engineering activities. The V-Model [26, 27] is a model that maps testing activities with the software development phases of the waterfall model. This model suggests that acceptance tests can be designed as soon as requirements are available. Starting testing activities at an early stage provides many benefits. For example it can help in finding and fixing the bugs in the requirements before system implementation [35]. Fixing the bug at the end can result into major changes in the design of the system. So, it is less expensive to find and fix the bug at early stages rather fixing at the later stages [8]. Also, developing tests at the last stage may put much pressure on the testing team; this may affect their efficiency [1].

Requirements can be used for designing tests [5, 35, 38], and designing tests from the requirements at early stages is very cost effective. This shows that requirement is an important source for designing tests, so it is important to know the level of support a requirement can provide for developing tests. A testable requirement is said to be suitable for designing tests as it is correct, unambiguous, complete and consistent with the specification standards [4]. According to Sommerville et al. testable requirement is that which provide opportunity to define one or more test cases so that executing these test cases shows that the requirement has been implemented in the system [3].

Market-driven requirements engineering (MDRE) is concerned with the requirements engineering of the products targeted for mass market. Requirements for such products are elicited from various stakeholders e.g. users, developers and marketers [19]. Requirements gathered from various sources and domains are of different levels of abstraction. Managing such requirements with different abstraction is a difficult task [18]. In MDRE it is more difficult to mange such requirements with various abstractions because requirements come continuously to the requirements engineering process [36].

The Requirements Abstraction Model (RAM) [18] is a model for the MDRE that helps in managing different level of abstraction among the requirements, and handles continuous arrival of the requirements. This model has been developed by Gorschek and Wohlin [18, 19, 20, 21]. RAM provides four levels of abstraction on which requirements can be placed. These levels include product level, feature level, function level and component level [18]. Requirements at product level are highly abstract; product level requirements are similar to the product goals and strategies [18]. Feature level requirements describe the features of the system; these requirements do not provide details about the functions that are required to support these features [18]. Function level requirements contain information about the functions of the system [18]. Component level requirements contain detailed information and provide how things are to be solved [18]. Abstraction of the requirements increases from component level to product level.

The need for the RAM model arose from problems being faced at Danaher Motion Särö AB (DHR), however this model is flexible and can be tailored for other organizations [18] e.g. it allows organizations to create number of levels according to their needs. As it has been discussed earlier that designing tests from the requirements

at early stages is very cost effective, so it is important to recognize the suitability of the RAM requirements for designing tests. Since the RAM model has potential to be catered for other organization [18], a study on understanding the suitability of the RAM requirements for testing will be of great worth in generalizing this model.

This master thesis project examines the testability of RAM requirements. The purpose is to analyze the effect of different abstraction levels in requirements (requirements abstraction increases from component level to product level) on high-level system tests. System and acceptance level tests can be designed from the requirements. The level of detail these requirements contain affects the test cases. Requirements with more detail can result in better test cases. This study will help in understanding how this connection between abstraction level and testability functions. Since the RAM model is in its initial versions the results of this study can help in improving future versions of the RAM model.

In addition to exploring the suitability of RAM requirements for developing high-level system tests, we will also explore the opportunities that RAM requirements provide for test decomposition. Test decomposition means dividing tests into multiple tests. Dividing a test into multiple tests can provide many benefits. Some of the benefits include better test prioritization, better resource utilization and meeting time-to-market constraints. This study will explore state of the art research in test decomposition. Moreover, the study analyse the test cases designed from the RAM requirements for finding any opportunity for test decomposition. Discovering test decomposition opportunity in the tests designed from the RAM requirements will enhance the worth of the RAM, as it could be possible to get the above listed benefits through the RAM requirements.

## 1.2 Purpose Statement

The purpose of this study is to analyze the effect of the increase of requirements abstraction from component level to product level, on designing high-level system tests. In addition this study will explore the decomposition of the tests designed from the requirements, and will investigate how tests designed from the RAM requirements can be helpful for test decomposition.

## 1.3 Research Questions

1. To what degree are product, feature, function and component level RAM requirements suitable for developing high-level system tests?
2. What kind of analysis can be performed on the RAM requirements in order to find their suitability for high-level system tests (acceptance and system level tests)?
3. What is the current state of the art research in the test decomposition?
4. Do tests designed from the RAM requirements provide any opportunity for test decomposition?

## 1.4 Research Methodology

Qualitative research methodology will be used to answer the above set of research questions. Detailed and comprehensive state of the art literature will be studied in requirements abstraction, tests decomposition and requirements-based testing areas. A document analysis of the RAM requirements document will be performed in order to find suitability of the RAM requirements for high-level system tests. This document analysis includes selection of test case template, selection of requirements from the RAM document and development of test cases from the selected requirements. Designed test cases will be graded on a 5 to 1 scale, based on the level of detail they contains.

## 1.5 Related Work

Requirements Abstraction Model was developed to handle requirements with multiple abstractions. At start the RAM model was developed to meet the needs of the Danaher Motion Särö AB (DHR) but it can be tailored for other organizations [18]. Gorschek, Garre, Larsson and Wohlin have performed an industry evaluation of the RAM model [39]. The evaluation was made by using the RAM in two separate requirements engineering processes at DHR and ABB. Effects of implementing the model at both organizations were studied, effects were evaluated from two perspectives i.e. work performed and requirements quality [39]. Results of the evaluation proved that RAM model can be tailored for various organizations. In addition, results show that the RAM model can improve requirements engineering process.

## 1.6 Structure of the Thesis

Chapter 2 provides understanding about basic concepts for requirements engineering, software testing, RAM model, V-Model and requirements-based testing. Chapter 3 consists of a document analysis performed on the RAM requirements; it includes description, reason and application of the method selected for document analysis. Chapter 4 describes the test case decomposition and current state of research in it. Chapter 5 includes discussion on the results of the study. Chapter 6 summarises the study with some future works and conclusions.

# 2 REQUIREMENTS ENGINEERING AND SOFTWARE TESTING

Requirements engineering and testing are the phases of the system development life cycle which have the most interaction with the customer. Requirements engineering processes interact with customer for requirements elicitation. In testing the customer is involved to test their requirements in the system. This chapter provides an introduction about different activities involved in both requirements engineering and testing processes. Moreover, we introduce the Requirement Abstraction Model (RAM), the V-Model and Requirements-Based Testing (RBT).

## 2.1 Requirements Engineering

Every system has certain characteristics that are collected at the start of its development. These characteristics are known as requirements. IEEE has defined requirements in their IEEE 610.12-1990 standard [13] as:

1. "**A** condition or capability needed by a user to solve a problem or achieve an objective."

2. "A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents."

Moreover, Aurum and Wohlin [14] have defined requirements as:

"A requirement is a collection of needs arising from the user and various other stakeholders (general organization, community, government bodies and industry standards), all of which must be met."

Since, these are the desired capabilities of the system these are elicited before system design and implementation. Requirements engineering is the initial phase of the system development life cycle in which system requirements are elicited and specified. Collecting the right requirements is an important task for requirements engineers which can provide a system fulfilling the true needs of the customer.

The requirements that are directly elicited from stakeholders are known as primary requirements. Requirements which are derived from primary requirements are called derived requirements [14]. However, requirement belonging to any of the above type can be put in to functional or non functional category.

- Functional Requirements
  These requirements are the required function or actions of the system. Generally, these requirements are specified in combination of natural language and modelling languages.
- Non-Functional Requirements
  These requirements are restrictions or constraints on the system e.g. performance, safety, availability. Non-functional requirements (NFR) are of great importance, especially for mission critical system. Generally NFRs are categorized as process, product and external requirements [3]. Table 1 contains different types of NFRs [3]

| | |
|---|---|
| 1 | Performance requirements |
| 2 | Interface requirements |
| 3 | Operational requirements |
| 4 | Resource requirements |
| 5 | Verification requirements |
| 6 | Acceptance requirements |
| 7 | Documentation requirements |

| | |
|---|---|
| 8 | Security requirements |
| 9 | Portability requirements |
| 10 | Quality requirements |
| 11 | Reliability requirements |
| 12 | Maintainability requirements |
| 13 | Safety requirements |

Table 2-1 Non-functional requirements in IEEE Std-830-1993 [3]

### 2.1.1 Requirements Engineering Process

The requirements engineering process is the set of activities used to identify system requirements so that these requirements can be provided as input to the next phases of the development life cycle. There is no standard requirements engineering process, different organizations have customized processes to suit their needs [3]. However, there are some standard activities that are part of the requirements engineering process as shown in Figure 2-1[3]. These activities include requirements elicitation, analysis and negotiation, documenting requirements and requirements validation. Requirements management is another activity, which runs in parallel to the overall requirements engineering process. Requirements engineering processes with these activities are used both for bespoke and market-drive projects [18]. In bespoke projects the requirements engineering process executes for a single project whereas in the market driven projects the requirements engineering process executes for each version of a single product.

These activities require certain resources; organizations can cater these activities according to their resource limitations. For larger systems these activities cost about 15% of the total budget and for smaller systems this ratio to total budget is 10% [3]. However, these stats can increase later if requirements are not handled well. In the following sections we will describe activities of the requirements engineering process; we will also explain the Requirements Abstraction Model (RAM).

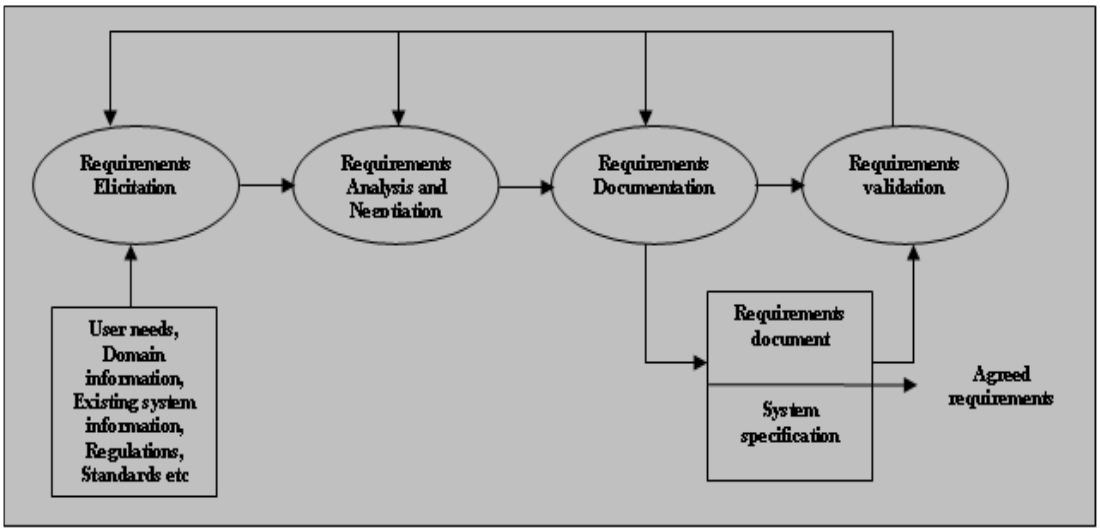


Figure 2-1 Coarse-grain activity model of the requirements engineering process [3]

#### 2.1.1.1 Requirements Elicitation

Requirements elicitation is an activity in which system requirements are discovered. These requirements are elicited from the potential stakeholders of the system or derived from the primary requirements. This activity consists of several sub-activities including identifying sources and stakeholders, selecting suitable technique (s) to elicit requirements, and finally eliciting requirements. Identifying the right stakeholders and using good elicitation technique is very important to

build right system. There are various elicitation techniques available which can enhance the chance of identifying the right requirements. Interviewing, brainstorming, collaborative, workshops, prototyping, questionnaires, observation, and modelling are few of the elicitation techniques [15]. Elicitation techniques are complementary in nature and are generally used in hybrid form. Interviews and prototypes are elicitation techniques that can provide better requirements.

#### 2.1.1.2 Requirements Analysis and Negotiation

Requirements analysis and negotiation is the process of finding problems between the requirements and negotiating them with concerned stakeholders for a common solution. This process runs in parallel with requirements elicitation, as problems can be identified while gathering requirements from the stakeholders and can be negotiated at the same time. However, this process revolves around the initial draft of requirements list, where a group of requirements analyst carefully review this draft for conflicts and other problems. There are number of techniques available for requirements analysis. Checklist is one of those techniques, in this technique each requirement is assessed against set of predefined questions [3]. Interaction matrices draw interaction among requirements, which helps in identifying requirements conflicts and their overlaps [3].

#### 2.1.1.3 Requirements Specification

The agreed set of requirements is documented in a document, which is known as a requirements specification. Although this name may differ from organization to organization it is supposed to have certain information. According to the IEEE/ANSI 830-1993 standard a requirements document should consist of introduction, a general description, specific requirements, appendences and indexes [3, 16]. Stakeholders of this document may include both technical and non-technical staff. This suggests that requirements should be simple and understandable. There are many formal languages available to describe requirements in visual form, which can produce a better understanding about the requirements. Data Flow Diagrams (DFD), Entity-Relationship Diagrams (ERD), Statecharts, Class Diagrams and Use Cases may be used to represent requirements in visual form [17]. A hybrid approach using both natural and formal languages can enhance the usability of the requirements specification [9].

#### 2.1.1.4 Requirements Validation

Requirements validation is a process in which various stakeholders of the system analyse requirements document for consistency, completeness and accuracy [3]. According to Sommerville et al. [3] Requirements analysis is concerned with “Have we got the right requirements?”, whereas requirements validation seek to answer “Have we got the requirements right?”. Like other processes, requirements validation also has some techniques which increase its efficiency. Requirements reviews and prototyping are mostly used validation techniques. In requirements reviews, a group of stakeholders reviews requirements documents for above said anomalies, discuss them and find agreed solution. The agreed set of actions is then performed to update requirement specification. Building a prototype for requirements validation is quite useful as it can exhibit how requirements will look like in an actual implementation. It is recommended to reuse prototype build during requirements elicitation.

#### 2.1.1.5 Requirements Management

Requirement management process is concerned with the activities required to accommodate changed requirements. Changes in requirements are always expected but we can decrease them by certain actions. System requirements can

be categorized into stable and volatile requirements [3]. Unlike stable requirements, volatile requirements are expected to change over the time. A careful handling of volatile requirements at early stage can decrease chance of change in those requirements. Whenever a change is required, an impact analysis is done which shows how much this change will cost. Once, impact of the change is agreed upon it is updated in requirements document. For impact analysis it is imperative to manage relationship between requirements. Tool support e.g. relational databases can help in managing these relationships.

Requirements management activities execute in parallel to other activities of the requirements engineering process, as it starts as soon as we have first draft of the requirements document [3]

### 2.1.2 Requirements Abstraction Model (RAM)

In Market Driven Requirements Engineering (MDRE) [18] requirements are expected to come from various sources and have various levels of abstraction. This makes it hard to manage the requirements. Gorschek and Wohlin [18, 19, 20, 21] have developed the Requirements Abstraction Model (RAM) to manage this continuous arrival of requirements in MDRE environments. Currently RAM exists as RAM version 1.0. The need for this model originated in problems faced at Danaher Motion Särö AB (DHR); however this model is flexible and can be tailored for other organizations.

MDRE development revolves around products in which different versions (releases) are produced over time. In this kind of requirements engineering potential stakeholders may include end users, marketing teams, product manager etc [19]. Requirements elicited from these sources may vary in their abstraction as they are from different domains of knowledge. RAM provides four levels of abstraction on which requirements are placed. These abstraction levels are product level, feature level, functional level and component level as shown in figure 2-2 [18].


Figure 2-2 Requirements Abstraction Model (RAM) Abstraction Levels [18]

Those requirements that are of high abstraction i.e. representing the goals are placed at Product Level in RAM. Product level requirements are of very high abstraction such that they can interact directly with product strategies and in an indirect manner with organizational strategies [18]. This property gives a chance to compare requirements with product and organizational strategies [19]. This comparison can help managers to include, exclude any requirement, moreover to set priorities of the requirements.

The second level of RAM is called the Feature Level. Requirements which are features of the product are placed at this level. Features are characteristics of the

system. Abstract description of these characteristics is present at the feature level [18, 21]. Functional Level contains functional aspect of the requirement i.e. at this level each requirement is described in such a way that it clearly shows what a user or system can do. Description at this level can be used to develop a design of the undergoing system, moreover requirements at this level are supposed to be testable [18, 21]. The last level in RAM is the Component Level. Requirements at this level are present in much detailed form [18].

After having discussed the structure of the RAM now we discuss how it works i.e. which steps are performed to place a requirement on a particular level. There are certain actions which are performed to assign a level to a requirement; these steps are shown in figure 2-3 [18].

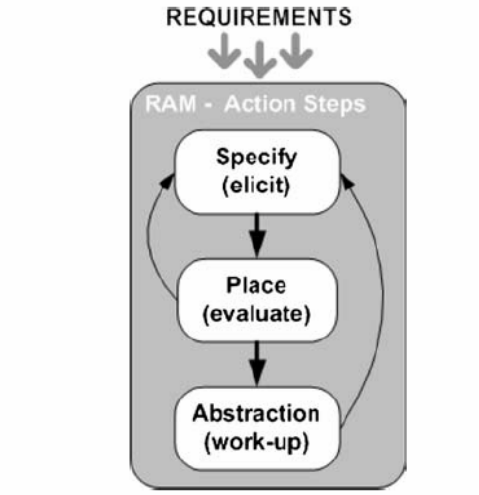


Figure 2-3 RAM Action Steps [18]

RAM consists of three action steps. First is to specify requirements from identified sources. For each requirement its Title, Description, Reason/Benefit/Rationale and Restrictions/Risks are stated. Place (evaluate) is the second action step in RAM in which each requirement is examined for its placement on one of the four levels i.e. product level, feature level, functional level and component level.
Main goal of the RAM is to make each requirement comparable to product strategies [18], abstraction (work-up) the last action step of the RAM helps to achieve this goal. In abstraction (work-up) activities work-up requirements are created. Work-up requirements are those requirements which are created in order to link an existing requirement to requirements at different levels, hence to the product strategies. The underline theme of this level exists in two rules defined in RAM [18, 21]

R1. “No requirement may exist without having a connection to the Product Level.”
R2. “All requirements have to be broken down to Function Level.”

To comply with R1, each requirement at lower level is abstracted upward; new requirements are created for this purpose. Requirements at each level are linked with these new requirements or existing requirements. Creating abstraction in the requirements results into links between component, function, feature and product level requirements. To fulfil rule R2, requirements at higher levels in the RAM are broken and linked down to the function level requirements. Requirements at higher levels are linked with newly created requirements or existing requirements at lower levels.
After these work-up steps each original requirement has upward and downward link. Removal of a requirement from this chain needs to execute work-up actions steps again so that whole chain should be deleted or remaining requirements in the chain should be re-linked to any other requirement.

Unlike at the Specify (elicit) activity where we need only four attributes to mention, in abstraction (work-up) activity we need to mention more attributes. These attributes include Requirement Source, Requirement Owner, Requirements Manager, Relation/dependency, State, Reject Reason, Due Date, Version, Date of creation and Last changed [18]. Adding these attributes enhance usability of the RAM e.g. Requirement Source, Requirement Owner, Requirements Manager can be used for requirements traceability [18]. More detail about RAM with example can be found in [18, 21]

## 2.2 Software Testing

Testing a software system is a process which makes sure the system meets its specification and/or expectations of the users. Some formal definitions of the software testing which capture different aspects are given below:

1. "Testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results." [22]
2. "Testing is the process of executing a program with the intent of finding errors." [23]

Software testing is a verification and validation technique in which a software system is verified and validated against what it was supposed to do. In this activity a set of test cases are executed on software system to find defects in it. It is impossible to find all defects but good test cases are those which can discover the maximum no of defects. Number of test cases depends upon the selected testing technique.

Test planning, designing, execution and evaluation are activities of the test process. Test planning defines test strategies, resource utilization, time estimations, test estimations, technique selection, tool selection, potential risks and completion criteria. Estimations are important tasks in test planning. Calculating the total number of tests is not an easy task as it varies from project to project. Number of tests depends upon number and nature of requirements to be tested. For critical systems every requirements is of great worth and the whole system is required to be tested. Another type of estimation required at test planning is estimating tests execution time. Execution time includes time spend on developing test environment, executing tests and reporting test results [24]. In addition test development time is also estimated, this time includes developing draft set of tests and time required to update these tests if any change is required once these have been executed [24]. Test designing involves designing test cases based on selected techniques and strategy, building test environment and developing test procedures. It is recommends to document all these activities on different test design documents. In test execution set of designed test cases are executed on the system. Once bug is found it is documented on report sheets and is forward to the development team for fixing. Data collected during test execution is mainly used for two purposes. Firstly, data related to bugs found is used to make changes in the system implementation i.e. bug fixing. Secondly, execution results are analysed. Based on this analysis test plan and design are updated to make them more effective.

The test process activities described above may be executed at different times in the system development life cycle. The following section introduces different levels of the testing and describes at what time different activities of the test process are executed.

### 2.2.1 Testing Levels

During software system development testing is performed at different times. Different kind of test cases are designed and executed in different stages. Testing at each stage or level has certain goals; test cases for different levels vary due to these goals. Rakitin [24] has divided testing into five levels i.e. Unit testing, Integration testing, System testing, Alpha and Beta testing, and Acceptance testing.

#### 2.2.1.1 Unit Testing

Unit testing also know as Module testing is performed on individual units of a software system in an isolated environment. The objective of unit testing is to make sure that each unit works as it was defined to work in its specification. Generally, this is considered as an informal activity as software developers test modules while developing them [25]. But, in those cases where unit testing is regard as formal activity, development team is responsible for testing under supervision of the team lead. Team lead is responsible for test planning and test design [22]. In its informal form it is very difficult to distinguish unit testing from program debugging as developers test their own build programs based on their own designed test cases. Huge number of tests are required to test all individual units of the system, this makes unit testing a laborious activity. There are many frameworks available which support unit testing to make it simple. Unit testing frameworks are available in various programming languages, few of these frameworks are AS2Unit, OakUT and JUnit. List of unit testing framework can be found in [28].

It is recommended to use checklist while designing test cases for unit testing, Rakitin [24] recommends considering following areas while developing test cases:

- Algorithms and logic
- Data structures (global and local)
- Interfaces
- Independent paths
- Boundary conditions
- Error handling

#### 2.2.1.2 Integration Testing

When different units (modules) of a software system are combined to interact with each other new bugs are expected. Integration testing is performed in an incremental way to find interface related bugs. As we have discussed earlier that each level has its own goals, there are two goals at integration level [25]:

- To identify bugs that produced when different modules interact with each other.
- To combine all modules so that a complete system is available on which system testing can be applied.

Integration testing can be carried out in two ways the top-down approach and the bottom-up approach.

In the top-down approach testing starts from the main module and the process executes until all of the desired modules have been integrated [24]. Starting from the main module which acts as driver, its stubs are replaced with other subordinate modules. Integration tests are executed once a module has been joined.

Inversely, bottom-up integration testing starts from the low level modules, these modules do not call other modules. Modules at one level are clustered and tested with specially designed drivers. Process executes until all of the modules have been integrated and tested. An obvious drawback with this approach is that complete system is not available until last module is added [23, 24, 25].

#### 2.2.1.3 System Testing

System testing, as name suggests is to test a system as a whole. The obvious goal of this testing level is to validate the system against its requirements specification [24]. Unlike previous levels, the system at this level is validated against both functional and non-functional requirements. Since testing at this level is concerned with the requirements elicited during requirements engineering process, it is recommended to strive for well defined and testable requirements. In addition, it is of great worth to plan and design test cases for system testing as soon as the requirements document is ready.

Generally, dedicated testing teams are responsible for planning and designing test cases for system testing. Furthermore, system is validated in a dedicate test or live environment [22]. In view of the fact that system will be handed over to customer after validation testing, test team is required to validate every aspect of the system. There are many techniques which are used for system testing; these techniques will be discussed in the following sections.

#### 2.2.1.4 Alpha and Beta Testing

At this level of testing, pre-release version of the system is handed over to a set of customers (users) for testing. Involvement of customer at this level provides an opportunity for them to test system with respect to their own expectation. Nevertheless, this is optional to conduct alpha or beta testing. Generally, customers are not ready to participate in such testing because [24]:

- Customers are not sure that identified bugs will be fixed
- Time and resource un-availability can stop customers to participate in this type of testing.

#### 2.2.1.5 Acceptance Testing

Once system testing is over, customer is asked to test the system for acceptance. Customers can test the system themselves by executing the set of test cases. However, customer and testers can perform the acceptance testing together too. System testing and acceptance testing have much in similar except degree of customer involvement. In acceptance testing customer is responsible for executing test cases. Test cases designed during system testing can be used by the customer for this type of testing. Otherwise, customers can ask for their own designed tests, in this case it is recommended to use these test cases during system testing [24]. Like alpha or beta testing, in acceptance testing customers get opportunity to test system for their own expectations. So, goal at this level of testing should be to meet the expectations of the customer.

### 2.2.2 Testing Techniques

Many testing techniques are available which can be used by testers to plan and design test cases. Watkins [22] has divided these testing techniques into three categories; listing of some testing techniques with respect to their category is given in following table 2-2. White box and black box are more like a methodology which use techniques from functional and non-functional categories.

| **General Testing Techniques** | **Functional Testing Techniques** | **Non-Functional Testing Techniques** |
|---|---|---|
| White Box & Black Box | Equivalence Partitioning | Documentation and Help Testing |
| Positive and Negative | Boundary Analysis | Fault Recovery Testing |
| Error Guessing | Intrusive Testing | Performance Testing |
| Automated | Random Testing | Reliability Testing |
| | State Transition Testing | Security Testing |
| | Static Testing | Load Testing |
| | Thread Testing | Usability Testing |
| | | Volume Testing. |

Table 2-2 Testing Techniques [22]

Let us now have a look at some of the testing techniques which will be used in this study later, detail about all above said techniques can be found in [22].

- White Box Testing

  Also know as glass box testing allows testers to test internal structure of the system. Test cases for this kind of testing are designed by using design and implementation artefacts. This makes white box testing techniques most valuable when finding design, logic and sequence, initialization and dataflow defects [25].

- Black Box Testing

  Unlike white box testing, this is not concerned with internal structure of the system instead it validates system against its requirements specification [2]. Testers design test cases based on system requirements. For this kind of testing design and coding knowledge is not required. In general this testing technique is used at integration and system level, however it can be used at unit level too [25].

- Performance Testing

  Generally, non-functional requirements of the system can be test when whole system is available. This implies that performance testing is done at system level. Performance may include many parameters like response time, throughput and memory utilization; to test these parameters specially designed tools are used. [22].

- Exploratory Testing

  Exploratory testing can be defined as "simultaneous learning, test design, and test execution" [29]. In this testing technique testers explore the system and the design and execute tests during this exploration. Test plans and designs in exploratory testing are informal as they are planned at run time. Generally, in this type of testing tests are designed based on the results of the previous tests [29]. Weak areas (where probability of finding bugs is higher) can be starting point for this type of testing. Testers learn as they proceed the testing. During testing testers are supposed to be creative in designing the tests as they are to design tests according to the condition of the system at a particular time. Exploratory testing is best option for situations where less time is available for testing.

### 2.2.3 V-Model

Generally, testing is considered the last phase in the traditional system development life cycle. But, starting testing activities at early stage is beneficial. Finding and fixing the bug at the early stage is less expensive than fixing it at later stages. The V-Model which is based on the conventional waterfall model maps testing activities with phases of the waterfall model. Figure 2-4 shows relationship between different phase of the model and corresponding testing activities.

Purpose of embedding testing activities with conventional development phases is to increase quality of the artefacts at these phases. In addition V-Model emphasis on testing smaller parts before embedding them into larger parts that is why this model starts testing activities as soon as we have system requirements [26]. Testing at different phases has its own test planning and set of test cases however, test plan at each phase is a subset of master test plan. Detecting and fixing defects at early stage are less expensive than at other phases of the development life cycle [8].

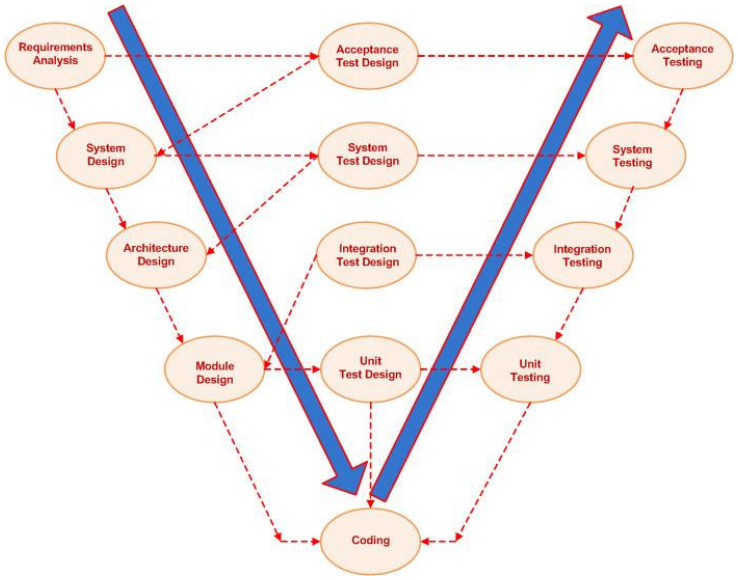

Figure 2-4 V-Model [27]

Testing activities mapped by the V-Model can be used to test both static and dynamic aspects of the software. Static testing does not require execution of the system and is used to test artefacts like requirements specification, design documents and code. To test dynamic aspect of the system test cases are used by executing the system. This depicts that each deliverable on a phase is tested before it is passed to the next phase; it makes all artefacts aligned with system requirements from start to end.

Tests can be planned and designed at early stages of the system development saving valuable time. Drawback in using this model is that testing in this model is document-driven [26]. In many cases documenting each testing activity is not possible especially when less time is available to develop a system.

### 2.2.4 Requirements-Based Testing

Good requirements are a critical factor in the success of a software system. Studies have shown that in many cases requirements are directly responsible for system failure [3]. Requirements document is always expected to have inconsistencies [30]. In case of generating test cases from requirements it is of great importance to have testable requirements. According to Sommerville et al. testable requirement provides opportunity to define one or more test cases, executing these test cases shows that requirement has been implemented in the system [3]. Requirements-Based testing (RBT) is a process that validates requirements testability and generates tests from those requirements [37]. Using the RBT process at early stages can enhance the correctness of the requirements specification along with achieving better test cases. RBT consists of two main activities i.e. making requirements testable and generating system test cases from these requirements. These activities are executed at early stages in software development process. The output of these activities is a complete and consistent requirements document [7].

In making requirements testable, requirements are made clear, consistent and correct. Testable requirements should have following characteristics [4]:

- **Correct**

  A requirement should be valid according to the needs of the stakeholder and it should be specified in a standard way.
- **Unambiguous**

  Requirements that provide clear understanding about the needs of the particular stakeholder are unambiguous.

- **Complete**
  A complete requirement contains all necessary information required to describe it in clear manners.
- **Consistent**
  Requirement is properly linked with business goals and follows specification standards.

Following eight testing activities are performed in the second part of the RBT process [38]:

- **Define Test Completion Criteria**
  Testing is an expensive activity, it is important to know when to stop testing. A test completion criterion is defined to avoid unnecessary prolongation of the testing activity.
- **Design Test Cases**
  Test cases are designed from the testable requirements; each test case has preconditions, test data, the inputs, expected outcomes and post conditions.
- **Build Test Cases**
  Build test cases includes gathering test data and providing component support required for the test execution.
- **Execute Tests**
  Test cases are executed on the system, and results are documented.
- **Verify Test Results**
  Results gathered through the test execution are verified against expected results.
- **Verify Test Coverage**
  Analyse the amount of test cases that have been executed and test coverage criteria.
- **Manage and Track Defects**
  Defects found by executing tests are managed in some database; this can help in defect tracking.
- **Manage the Test Library**
  A test library is a database which contains relationship between tests and the program being tested, tests that have been executed, tests that have not been executed, passed tests and failed tests.

Requirements based test generation seems very cost effective as it helps in producing test cases at early stage. Starting testing activities at early stages can reduce cost and efforts; this has been elaborated in chapter 1 in much detail. Moreover, test cases based on requirements can test the system in better way as they are generated directly from customer requirements.

Many different techniques can be used to make requirements correct, unambiguous, complete and consistent. Prototyping is one of those techniques, this helps in listing true requirements of the customer as this aids customers to describe requirements in better way. An early stage prototyping uses basic form of prototype. Reviewing requirements specification is another way to make requirements testable, in reviewing process requirements specification document is reviewed by selected entities involved in requirements engineering process. These entities review the document and make their recommendations. After discussing with other stakeholders of the document these recommendations may be implemented or rejected otherwise.

Requirements based testing (RBT) is one of the potential opportunities which can pave way for the test decomposition. However, this field of research is facing many problems. Challenges highlighted by Hsia et al. [6] are as follow:

- Generating test cases from requirements specification is still a problem, requirements specification is not standardized enough that it can help in producing quality and right no of test cases for a system.
- Requirements coverage i.e. how much requirements have been satisfied by a test case is still a challenge being faced.
- Generating and maintaining requirements traceability links automatically requires more efforts. Some tools are available but more reliability is necessary in order to provide proper traces between system tests and requirements.

As it has been described earlier that in RBT system tests are derived from requirements, so it is of great importance that how these requirements have been specified. Requirements specifications using formal languages to describe requirements are much suitable for generating tests. Specifications in natural language are generally blurry and hard to produce better tests, especially for automatic test case generation where test cases are derived with standard criteria. However, using only formal languages to specify requirements are difficult to understand for various stakeholder of the document. This may suit system engineers to understand technical diagrams and logics used to depict requirements. To make a median way it is better to use both natural language and formal languages at the same time [9]. Having provided models and diagrams with description in natural language can enhance the understand-ability of the document. Better understanding with the requirements specification leads to make requirements better testable. Stakeholders may judge requirement's correctness, ambiguity, completeness and consistency in better way.

# 3 TEST CASES FROM REQUIREMENTS ABSTRACTION MODEL REQUIREMENTS

In this chapter we investigate how requirements at different levels in the RAM can be used to create test cases for higher level testing i.e. system and acceptance testing. This chapter contains a document analysis of the RAM requirements document. Details of this document analysis are given in the following sections.

## 3.1 RAM Document Analysis

Requirements in the RAM exist at four different levels i.e. product, feature, function and component. The level of detail for the requirements varies from component to product level. A RAM requirements document contains various kind of information for each requirement such as description of the requirement, its level in the RAM, and its relation to other requirements at adjacent levels. We have conducted a document analysis of a RAM requirements document to analyze how the design of tests differs when it is based on RAM requirements at different levels, from component level to product level.

This document analysis will help in determining the suitability of the RAM model for generating tests for system and acceptance level testing. In requirements based testing tests are generated at early phases of the development life cycle. This document analysis will help in understanding the appropriateness of the structure of requirements given in the RAM for requirements based testing. Following sections will provide detail about the test case template, requirements document and the method selected for this document analysis.

### 3.1.1 Test Case Specification

We use a test case template to document the test cases. Test case template consists of a set of elements that are filled for each test case. The test case template selected for this document analysis is an extension of the IEEE 829-1998 [33] test case specification standard. The original test case specification given in the IEEE 829-1998 [33] consists of the following elements:

a) Test case specification identifier;
b) Test items;
c) Input specifications;
d) Output specifications;
e) Environmental needs;
f) Special procedural requirements;
g) Intercase dependencies.

IEEE 829-1998 test case specification template is of theoretical nature and does not describe how the actual execution of the test case will occur. To realize the actual execution of the test case, we have extended the template. Our extended template is given below.

To actually execute at test case it is necessary that test case has preconditions, states test data, defines what steps are required and states expected results. Thus, if a test case contains this all information it is ready for execution and automation. Requirements will be scrutinized to find information for the fields of the following test case template:

1. **Identification**

A unique identifier of the test case

2. **Test Case Purpose**

A brief description of the purpose of the test case.

**3. Test item**
Brief description of the items and features of the system selected to be tested through this test case.
**4. Preconditions**
Assumption or conditions required to be met before the execution of the test.
**5. Inputs**
List of input data required to execute the test case, this may include variable values, files etc.
**6. Steps**
Listing of the steps required to carry out the test.
**7. Expected Results**
Expected output resulted from the execution of the test case.
**8. Environmental Needs**
Hardware and software required to run the test case.

The template for specifying a test case varies from organization to organization. Some of the important elements of the test case template are test purpose, precondition, input, steps and expected results. Additionally, the selected template for this document analysis contains these elements.

It is quite difficult to identify what exact information is required to make a test case executable. However we will consider a test case executable if it contains detailed information for much of the fields of the above test case template. Additionally, it will be easy to automate a test if it contains much detail for all of the above fields. Let us see how these fields are necessary for the actual execution of the tests.

Identification, test case purpose and test item are general part of the test case and these items should be filled for each test case, this will help in maintaining repository of the test cases. The test case purpose is necessary as this will help in the end to know whether test goals have been achieved or not. Moreover, it is very important to identify the preconditions of the test case as they describe the assumptions and conditions required to be met before the test case execution starts. Input is a necessary part of the tests as it states the inputs required to execute a test case. To execute a test it is necessary to identify the steps required to run the test case. Especially in test automation where test cases are executed automatically it is of great importance to clearly state the step required to run the test. A test is said to be pass or fail after matching actual results with expected outcome. Since the purpose of a test is to validate whether system meets its specification or not, this is analyzed with the help of actual and expected results. It is necessary to state the hardware and software needs of the test; otherwise it may not be possible to run the test case.

In addition to the selected fields there are many more which are used while writing a test case e.g. owner of the test case, version number. However, there is no need to specify owner of the test case and version number for this document analysis, as this study is concerned with the extraction of information from the description of the requirements.

### 3.1.2 Requirements Document

A requirements document that contains RAM requirements from a Course Management System (CMS) [34] will be used for this document analysis. CMS is a stand-alone intranet solution for universities to facilitate teachers and students with course related features. These features include information regarding course, course news, schedule, discussion forums and management of course participants etc. The users for this CMS system are Course Manager, Course Tutor, Course Participants and System Administrator. The system can be accessed by a large number of users, and can handle approximately 1 million accesses each day. More detail about the CMS can be found in Appendix 1.

CMS requirements document selected for this document analysis contains requirements description, level of the RAM at which requirement resides and, relationships with the other requirements. Requirements have been placed at their levels according to the criteria given in the manual of the RAM model. The document consists of 183 requirements; it is quite difficult to design and analyse test cases from all of the requirements. However, some requirements will be selected randomly.

### 3.1.3 Method

Each requirement from the selected requirements will be scanned manually to get information required to fill the fields of the selected test case specification. Requirements descriptions, their rationale and restrictions on them are the main source of information. Rationale and restrictions are optional fields and are not available for each requirement in the document. Requirement description, rationale and restriction will be used to design the test cases. The text in these sections will be scanned to fill the selected test case template. Once the test case template has been filled with the information extracted from the requirement it will be analyzed. The test case will be analyzed for the level of detail it contains and will be graded on a 5-1 scale. The scale of 5-1 grading will be as follow:

**5**: Test case contains detailed information for each of its field. Test case has clear preconditions, correct input data, obvious steps to run the test case and comprehensible expected outcomes. And, most of the information has been extracted directly from the requirement (s) from which the test case has been created.

**4**: Test case has information for almost all of fields and it is in executable form. But, some information is missing which is not directly available from the requirement (s). This missing information can easily be gathered from other requirements or from the context of the requirement.

**3**: Test case lacks information and is not in executable form. However, missing information can be inferred from other requirements or from the context of the requirement. Adding this information can make the test case executable.

**2**: Test case has some information extracted directly from the given description of the requirement but it cannot be considered as executable. Moreover, it is not possible to extract more information for this test case from other requirements in the requirements document.

**1**: Test case is severely incomplete and does not provide clear understanding how to test the requirement. In certain cases information given with the requirements does not provide understanding about what to test and how to test. It is quite difficult to design test cases from such requirements. However, the test cases designed from such requirements are very ambiguous and generally state the test purpose (goal) only.

In this way test cases will be designed from each selected requirement, these test cases will be analyzed and graded according to the above scale. This process will be executed for all the selected requirements. Once, finished with designing and grading test cases each requirement will be analyzed for the grade of its test cases. This analysis of requirements and their test cases will help in finding the answer to the following question:

How much information for tests can we get from requirements present at different levels of abstraction (product, feature, function and component) in RAM?

Purpose of this document analysis is to study how tests are affected with the change in requirements abstraction from component to product level. Answering to the above questions will help in achieving the goal of this document analysis.

## 3.2 Applying Method on RAM Requirements Document

In this section the above stated method will be applied on the selected RAM requirements. This section will describe how we sampled the requirements, wrote test cases from them and then evaluated and graded these test cases.

### 3.2.1 RAM Requirements

Five random requirements will be selected from each level of the RAM. In total there will be 20 requirements as the RAM consists of four levels of requirements abstraction. Since, the RAM requirements document has 183 requirements and they may result more than one test cases for a single requirements. It is not possible to apply this method for each requirement in the document. Looking at the scope of this master level study, 20 requirements (with 5 requirements from each product, feature, function and component level) will provide enough understanding about the suitability of these requirements for generating test cases.

Sample requirements have been selected randomly to overcome any unfairness which may occur by choosing the requirements suitable to predefined concept or results. Above stated method will be applied on the following requirements. Each requirement has an id, its level in the RAM at which that requirement exists, a title, description of the requirement, rationale and restrictions on the requirements. The rationale element describes the reason for the requirement to be included in the system. Restriction explains the constraints on the requirements.
All these fields have been selected from the original document selected for the document analysis. Test cases will be designed by using the text given in requirements descriptions, rationale and restriction fields of the RAM document.

| ReqID | Level | Title | Description | Rationale | Restriction |
|---|---|---|---|---|---|
| 1 | Product | Distribute Information about the course | The product shall provide information to interested authorized system users about course. | | Please define “interested parties” |
| 2 | Product | Secure Product | The product shall prevent unauthorized use. | Users shall only be able to access predefined sets of functionality. Users must be registered in the system before gaining access. | |
| 3 | Product | Swedish Market | The product is targeted towards Swedish market | | This may change in coming releases of the system. |
| 4 | Product | Product interface | All access to the system must take place via the system own user interface, i.e. access from third party product is not allowed. | Control look and feel. Avoid security issues and compatibility problems. | |
| 5 | Product | Course | The product shall provide | | |

| | | | | | |
|---|---|---|---|---|---|
| | | Participant Administration | functionality to administer the participants of a course. | | |
| 6 | Feature | Course File Archive | A course shall have a file archive | Ability to distribute material to the course participants. | File may be very large, which consumes resources that may be scarce. |
| 7 | Feature | Course News | It shall be possible to attach news items to a course. | Keep the students informed about the course. | |
| 8 | Feature | User Calendar | It shall be possible for users of the system to have and maintain a personal calendar, but only accessible within the product. | Allow users to plan their time | |
| 9 | Feature | Product Access | Only users with the right privileges shall be able to view, add, edit or remove contents in the product. | | |
| 10 | Feature | Personal Profile | Users in the system shall have a Personal Profile. | Allow other users to find contact information, Facilitate interaction between users. | |
| 11 | Function | Login | The user must login before the product usage. | Unauthorized users should not have access to product functionality. Customize information for user. | Usability threat: user may forget login id. |
| 12 | Function | Course Start Page Contents | The course start page shall contain: course news, link to course participators, link to course information, link to course literature list, link to course links, link to course discussion forum. | | |
| 13 | Function | Add Course Links | It shall be possible to add links to the list of links attached to a course | | |
| 14 | Function | Access to add and remove course participators | Only system administrators or the course administrator of a course shall be able to add and remove participators to and from a specific course. | | |
| 15 | Function | Access to View Social Security Number | Only the user, the course administrator and the system administrator shall be able to view the social security number. | Maintain personal integrity | |
| 16 | Component | Incorrect Login | If the user enters an incorrect user id and /or password the login page shall be reloaded with information | Feedback to user | User may not understand information |

| | | | | | |
|---|---|---|---|---|---|
| | | | showing that an incorrect login has been attempted | | |
| 17 | Component | Restriction for Using the Product | A computer accessing the product must comply to the following minimum requirements: screen resolution of 800x600 or higher, web browser that supports CSS 2, HTML 4.0 and JavaScript. | | |
| 18 | Component | Reply to Message | When viewing a message, it shall be possible to reply directly to the message. This adds a new message with the same subject as the message currently viewed. | | |
| 19 | Component | Successful Login | When a user has successfully logged in, the product shall display the user's personal start page. | After user authentication, the user wants to start working. | |
| 20 | Component | Content of Message | A message shall contain: date and time of post, author, subject, contents. | | |

Table 3-1 Selected RAM Requirements

### 3.2.2 Test Cases

In this section test cases will be designed from the requirements given in table 3-1. Designing test cases from the requirements mainly consists of two steps. Firstly, extract information for the test case from the description of the given requirement. If test case lacks information then analyze all other, related requirements in the requirements document to extract more information to make the test case more detailed. In this way test cases will be designed from each selected requirement. These test cases will be discussed one by one and based on the grading criteria each test case will be assigned a suitable grade. Information for each test case will be extracted directly from the requirements, from the context of the requirements and/or from the other requirements in the RAM requirements document. Information which is extracted from other requirements will be shown in italic format.

| | |
|---|---|
| **Test Case ID** | TC.REQ1.1 |
| **Test Case Purpose** | Test that registered user can access the course information |
| **Customer Requirement** | 1 |
| **Preconditions** | User is registered |
| **Inputs** | Course (id or name) |
| **Steps** | - |
| **Expected Results** | - |
| **Environmental Needs** | - |

Table 3-2 Test Case for ReqID 1

Table 3-2 consists of a test case designed from the product level requirement with Req ID 1. This test case will validate that a registered user can view the information about the course. The test case template contains the information extracted from the description of the requirement given in RAM requirements document.

It is obvious from the information given in the test case that this test is lacking some critical information. For example the test case dose not state which information about the course will be tested. This is due to the lack of information given for this requirement; the requirement description does not state which information will be accessible to the users of the system. Due to the lack of information it is not possible to

identify the expected results for this test case. Moreover it is also difficult to identify steps required to run this test case as it is not clear from the requirement that from where users can access the information.

For this test case we can see that test purpose can be extracted from the description of the given requirement. Along with the test goal we can also get some values for preconditions and inputs for the tests. From description we can find that user should be registered with the system and user can find information about the course. But with this information it is not possible to execute this test as it lacks expected results and possible steps required to run this test case.

It is not possible to get more information for this test by looking at other requirements in the document. This test case cannot be considered as complete test case but it has a test purpose and some other information like preconditions and inputs. Hence, this test case can be assigned grade 2.

| | |
|---|---|
| **Test Case ID** | TC.REQ2.1 |
| **Test Case Purpose** | Test that unregistered user cannot access the system functionality. |
| **Customer Requirement** | 2 |
| **Preconditions** | User is not registered |
| **Inputs** | *System URL* |
| **Steps** | *1. Open web browser*<br>*2. Enter system URL*<br>*3. Click on Go button* |
| **Expected Results** | - |
| **Environmental Needs** | - |

Table 3-3 Test Case for ReqID 2

TC.REQ2.1 test case has been designed from the requirement with id=2. Information provided for this requirement is of very abstract nature. From the description of the requirement and its rationale only the test purpose and one precondition can be extracted. From the text given in the rationale field we can extract that user should be unregistered to meet the above set test goal. Other than test case purpose and one precondition this requirement does not give any information for the remaining fields of the test case. It will not be possible to execute this test case with this short information. On the other hand, looking at some other requirements in the document it is possible to add some useful information for this test case. With the given requirement it is not possible to know how system will be accessed. But, "Web-based user interface" requirement can help in knowing that system will be accessed through web by providing URL. Expected results are not available for this test case. Carrying out a test case without knowing its expected results will make it difficult to know whether test has passed or not. Therefore, it can be concluded that this test case is not executable though has some information for few of its field, and it resides at grade 2 of the selected scale.

| | |
|---|---|
| **Test Case ID** | TC.REQ3.1 |
| **Test Case Purpose** | *Test that system supports Swedish language* |
| **Customer Requirement** | 3 |
| **Preconditions** | - |
| **Inputs** | - |
| **Steps** | - |
| **Expected Results** | - |
| **Environmental Needs** | - |

Table 3-4 Test Case for ReqID 3

It is obvious from the test case specification for TC.REQ3.1 that this test case severely lacks information. Requirement from which this tests case has been designed is very abstract, even it is difficult to extract any test purpose. Test purpose for this test has been stated by exploring other requirements in the RAM requirements document. From "User interface language" and "Support Swedish alphabets" requirement it can be implied that system should support Swedish language. So, it can be concluded that the given requirement provides very limited information for test case. With this conclusion TC.REQ3.1 acquires grade 1.

| | |
|---|---|
| **Test Case ID** | TC.REQ4.1 |
| **Test Case Purpose** | Test that system cannot be accessed through third party products. |
| **Customer Requirement** | 4 |
| **Preconditions** | - |
| **Inputs** | - |
| **Steps** | - |
| **Expected Results** | - |
| **Environmental Needs** | - |

Table 3-5 Test Case for ReqID 4

TC.REQ4.1 has been designed from the "Product interface" requirement. This test case has value for test case purpose field only. Description of the given requirement is very ambiguous; it does not provide clear understanding about the third party products. It is not obvious that what kind of information is required for third party product to access the system, and how to access the system. It is not possible to make this test case complete neither from the given requirement nor from the other requirements in the RAM requirements document. With this rigorously incomplete information TC.REQ4.1 remains at level 1 on the scale.

| | |
|---|---|
| **Test Case ID** | TC.REQ5.1 |
| **Test Case Purpose** | Test that administrator of a course has access to the course participants management feature |
| **Customer Requirement** | 5 |
| **Preconditions** | - |
| **Inputs** | - |
| **Steps** | - |
| **Expected Results** | - |
| **Environmental Needs** | - |

Table 3-6 Test Case for ReqID 5

This test case has been designed from the "Course participant administration" requirement, like other product level requirements this requirement is ambiguous too. Only test case purpose can be inferred from the give requirement. It does not tell which features participant management consists of, where these features are available in the system. It is not possible to infer information for this test case from the other requirements too. Since this test case is not executable and is severely incomplete so it gets grade 1.

| | |
|---|---|
| **Test Case ID** | TC.REQ6.1 |
| **Test Case Purpose** | Test that a course has a file archive. |
| **Customer Requirement** | 6 |
| **Preconditions** | Course exits |
| **Inputs** | Course (id or name), archive name |
| **Steps** | *1. Go to course page* |

| | |
|---|---|
| | *2. Look for file archive link*<br>*3. Click on file archive link* |
| **Expected Results** | - |
| **Environmental Needs** | - |

Table 3-7 Test Case for ReqID 6

TC.REQ6.1 test case has been designed from the requirement having id 6. With very short description given for this requirement it is quite difficult to get much for the test case. Even it is very difficult to make a clear test objective for the test. One possible test objective which can be design from this short information is "Test that a course has a file archive" but this is very abstract in nature and does not provide clear understanding how to test. It is not possible to identify possible steps required to run this test case.

Since this test case lacks information for various fields like it does not have steps required to run the test case and it does not contain expected results, so this tests cannot be said as an executable test case. However, looking at some other requirements in the RAM document we can get more information to make this test case executable. Probing the "Add files to course file archive" and "Access to use course file archive" we can find some possible steps required to execute this test. Information in italic font shows that this information has been taken from other requirements in the RAM document. Making this test executable by adding information from other requirements puts the test at grade 3 on the scale.

| | |
|---|---|
| **Test Case ID** | TC.REQ7.1 |
| **Test Case Purpose** | Test that news can be added to a course |
| **Customer Requirement** | 7 |
| **Preconditions** | Course exists, *Course Administrator exists* |
| **Inputs** | Course (id or name), News, |
| **Steps** | - |
| **Expected Results** | - |
| **Environmental Needs** | - |

Table 3-8 Test Case for ReqID 7

Test case in table 3-8 has been designed from the requirement "Course news". Preconditions and inputs to the test can easily be identified form the given requirements. From the description it can be inferred that course should exist so that news can be added to that course. And two possible inputs to the test can be Course (id or name) and News. But, with this information test case cannot be executed, because give requirement does not state how to add the news. The given requirement does not tell who can add the news items. However form the "Access to add, edit and remove course news items" requirement it is clear that only users with given privilege can add the news items. From this requirement another precondition can be identified i.e. course administrator should exist to add news item to the course.

But still it is not clear that how news will be added to the course (steps). Also, no clue about what kind of response course administrator will get if the news has been added successfully or could not be added. This test case holds some information but it is not possible to make it complete, and it gets grade 2.

| | |
|---|---|
| **Test Case ID** | TC.REQ8.1 |
| **Test Case Purpose** | Test that user can access personal calendar when logged in |
| **Customer Requirement** | 8 |
| **Preconditions** | User is logged in |
| **Inputs** | - |
| **Steps** | - |

| | |
|---|---|
| **Expected Results** | - |
| **Environmental Needs** | - |

Table 3-9 Test Case for ReqID 8

TC.REQ8.1 test case has been designed from the requirement with id=8. It can be implied from the description of the requirement that user need to be logged in when accessing the calendar, this has been identified as a precondition for this test. Requirement says that user can have and maintain a personal calendar, but there is no information that how user will access and add calendar to its personal profile, and how user can maintain that calendar. This test case is not in executable form though it has some information, and is ranked as grade 2.

| | |
|---|---|
| **Test Case ID** | TC.REQ9.1 |
| **Test Case Purpose** | Test that a course participator (user) can view course contents. |
| **Customer Requirement** | 9 |
| **Preconditions** | User is registered |
| **Inputs** | Course (id or name), *User id, User password* |
| **Steps** | *1. Enter user id*<br>*2. Enter password*<br>*3. Click on login button*<br>*4. Select the course with given course id or name* |
| **Expected Results** | *User will see a web page containing course news, file archive, course participators, course calendar, course information, course literature list, course links and discussion forum.* |
| **Environmental Needs** | - |

Table 3-10 Test Case for ReqID 9

Table 3-10 contains a test case designed from the "Product access" requirement. Like above requirement this also has many ambiguities e.g. what are those contents, a user can add, edit or remove, and what kind of users with different privileges exists. The only information which can be extracted from the given requirement is that user is registered and course exists. With this short information this test case cannot be termed as executable. However for this test case it is possible to get some information from other requirements in the document. From "Course start page contents" we can identify possible expected results for this test case. In addition "Personal start page contents" and "Successful login" requirements can help in identifying the actions required to view the course contents i.e. steps to run the test case. Text in italic form has been extracted from other requirements in the RAM requirements document.

By adding this information TC.REQ9.1 test case can be categorized as executable test case. Since this test case was made executable by adding information from other requirements so it can be assigned grade 3.

| | |
|---|---|
| **Test Case ID** | TC.REQ10.1 |
| **Test Case Purpose** | Test that a user can access its personal profile. |
| **Customer Requirement** | 10 |
| **Preconditions** | User exists |
| **Inputs** | *User id, User password* |
| **Steps** | *1. Enter user id*<br>*2. Enter password*<br>*3. Click on login button*<br>*4. Click on the personal profile link* |
| **Expected Results** | *User will see a webpage containing first name, last name and* |

| | |
|---|---|
| | *social security number.* |
| **Environmental Needs** | - |

Table 3-11 Test Case for ReqID 10

"Personal profile" requirement contains very short description hence provide very limited information for TC.REQ10.1 test case. From the given description we can identify test goal (purpose) and one precondition that user should exist to view its personal profile. With such short information test cannot be declare as executable. But exploring other requirements in the document it is possible to get more information for this test case. "Basic contents of personal profile", "Extra contents of personal profile", "Edit basic contents of the profile", and "Personal start page contents" can help us in identifying possible expected results and steps required to view personal profile.

Like previous test case it is not possible to make this test case as executable based on the give requirement. But other requirements in the RAM requirements document can be used to make this test case executable. Based on this analysis this test case gets grade 3.

| | |
|---|---|
| **Test Case ID** | TC.REQ11.1 |
| **Test Case Purpose** | To test that system functionality cannot be accessed without providing login information |
| **Customer Requirement** | 11 |
| **Preconditions** | |
| **Inputs** | *System URL* |
| **Steps** | *1. Open web browser*<br>*2. Enter system URL*<br>*3. Click on Go button* |
| **Expected Results** | No access to the system functionality. |
| **Environmental Needs** | *Web browser (Internet Explorer ver>=5)* |

Table 3-12 Test Case for ReqID 11

TC.REQ11.1 test case has been designed from the requirement with id 11. Purpose of the test and the expected results are the only items for this test which can be extracted from the description of the requirement. Description of the requirement clearly states that system functionality can only be accessed through system login; this information helps in making the goal for this test case. If we look at the information given in the rationale field for this requirement we can see that unauthorized users will be refused to access any feature of the system. But, getting the test goal and expected results are not enough to be an executable test case. From this requirement it is not clear how to access the system, and what will be the possible inputs for this test case. However, by looking at the other requirements in the document it is possible to get some information. By adding this information makes this test case more detailed hence executable. For example "Web-based user interface" (another requirement in the RAM document) requirement says that system will be accessed through web-based interface, so we can use this information to determine the environmental needs for this test case. In addition "Support internet explorer" requirement can help in making the environmental needs more specific as this requirement tells that Microsoft Internet Explorer version 5 or greater is required to access the system. It is not possible to identify the input for this test case i.e. System URL from the description of the given requirement, But, exploring the other requirements in the RAM document we can fined that system will be accessed through a URL. Similarly, it is not possible to identify possible steps required to run this test case from the given requirement. But, these steps can be discovered by looking at other requirements. Italic text for the test

case shows that this information has been taken from any other requirement in the RAM document.

From above discussion it can be concluded that test case is not executable unless some information is added from the other requirements. Hence, this test case gets grade 3.

| | |
|---|---|
| **Test Case ID** | TC.REQ12.1 |
| **Test Case Purpose** | Test that a course start page contains link to course discussion forum. |
| **Customer Requirement** | 12 |
| **Preconditions** | Course exists, Discussion forum exists, *user is logged in, User is at personal page.* |
| **Inputs** | Course (id or name), Discussion forum |
| **Steps** | *1. Click on the course name*<br>*2. Click on discussion forum* |
| **Expected Results** | User moves to discussion forum |
| **Environmental Needs** | - |

Table 3-13 Test Case for ReqID 12

TC.REQ12.1 has been created from the "Course start page contents" requirement. From the given description we can easily extract information for preconditions, inputs and expected results. This test case has good information directly extracted from the given requirement. But, there is some information which has been extracted from other requirements. From "Personal start page" and "Course start page contents" requirements we can identify the possible steps required to perform the test. Text in italic exhibits this information has been extracted from other requirements in the RAM requirements document. This test case has detailed information extracted directly from the given requirements, but some information has been included from other requirements too. With this discussion TC.REQ12.1 can be assigned grade 4.

| | |
|---|---|
| **Test Case ID** | TC.REQ13.1 |
| **Test Case Purpose** | Test that a link can be added to the list of links for a course. |
| **Customer Requirement** | 13 |
| **Preconditions** | Course exists, List of links exists, *Course administrator is logged in* |
| **Inputs** | List (name or id ), New link (link to add), Course (id or name) |
| **Steps** | *1. Click on the course*<br>*2. Click on the list of links*<br>*3. Click on add link* |
| **Expected Results** | - |
| **Environmental Needs** | - |

Table 3-14 Test Case for ReqID 13

TC.REQ13.1 test case has been designed from the "Add course links" requirement. A good quantity of information can be implied from the description of the requirement. It can be deduced that course for which a link is to be added should exist, list of links for that course should be there before executing the test. And some possible inputs which can be inferred from the given requirement are list, link to add and the course to which the link is to be added. But, we don't know how to add these links, where these links are available etc. With the present information this test case cannot be termed as executable. However, some information can be extracted from other requirements to make this test more detailed. From "Access to add, edit and remove course links" requirement it is clear that only course administrator can add the link, so a precondition can be added that course administrator should exits, logged in and is at its personal page. Furthermore, with the help of "Course start page contents",

"Access to add, edit and remove course links" and "Course start page contents" we can identify how to add a new link to the course. With this new information test case looks more detailed and can be said as executable. This test case gets grade 3 because it was not possible to make test executable from the given requirement, other requirements were required to make it executable.

| | |
|---|---|
| **Test Case ID** | TC.REQ14.1 |
| **Test Case Purpose** | Test that course administrator can add course participators to the course. |
| **Customer Requirement** | 14 |
| **Preconditions** | Course administrator exists, Course exists, *User is at personal page* |
| **Inputs** | Course (id or name), User (to add as participator) |
| **Steps** | *1. Open course start page*<br>*2. Click on course participator link*<br>*3. Add course participator* |
| **Expected Results** | - |
| **Environmental Needs** | - |

Table 3-15 Test Case for ReqID 14

TC.REQ14.1 test case has been designed from the requirement with id=14. It is possible to extract test goal, preconditions and inputs for this test case from the given requirement. But requirement description does not provide any clue for how to add a course participator i.e. Steps required to carry out this test. Similarly expected results cannot be inferred from the give requirement. Other requirements like "Personal start page", "Course start page" and "Manage course participators" state that course start page contains a link to the course participators management section. Steps required to carry out this test case are given in above table, these steps has been identified with the help of above stated requirements. From the given requirement it is not possible to make this complete, but other requirements in the document can make this test case more detailed i.e. executable. So, TC.REQ14.1 gets grade 3.

| | |
|---|---|
| **Test Case ID** | TC.REQ15.1 |
| **Test Case Purpose** | Test that a user can view its social security number. |
| **Customer Requirement** | 15 |
| **Preconditions** | User exists, User is logged in, User have social security number |
| **Inputs** | - |
| **Steps** | *1. Click on personal profile link*<br>*2. Click on view social security number* |
| **Expected Results** | User can see its social security number |
| **Environmental Needs** | - |

Table 3-16 Test Case for ReqID 15

TC.REQ15.1 test case has been designed from the requirement with id=15. Easily, we can state a purpose of the test case based on the given description of the requirement. It can be implied from the requirement that user should be registered, logged in and has a social security number in the profile. But, this requirement does not provide any clue where and how user can see its social security number. Looking at other requirements in the document we can find that personal profile contains a link to the social security number. "Personal start page" and "Basic contents of personal profile" requirements are helpful in identifying steps required to view the social security number. Without these steps test case is incomplete i.e. not in executable form. It can be concluded that test case has some information but was not in

executable form until information has been added from other requirements. With this conclusion this test case gets grade 3.

| | |
|---|---|
| **Test Case ID** | TC.REQ16.1 |
| **Test Case Purpose** | Test that on providing an invalid login id and password user is shown a message that incorrect login information has been entered |
| **Customer Requirement** | 16 |
| **Preconditions** | User is not logged in |
| **Inputs** | User ID=invaliduser , User Password=invalidpassword |
| **Steps** | 1. Open the system login<br>2. Enter User Id<br>3. Enter User Password<br>4. Press login button |
| **Expected Results** | User shall see a message showing that you have entered an invalid user id and/or password. |
| **Environmental Needs** | - |

Table 3-17 Test Case for ReqID 16

TC.REQ16.1 test generated from the requirement with id 16 is given in table 3.17. This test case has good detail for all of its fields. And almost all of the information for this test case has been implied directly from the given requirement i.e. ReqID 16. Since this test case contains clear test goal, preconditions, inputs, comprehensive steps and obvious expected results so it can be ranked as executable test case with almost all information extracted from the same requirement. With this level of detail it can be assigned grade 5.

| | |
|---|---|
| **Test Case ID** | TC.REQ17.1 |
| **Test Case Purpose** | Test that system can be accessed when screen resolution, browser, CSS, HTML and JavaScript requirement are met. |
| **Customer Requirement** | 17 |
| **Preconditions** | |
| **Inputs** | System URL |
| **Steps** | 1. Set screen resolution to 800x600<br>2. Open Internet Explorer version 6.0 (This version supports CSS 2, HTML 4.0 and JavaScript)<br>2. Enter URL of the system in the address bar<br>3. Press go button |
| **Expected Results** | *User will see system login page* |
| **Environmental Needs** | Internet Explorer version 6.0 |

Table 3-18 Test Case for ReqID 17

This test case holds good level of information for almost every field, extracted directly from the given requirement. Given requirement does not say any thing about expected results, but looking at other requirements in the document we can find expected result for this test case. From “Web-based user interface” and “Login” requirements we can find that user sees login page when system is accessed through web browser. For this test case it is clear that test case has detailed information, and almost all of the information has been extracted directly from the given requirement. So, TC.REQ17.1 gets grade 5.

| | |
|---|---|
| **Test Case ID** | TC.REQ18.1 |
| **Test Case Purpose** | Test that reply to a message adds the same subject as of the message being replied. |

| | |
|---|---|
| **Customer Requirement** | 18 |
| **Preconditions** | Discussion forum exists, Message exists, User is logged in, User is at course start page |
| **Inputs** | Contents of the message |
| **Steps** | *1. Open discussion forum*<br>*2. Open a message*<br>*3. Click on reply message*<br>*4. Click on add reply* |
| **Expected Results** | |
| **Environmental Needs** | - |

Table 3-19 Test Case for ReqID 18

TC.REQ18.1 test case has been designed from the "Reply to message" requirement. From the given description of the requirement it is possible to extract test goal, preconditions and input parameters of the test case. Requirement does not provide any information for other fields of the test case. By looking at other requirements it is possible to get some more information. Requirements "Course start page" and "Add message" can help in identifying the actions required to add reply to a message on discussion forum. Information in italic form has been taken from other requirements not from the given requirement. From table 3-19 it can be analysed that this test case cannot be made executable by using only the given requirements, adding information from other requirements can make it more detailed test case i.e. executable. With this analysis TC.REQ18.1 can be place at grade 3.

| | |
|---|---|
| **Test Case ID** | TC.REQ19.1 |
| **Test Case Purpose** | Test that when a registered user provide correct user id and password user sees personal start page. |
| **Customer Requirement** | 19 |
| **Preconditions** | User exists, User is not logged in |
| **Inputs** | User ID=validuser , User Password=validpassword |
| **Steps** | 1. Open the system login<br>2. Enter User Id<br>3. Enter User Password<br>4. Press login button |
| **Expected Results** | User sees personal start page |
| **Environmental Needs** | - |

Table 3-20 Test Case for ReqID 19

TC.REQ19.1 has been designed from the requirement "Successful login". Description of the requirement is clear enough that it is possible to inferred information for almost every field of the test case. Since, this test case contains detailed information directly extracted from the given requirement so this test case can be assigned grade 5 on the scale.

| | |
|---|---|
| **Test Case ID** | TC.REQ20.1 |
| **Test Case Purpose** | Test that a discussion forum message contains date and time, author, subject and contents. |
| **Customer Requirement** | 20 |
| **Preconditions** | Discussion forum exists, User is at a discussion forum |
| **Inputs** | Date, Time, Author, Subject, Contents |
| **Steps** | 1. Click on add new message<br>2. Enter current date<br>3. Enter current time<br>4. Enter author name<br>5. Provide subject for the message |

| | |
|---|---|
| | 6. Provide content<br>7. Click on post message |
| **Expected Results** | User can view message with date and time, author, subject and contents. |
| **Environmental Needs** | - |

Table 3-21 Test Case for ReqID 20

TC.REQ20.1 test case has been designed from the requirement "Content of message". Requirement contains good level of detail as it provides values for almost all of the fields of the test case. This test case is complete and holds detailed information directly inferred from the given requirement. With this level of detail and use of the given requirement only this test case can be assigned grade 5.

### 3.2.3 Test Case Grades

In the previous section test cases have been generated from the selected requirements. After making an analysis, based on the selected criteria these tests have been graded on a 5-1 scale. Following table consists of these test cases and their grades.

| Requirement Id | RAM Level | Test Case Id | Grade |
|---|---|---|---|
| 1 | Product | TC.REQ1.1 | 2 |
| 2 | Product | TC.REQ2.1 | 2 |
| 3 | Product | TC.REQ3.1 | 1 |
| 4 | Product | TC.REQ4.1 | 1 |
| 5 | Product | TC.REQ5.1 | 1 |
| 6 | Feature | TC.REQ6.1 | 3 |
| 7 | Feature | TC.REQ7.1 | 2 |
| 8 | Feature | TC.REQ8.1 | 2 |
| 9 | Feature | TC.REQ9.1 | 3 |
| 10 | Feature | TC.REQ10.1 | 3 |
| 11 | Function | TC.REQ11.1 | 3 |
| 12 | Function | TC.REQ12.1 | 4 |
| 13 | Function | TC.REQ13.1 | 3 |
| 14 | Function | TC.REQ14.1 | 3 |
| 15 | Function | TC.REQ15.1 | 3 |
| 16 | Component | TC.REQ16.1 | 5 |
| 17 | Component | TC.REQ17.1 | 5 |
| 18 | Component | TC.REQ18.1 | 3 |
| 19 | Component | TC.REQ19.1 | 5 |
| 20 | Component | TC.REQ20.1 | 5 |

Table 3-22 Test Cases Grades

### 3.2.4 RAM Requirements and Test Cases Analysis

In the previous section test cases have been designed from the product, feature, function and component level requirements. Test cases have been graded on a 5-1 scale based on how much information each requirement provides for the test case. In this section requirements will be discussed with their test cases grades to answer the following question.

#### 3.2.4.1 How much information for tests we can get from requirements present at different levels of abstraction (product, feature, function and component) in RAM?

Table 3-22 contains requirements, their test cases and grades for the test cases. The list consists of five requirements from each abstraction level of the RAM. Grading of the test cases has been done based on a 5-1 scale. A test case was assigned grade 5 if it had detailed information (executable) for all of fields of the test case template and all of this information has been inferred directly from the given requirement. If a test case had good detail (executable) but some information has been inferred from the other requirements in the requirements document then test case was allotted grade 4. If a test case did not have enough information (not in executable form) directly available from the given requirement, but information from other requirements could make it complete then test case was assigned grade 3. Grade 2 test cases did not have enough information that they could be termed as executable, other requirements could not make it complete too. A test case which was very incomplete was assigned grade 1. Here we will discuses the selected requirements with the grades of their test cases for each level separately.

- **Product Level**

  Requirements with the ids 1,2,3,4 and 5 in the table 3.1 are product level requirements. Test cases designed from these requirements and their grades are given in table 3-22. Grades of the test cases designed from these requirements are 2, 2, 1, 1 and 1 respectively. Test cases with grade 1 show that their requirements were very ambiguous and could provide very limited information. The only information extracted from these requirements was the test purpose. Grade 2 test cases show that requirements could not provide enough information that they can be said as executable. Analysing test cases with grade 2, we can see that these test cases have test purpose, some information for preconditions and inputs. But requirements from which these test cases have been designed could not provide information about the actions required to run these test cases and their expected results. These test cases are also not in executable form but have some information for few fields of the test case template.

  Three out of the five test cases have grade 1, this shows that many of the selected product level requirements could provide very limited information for the test cases. The other two requirements whose test cases have grade 2 could not provide much information. For both grade 1 and 2 test cases it is obvious that their requirements could not provide enough information that these test cases can be termed as executable. Also, with such short information these test cases cannot be atomized. It can be analyzed that product level requirements are ambiguous and only provide a test purpose. It can be concluded that product level requirements are very abstract and are not in testable form.

- **Feature Level**

  In table 3-1 requirements with ids 6, 7, 8, 9, 10 are feature level requirements. Test cases designed from these requirements are given in the table 3-22. Test cases created from these requirements hold grades 3, 2, 2, 3 and 3. As it has been discussed earlier that test cases with grade 2 do not have enough information that they can be said as executable. Even other requirements cannot help in making them

executable. Requirements from which these test cases have been designed do not offer much information. Requirements 7 and 8 which could provide grade 2 test cases are very abstract and do not give clear understanding about the actions required to carry out these test cases.
Grade 3 test cases are those which are initially not in executable form but adding information from other requirements make them executable. This means, those feature level requirements which have grade 3 test cases cannot make their test cases executable with the help of their own information. Though these feature level requirements cannot produce executable tests but can provide better information than those requirements which produce test cases of grade 2.
It is obvious that feature level requirements which have test cases of grade 2 or 3 cannot produce executable test cases based on their own information. So it can be concluded that feature level requirements do not hold much information and cannot produce executable test cases.

- **Function Level**
  Requirements 11, 12, 13, 14 and 15 in table 3-1 are function level requirements. Test cases designed from these requirements hold grades 3, 4, 3, 3 and 3 respectively. As discussed in previous section, test cases with grade 3 cannot be made executable with the help of given requirement only. Other requirements contribute to make these test cases executable. Four out of five test cases have grade 3. This shows that almost all of the function level requirements do not have enough information that they can produce executable test cases at their own. Though, these requirements provide enough information that by adding information from other requirements test cases can be made executable.
  Grade 4 on the scale shows that the test case is in executable form by extracting information directly from the given requirements, but the test case also holds some information from the other requirements. Such requirements provide enough information that the test case is in executable form but adding information from other requirement can make this test case more efficient. Since, most of the test cases designed from function level requirements hold grade 3, it can be concluded that the function level requirements have much information but they cannot produce executable test cases at their own.

- **Component Level**
  In the table 3-1 the requirements with ids 16, 17, 18, 19 and 20 are component level requirements. Grades of the test cases designed from these requirements are given at the end of the table 3-22. Test cases designed from these requirements hold grades 5, 5, 3, 5 and 5.
  Test cases with grade 5 are those test cases that are in executable form and have been designed only by using the information of the given requirement. This shows that component level requirements provide clear understanding about what is to be tested and how to test. Four out of the five select component level requirements have grade 5 test cases which means that component level requirements are good for designing test cases. One test case designed from the requirement with id 18 has grade 3. As discussed earlier grade 3 level test cases cannot be made executable by using information from the given requirement only. These test cases are made complete by using information from the other requirement in the RAM requirements document.
  Almost all of the test cases designed from component level requirements have grade 5. So, it can be concluded that component level requirements are in testable form and most suitable for generating test cases.

In the above discussion we have analyzed the suitability of product, feature, function and component level requirements for designing test cases. That discussion analyzed

requirements at each level separately. Now, we discuss the RAM requirements document overall for designing test cases.

As it was discussed above, component level requirements contain good level of detail and test cases can be designed from these requirements independently. Component level requirements do not need contribution of other requirements for designing executable test cases. Function level requirements hold good information but still cannot produce executable test cases without the help of other requirements. Function level requirements provide most of the information for a test case. To make test cases complete however information from other requirements is required. Same is the case with the feature level requirements. Many of the feature level requirements provide some information to the test cases but they need contribution of the other requirements to make test cases complete. Few feature level requirements do not provide enough information that the test cases can be said as executable. These test cases cannot be completed with the help of other requirements too. Product level requirements are very abstract that they give the impression like product strategy. Executable test cases cannot be designed form these requirements, even by involving other requirements. Eight out of twenty test cases have grade 3 which shows that most of the time test cases have been designed with the help of more than one requirement. Overall thirteen out of twenty test cases are in executable from (either from single requirement or from more than one requirement).

# 4 TEST DECOMPOSITION OF REQUIREMENTS BASED TESTS

This chapter will provide current state of the research for the test decomposition. In the following sections of the chapter there is an introduction about the test decomposition, current state of research and opportunities in this area.

## 4.1 Introduction

Test decomposition means to break a test case into several test cases, in such a way that each sub-test case tests a unique aspect of a function/feature of the system. Decomposing test cases can provide a better selection of test cases while prioritizing tests for the system testing. Instead of selecting the main test for prioritization sub-tests can be selected, thus test case prioritization can be made possible at deeper level with test decomposition.

With the help of test decomposition it will be possible to study the impact of main and sub-test cases on the requirements. In this way test cases that have more impact on the system requirements can be selected for testing. Executing the tests that have more impact on the requirements can help in meeting time-to-market constraints, create better resource utilization and better return on investment.

Literature for the test case decomposition was searched from many research information databases by using many related keywords but it was hard to find information about this area. Some of the related keywords that were used to search literature for the test decomposition are:

- Test decomposition
- Test case decomposition
- Hierarchical testing
- Functional test decomposition
- Test division
- Requirements-based test case generation
- Test case generation based on formal specification
- Test case synthesis

Primary sources for the search were journals, conferences, workshops, peer reviewed articles, books, surveys and other information resources related to the software testing. It was hard to find information for the test case decomposition; however there are some opportunities where test case decomposition seems possible. These opportunities have been explored and proposed by the author. Following sections will discuss these opportunities with examples.

In previous chapter we have developed test cases from the RAM requirements and we saw that in most cases executable test cases can be designed from these requirements (either from single requirement or from more than one requirement). If it becomes possible to decompose these tests then organizations can utilize benefits like better test prioritization, meeting time-to-market constraints and better resource utilization. From the structure of the RAM it seems possible to decompose tests designed from the RAM requirements; section 4.3 discusses this opportunity with example.

## 4.2 Test Decomposition from the Scenario Based Tests

Use cases are excellent way to represent user requirements, and this representation is generally acceptable to all of the stakeholders [32]. In this section we will see how and where use cases provide opportunity for the test case decomposition. Since use

cases are being used commonly to represent requirements so scenario based testing has been selected to examine its suitability for test case decomposition.

"A scenario is an instance of the use case. It describes one specific path through the flow of events." [11] In scenario based testing test cases are generated from the scenarios that are derived from use cases. Use cases are used to exhibit functional requirements of the system.

A typical process for generating test cases from use case scenarios consists of following steps [10]:

1. For each use case identify all possible scenarios.
2. For each scenario develop a test case and identify conditions
3. Identify data values for each test case.

We will not go into details of how test cases are generated from the use case scenarios as we will only analyse the test cases generated from the scenarios (we will use tests for the use case example given below). But, some of the concepts necessary to understand are basic flows and alternative flows. Basic flow is the set of steps which are taken when every thing goes correct [11]. Alternative flows are the variations in the basic flow, these variations occur due to some exceptions or available as options to the basic flow [31]. Scenarios consist of the basic flow, alternative flows and combination of both flows. Test cases traverse these flows.

Many types of methods and tools exist which use scenarios to generate system test cases e.g. Rational RequisitePro[11] tool by IBM and SCENT method by Ryser And Glinz [9, 12]. Each approach develops use cases and their scenarios. Afterwards test cases are generated by traversing basic and alternative paths for each scenario. SCENT formalizes scenarios into state charts with all states, basic and alternative paths for a use case [9, 12].

The "Select Product" use case will be used as an example to understand the concept of test case decomposition from the scenarios. The "Select Product" use case is used to build a purchase order. This use case adds new line item and selects the product which is to be ordered. Purchase order can have many line items. Each line item orders a quantity of one product from the vender specified in the purchase order. This use case has been taken from an example used by Ross Collard in [10].

| The user | | The system | |
|---|---|---|---|
| U1 | Requests that a new line item be added to the PO. [S1.1, S1.2] | S1.1 | Adds a new line item, and assigns the next sequential line item number within the PO. [U2.1, U2.2] |
| | | or | |
| | | S1.2 | Alternatively, states that a new line cannot be added. (The PO already contains twenty-five line items.) Exit from use case. |
| U2.1 | Requests a list of products supplied by the vendor to whom the PO is addressed. [S2] | S2 | Provides the product list for this vendor. [U3.1,U3.2] |
| or | | | |
| U2.2 | Alternatively, enters the specific product number directly into the line item. [S3.1, S3.2] | | |
| U3.1 | Selects the desired product from the vendor's list. [S3.1, S3.2] | S3.1 | Verifies that the product number selected or entered is a valid one. Displays the description of the product for visual verification by the user. [U4] |
| or | | or | |
| U3.2 | Alternatively, decides there is no suitable product available from this vendor to meet the need. The user aborts the process. Exit from use case. | S3.2 | Alternatively, rejects the entered product number. [U2.1, U2.2] |
| U4 | Confirms that the right product has been selected or entered, by comparing the displayed response to what the user had intended to order. [S4.1, S4.2] | S4.1 | Verifies that the product number (either selected via the list or entered directly) does not duplicate any of the product numbers on the other line items for the same PO. Exit from use case. |
| | | or | |
| | | S4.2 | Alternatively, if the product number is a duplicate, rejects it. [U2.1, U2.2] |

Figure 4-1 User-System Interaction for Select Product use case [10]

Figure 4.1 shows the interaction between a user and the system for the "Select Product" use case. This interaction consists of the user actions and the system responses. And, Figure 4.2 shows flows of events for the "Select Product" use case. In this figure U represents the User action and S symbolizes the system response.

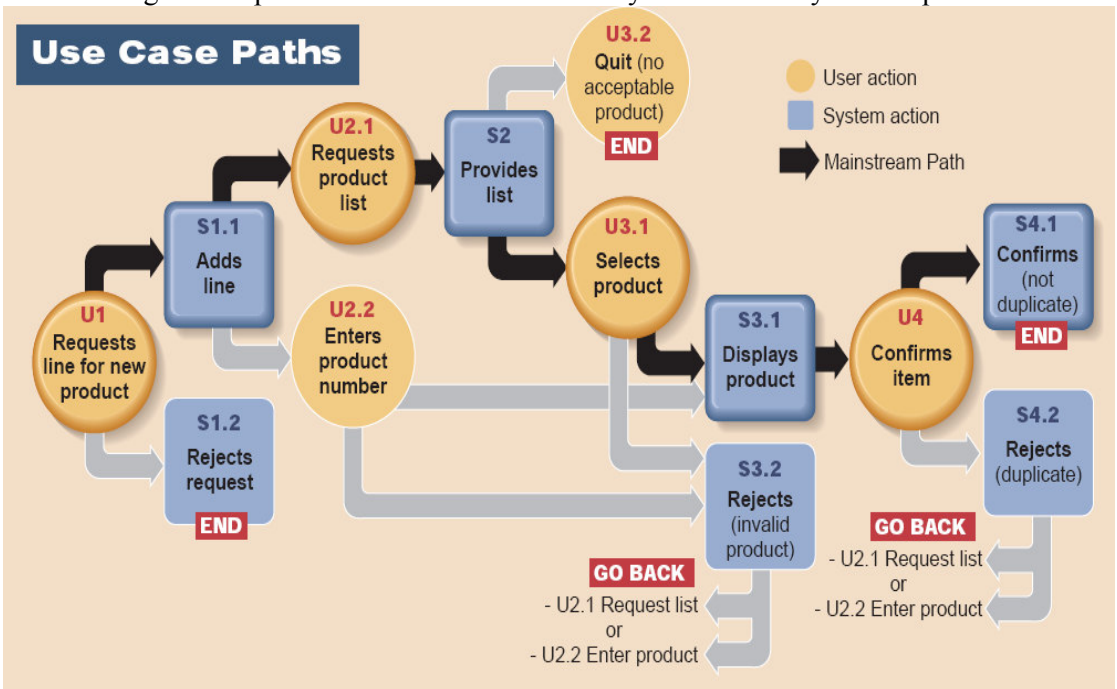


Figure 4-2 Flow of events for Select Product use case [10].

Following are the possible paths for the "Select Product" use case. P1 (Path) verifies that product number does not duplicate any other number; product number in this path has been selected from the list. P2 verifies that product number does not duplicate any other number; product number in this path has been entered manually. P3 rejects the product due to duplication; product number was selected from the list. P4 rejects the product due to duplication; product number was entered manually. In P5 user quits the use case as there is no suitable product available in the list provided by the system. P6 path shows that request cannot be fulfilled. P7 path rejects the product number; product number was selected from the list provided by the system. P8 rejects the product; product number was entered manually by the user.

P1 (Confirmed): U1 -> S1.1 -> U2.1 -> S2 -> U3.1 -> S3.1 -> U4 -> S4.1
P2 (Confirmed): U1->S1.1->U2.2->S3.1->U4->S4.1
P3 (Reject Duplicate): U1 -> S1.1 -> U2.1 -> S2 -> U3.1 -> S3.1 -> U4 -> S4.2
P4 (Reject Duplicate): U1->S1.1->U2.2->S3.1->U4->S4.2
P5 (Quit): U1->S1.1->U2.1->S2->U3.2
P6 (Reject Request): U1->S1.2
P7 (Invalid Product): U1->S1.1->U2.2->S3.2
P8 (Invalid Product): U1->S1.1->U2.1->S2->U3.1->S3.2

We can get test cases by traversing through each path. Following are some test cases designed by following the above paths.

| Test | Description | Path |
|---|---|---|
| T1 | Add a new line item for a product successfully | P1 (U1 -> S1.1 -> U2.1 -> S2 -> U3.1 -> S3.1 -> U4 -> S4.1) |
| T2 | Add a new line item for a product successfully | P2 (U1->S1.1->U2.2->S3.1->U4->S4.1) |
| T3 | rejects the duplicated product | P3 (-> S1.1 -> U2.1 -> S2 -> U3.1 -> S3.1 -> U4 -> S4.2) |
| T4 | rejects the duplicated product | P4 (U1->S1.1->U2.2->S3.1->U4->S4.2) |
| T5 | User does not find suitable product (Quit) | P5 (U1->S1.1->U2.1->S2->U3.2) |
| T6 | Reject add newline item request | P6 (U1->S1.2) |
| T7 | Invalid product provided | P7 (U1->S1.1->U2.2->S3.2) |
| T8 | Invalid product provided | P8 (U1->S1.1->U2.1->S2->U3.1->S3.2) |

Table 4-1 Tests and corresponding paths for the "Select Product" use case [10].

From Table 4.1 we can see that a certain function can be tested by many different paths. For example, adding a new line item for a product successfully can be tested by the test cases designed from two different paths P1 and P2. In both test cases (from P1 and P2) system will test the successful addition of the line item for a product, though these test cases have been generated from two different paths. Similarly, system rejects the duplicate product function can be tested by two different test cases i.e. T3 and T4. Here, T3 has been designed by traversing path P3 and T4 test case has been designed from the path P4. Same is the case when testing that user has provided invalid product number. T7 and T8 test cases have been designed from two different paths (P7 and P8 respectively) but both can test invalid behaviour of the system when user provides invalid product number.

If Ta is a test that tests the successful addition of a new line for a product then we can say that Ta consist of T1 and T2 i.e. Ta=T1+T2. And, if Tb is a test that validates that provided product number is duplicate then Tb consists of T3 and T4 i.e. Tb=T3+T4. Likewise, if Tc validates that invalid product number has been supplied then Tc comprises of T7 and T8 i.e. Tc= T7+T8.

From these examples we can see that there is an opportunity to decompose the tests. This test decomposition opportunity is based on the concept that to test a certain function test cases can be designed from many different paths. And, we can decompose the main test into tests from each path. This area needs more attention so that it can be explored that to what extent we can utilize the test case decomposition prospect from the scenario based testing.

## 4.3 Test Decomposition in the RAM Requirements Based Tests

In the previous sections we have discussed test decomposition, now we will analyze the RAM requirements support for test decomposition. It has been discussed earlier that test decomposition can provide many benefits, but more work is required to be done in this area. We saw that tests designed from use-case scenarios provide opportunity for breaking them into sub-tests. Now, we discuss how RAM support test decomposition.

The RAM model is adaptable and can be catered to meet the needs of many organizations. If test cases designed from the RAM requirements provide an opportunity for the test decomposition this will make the model more valuable. Using the RAM model will make it possible to utilize the benefits of the test case decomposition along with handling requirements abstraction.

One property of the RAM is that each requirement is comparable to the product strategies, by linking the requirement with other requirements at adjacent level(s). So each requirement is connected to the product strategies through a link with other requirements. This means that a component level requirement is connected to the product level requirements through function and feature level requirements, and the product level requirement is connected to the product strategy.

On the other hand results of the document analysis performed in this study show that product level requirements are very abstract, the only information that can be extracted from them is test purpose (goal). Test goal is theoretical description of the test, but it does not tell how to execute this test. We have a chain of requirements from component level to product level. Tests designed from these requirements test different aspects but are from the same domain of the system. We can say that test designed form the requirements in a chain can be linked with each other. One possible opportunity which RAM can provide for test decomposition is that, test designed from the product level requirement can be decomposed into the tests generated from the feature, function and component level requirements, in the same chain. This concept can be elaborated with the following example. Table 5-1 contains requirements that are linked with each other and to the product strategies. We will explore the opportunity that test cases designed from “Manage and Conduct a Course” can be decomposed into the test cases designed from “Discussion Forum”, “Add Message” and “Content of the Message” requirements.

| Organizational Strategies | |
|---|---|
| Product Strategies | |
| Product Level | Manage and Conduct a Course |
| Feature Level | Discussion Forum |
| Function Level | Add Message |
| Component Level | Content of the Message |

Table 4-2RAM Requirements

Many test cases can be designed from these requirements; some of them are given below:

T1 (from product level requirement): *test that course participants can exchange course related information.*

T2 (from feature level requirement): *Test that a discussion forum can be attached to the course.*

T3 (from function level requirement): *Test that a new message can be added the course's discussion forum.*

T4 (from component level requirement): *Test that a message contains contents when it is posted to the discussion forum.*

T1 tests the feature of the system that supports exchange of course related information. And, discussion forum is the way of exchanging course information that CMS supports. T2, T3 and T4 have been designed to test different aspects of the discussion forum. So, T1 is used to test if the system supports information exchange and T2, T3, T4 are used to test the method (discussion forum) of exchanging the information. To test that course participants can exchange information means testing the discussion forum (a way of exchanging the information). So, we can say that testing through tests T2, T3 and T4 can also full fill the goal of testing through T1 i.e. T1=T2+T3+T4.

The structure of the RAM is helpful for creating test decomposition. With such test decomposition it will be possible to select test cases at deeper level. Instead of selecting the main tests, sub-tests will be available for selection. Better test prioritization can save time, hence helps in meeting the time-to-market constraints. In market driven engineering, different versions of the product are send into the market at short times. Test prioritization using sub-tests can help in testing only highly prioritized requirements. This will help in delivering the short time versions with right functionality. The RAM helps in managing requirements abstraction; now with this opportunity it seems possible to have better test prioritization. These factors can enhance its significance and suitability for various organizations.

This is the opportunity for test decomposition that has been explored during the study. More work is required to better understand and validate it. Only one example has been quoted here, more examples are required to authenticate and generalize this concept. Another concept that is closely related to the test decomposition is measurement of the proportion sub-tests have to their main test. With proportion measurement it will be possible to identify how much part each sub-test contains to the main test. With this measurement it will be possible to calculate the level of fulfilment each test have in term of satisfying a requirement. A future study is required on the test decomposition opportunity provided by the RAM.

# 5 DISCUSSION

This chapter discusses results of the above performed document analysis in detail, and describes the threats that may affect the results of this study.

## 5.1 RAM Document Analysis

Document analysis of the CMS RAM requirements document has been performed in this study. Results of the test cases designed from product, feature, function and component level requirements are given in the table 3-22. Here we analyze and discuss these results from various aspects.

### 5.1.1 Comparison of the Test Cases' Results

We start by comparing the results of the test cases of each level requirements with the test cases of other level RAM requirements. The comparison of the results will be discussed in perspective of the criteria of placing a requirement at product, feature, function or component level. The criteria have been defined by the Gorschek and Wohlin [18]. Comparison of the test cases of each level requirements with test cases of other level requirements will help in understanding the relative testability of the requirements. In addition, this comparison will validate the criteria of placing requirements at four levels in the RAM. Before starting the discussion let us see the criterion for placing a requirement on a specific level in the RAM. Each requirement is analyzed against a set of questions. The requirement that answers yes to a question is placed at the level associated with that question. Set of questions that are asked for product, feature, function and component levels are:

- **Product Level:**
  Is the requirement abstract enough to be comparable to the product strategies?

- **Feature Level:**
  Doses the requirement describe what the system should include? and/or
  Does the requirement describe a feature that should be supported?

- **Function Level:**
  Is the requirement functional or does it describes testable characteristics that the product should have? and/or
  Does the requirement describe functionality that is to be performed by the user?

- **Component Level:**
  Does the requirement consist of a specific suggestion of HOW something should be done/solved?

Now we compare the results of the test cases for each level requirements with the results of the other three level requirements.

#### 5.1.1.1 Component Level Test Cases

We start by comparing the results of the test cases designed from the component level requirements with those of function, feature and product level requirements. Most of the test cases designed from the component level requirements have grade 5 as shown in the Table 3-22. For function level requirements test cases we can see that most of the test cases have grade 3. So, comparing results of the component level requirements with function level requirements we can see that the component level requirements are more testable than function level requirements. Results show that component level requirements have enough information that detailed test cases can be

designed solely from them. But, it is not possible to design detailed test cases exclusively from the information of the given functional level requirement. Unlike component level requirements, function level requirements need information from other requirements too, in order to make test cases more detailed. From the above listed questions we can see that a requirement is placed on the component level if it provides information about how a function can be performed or solved. And, function level requirement only describes the functionality of the system. So, rightly a requirement that provides a solution to a function is more testable than function level requirement that only describe the functionality. Results of the grades of the test cases designed from these level test cases also exhibit this difference. So, results verify that function and component level requirements contain distinct information, and validate the criteria for placing requirements at function and component levels.

Grades of the test cases designed from the feature level requirements suggest that like function level requirements, executable test cases cannot be designed exclusively from the feature level requirements. They need information from the other requirements in order to make test cases executable. A requirement is placed at feature level if it contains information about the features of the system. Features are the characteristics, which a system can have. So a feature level requirement contains information about the feature but it does not tell how this feature works and how this feature will be implemented. Only with the description of the features of the system it is not possible to produce executable test cases, information from the other requirements may be required to make them executable. This is what results of this study have proved, that most of the time it is not possible to design executable test cases solely from the description of the given feature level requirement. Other requirements may contribute to make these test cases more detailed, hence executable. Comparing the questions for component level and feature level requirements, we can find that feature level requirements only contain description of the features of the system whereas component level requirements contain information about the implementation (solution) of the functions of the system. And, comparing the results of the test cases designed from the component level requirements with the feature level requirements, we can see that testability characteristic of the feature level requirements is less than those of component level requirements. So, the results prove and strengthen the argument made by the RAM that component level requirements are more testable than feature level requirements.

While making comparison of the results of the test cases designed from the component level requirements with the product level requirements, we can see that executable test cases can be produced from component level requirements. It is not possible to design executable test cases from the product level requirements. So, requirements at lowest level of the RAM model are more suitable for designing executable test cases whereas requirements at top level are not suitable for this purpose. A requirement is placed at the product level if it is abstract like product strategies. And, placing a requirement on the component level requires that requirement contains information about how something will be solved. Questions that put requirements on product and component levels show that both level requirements contain very distinct information with respect to each other. The results of the document analysis prove that the level and kind of information component level requirements contain is different and better than that of product level requirements.

#### 5.1.1.2 Function Level Test Cases

Now we compare the results of the test cases designed from the function level requirements with the results of the test cases designed from the other level requirements. As given in the table 3-22, most of the test cases designed from the function level requirements have grade 3. This mean it is not possible to design executable test cases only from the information of the given requirements. But, contribution of the other requirements in the RAM document is required to make them

executable. We have already compared the results for the feature level requirements with component level requirements.

Looking at the results of the document analysis given in table 2.23, we can see that most of the test cases designed from the function and the feature level requirements have grade 3. In total 7 out of 10 (function and feature level) test cases have grade 3. Grade 3 means, the given requirement could not produce executable test cases. Contribution from the other requirements was required to make these test cases executable. 4 out of 5 test cases for the function level requirements and 3 out of 5 test cases for the feature level requirements have grade 3. This shows that most of the requirements at the function and the feature level are of same abstraction. But, the RAM suggests that there is a clear difference between function and feature level requirements. In RAM, function level requirements are more testable than feature level requirements. But, the results here show that most (7 out of 10) of the function and the feature level requirements have same testability level. Requirements that produced test cases with the same grade should either belong to the function level or the feature level. A possible issue that might have created this problem is that the requirements have not been placed at the appropriate levels. The criteria for placing the requirements on the function and the feature level are set of questions, given above.

Analyzing requirements in the document against their corresponding questions for function and feature levels we can identity some problems. For example, requirement Course News with id=7 (ids are given in table 3-1) is a feature level requirement and requirement Add Course Link with id=13 is a function level requirement. Course News requirement states that “It shall be possible to attach news items to a course” and Add Course Link requirement states that “It shall be possible to add links to the list of links attached to a course”.

It is clear from the description of the Course News requirement that it does not describe any functionally. Description of this requirement suggests that it is a feature of the CMS system and this requirement has already been placed rightly on the feature level. Description of the Add Course Link is almost same as of the Course News. Like Course News, Add Course Link does not describe any functionality of the system. Description of the requirement suggests that it is a feature of the CMS.

From this example we can say that Course News and Add Course Link requirements should be at the same level (at the feature level according to the criteria given in the RAM). Or, if these requirements have been analyzed from any other perspective (that suggests that Course News belongs to feature and Add Course News belongs to function level) then questions should clarify that perspective. The questions should be capable enough that they can distinguish requirements with different testability. Improving the set of questions can help in placing the requirements at right level. Five requirements from each level were used to design test cases. To understand and describe the above discussed issue more accurately, more number of requirements can be used to design test cases in future studies. Test cases designed from more requirements will help us in understanding the problem with the function and the feature level requirements in better way.

Let us now discuss how the results of the document analysis differ for the function level requirements and the product level requirements. As we have discussed earlier that function level requirements cannot produce executable test cases from their own information but need other requirements to make test cases executable. Results of the test cases designed from the product level requirements show that it is not possible to design executable test cases from them. Product level requirements provide very limited information for test cases, and it is not possible to design executable test cases with the help of other requirements too. From the questions which are used to place the requirements on the function and product level, difference can be observed. Product level requirements are similar to the product strategies whereas function level requirements describe the functionality of the product. Requirements similar to the product strategies contain very abstract information which cannot help in designing

detailed test cases. On the other hand function level requirements describe the system functionality so it is possible to design executable test cases (with the help of other requirements in the RAM document) from them. Results of this study strengthen the difference between function and product level requirements given by the RAM.

#### 5.1.1.3 Feature Level Test Cases

Now we will see how results of the feature level requirements differ from the other level requirements. Earlier in this section we have compared results of the feature level requirements with the function and the component level requirements. So, here we will compare results of the feature level requirements with the product level requirements. Results given in the table 3-22 show, it is not possible to create executable test cases solely from the information of the given feature level requirement. But contribution from the other requirements can make these test cases executable. Product level requirements cannot produce executable test cases too, neither from the help of the other requirements in the RAM document. Looking at the questions used to identify feature and product level requirements, we can see that product level requirements describe the business goals, whereas feature level requirements describe the features of the system. Designing test cases from the business goals is difficult than from the description of the features. Results of the document analysis performed in this study also prove this difference in the feature and the product level requirements.

#### 5.1.1.4 Product Level Test Cases

In the previous sections we have already compared the results of the component, function and feature level test cases with the product level test cases. So, we will not replicate that discussion here in this section.

#### 5.1.1.5 Conclusion of the Comparison

In above discussion we have compared the results of product, feature, function and component level requirements with each other. Conclusion of this comparison has two perspectives.

Firstly, it strengthens the general perception about the RAM requirements. Description of the RAM says that product level requirements are highly abstract, almost same as business strategies and goals. Features level requirements contain description of the features of the system. Function level requirements describe the functionality of the system. Component level requirements provide information about how to solve a particular problem. From this short description of each level we can see that a general perception is that component level requirements are better testable. And, function level requirements contain good level of information than feature and product level requirements, but are less testable than component level requirements. Feature level requirements are not testable like function and component level requirements but have better level of information than product level requirements. Product level requirements are not in testable form because they are highly abstract. Results of the study show that as we move upward on the RAM levels, testability of the requirements decreases. So, results of the document analysis performed in this study reinforce above discussed perception about the RAM requirements at different levels.

Secondly, the above discussion validates the criteria of placing the requirements at product, feature, function and component level in the RAM. During discussions we saw that the results for the product level requirements show that the criterion (set of questions) for placing requirements on the product level is correct. We also found that the results of this document analysis identify a problem in the criteria for placing the requirements on the feature and the function levels. It was observed that both level requirements produce test cases with same grade. The results of this study also verify the criteria for placing requirements on the component level in the RAM.

Major tasks of this document analysis includes selection of the test case template, selection of the requirements, designing test cases and grading test cases based on a chosen criterion. Now we discuss some of these activities and their effect on the results of the document analysis.

Test case template selected for this document analysis is based on the IEEE 829-1998 standard. Since this template is of theoretical nature so some alterations were made after analyzing the different templates being used in the industry. Many organizations use their own test case template, for these organizations results of the document analysis performed here may not be of great worth. But, since test case template used here is bases on IEEE 829-1998 standard, organizations which are using test case templates based on this standard can utilize results of this document analysis.

Twenty requirements from the CMS RAM requirements document have been selected to perform the document analysis. Twenty requirements have been selected in such a way that five requirements from each level of the RAM. CMS requirements document consist of 183 requirements. During designing test cases from a requirement, every other requirement (182 requirements) in the RAM document was analyzed to extract more information for the test cases. It is very time consuming and laborious task, and due to the time limitation only 20 requirements were selected to perform this document analysis. Use of more than 20 requirements can enhance the authenticity of the results, and can generalize the results in a better way. Use of more requirements can help in getting the right picture of the issue (produce test cases with same grade) found with the feature and the function level requirements. A future document analysis of the RAM document can be made with more requirements.

Test cases designed from the product, feature, function and component level requirements were graded on a 5-1 scale. Grade is based on the level of detail extracted for a test case from the requirement. Details about the grading scale are given in the section 3.1.3. This grading scale is based on the information extracted from the requirements only; tester (test designer, author in this case) was not allowed to add information at its own. Adding information from the past experience of the tester may affect the results of the study. Since it is possible that tester may complete the incomplete test cases by adding information from its past experience with similar tests. Since, designing of the test cases is solely based on the information from the requirements in the RAM document so grading is quite valid.

## 5.2 Validity Threats

This section will describe the possible threats that may affect the results of the document analysis performed in this study.

### 5.2.1 Placement of Requirements at Different Levels in the RAM

The RAM provides four requirements abstraction levels (product, feature, function and component), each requirement that comes into the requirements engineering process is placed at its suitable level. Each requirement is placed at a suitable level by analyzing the requirement against a set of questions. Placement of requirements is done manually by the requirements engineers. Interpretation of the requirements against questions may vary for different requirements engineers. If the requirements have not been placed at appropriate level this may affect the end results.

### 5.2.2 Work-Up Activities in the RAM

After placing a requirement at a specific level, work-up activities are performed on the requirement. These activities involve abstracting and/or breaking down the requirement, depending upon the position of the requirement [18]. Any deficiency in creating appropriate abstraction in the requirements or breaking the requirements can result into ambiguous requirements. Much depends upon the expertise of the persons

involved in these activities. If the abstraction in the requirements has not been properly created or requirements have not been broken correctly, this may affect the results of this study.

### 5.2.3 Test Designing Skills

Test cases have been designed by the author during the document analysis of the RAM requirements document. In most of the cases designing effective test cases depends upon the experience and skills of the test designer. Skills of the author in this case may have affected the results of the study. This threat to the validity of the results is not very extensive, as it was tried to extract information for the elements of the test case template directly from the given description, rationale and restrictions on the requirements. The tester (author) was not allowed to add information from his own experience.

### 5.2.4 Number of Selected Requirements

Course Management Requirements Document has 183 requirements in total. Due to the time limitations only 20 requirements (5 from each level) have been selected. Selecting more requirements may produce better results. Selecting only 20 requirements can affect the generalization of the results for all requirements in the requirements document.

### 5.2.5 Executable Tests

After having an extensive exploration of the term executable tests it was found that there is no clear definition and criteria for stating a test as executable. It was assumed that a test case will be executable if it has good level of detail for all fields in the test case template. With having no concrete definition of the executable tests may have affected the results.

# 6 CONCLUSIONS AND FUTURE WORK

This chapter summarizes the study. It includes some possible future works that were introduced during this study. At the end of the chapter there are conclusions to the study.

## 6.1 Conclusions

The purpose of this thesis was to analyze the requirements at product, feature, function and component level to evaluate their suitability for supporting the creation of high-level system tests. Another task was to understand and explore the test decomposition area of research, and to analyze how tests designed from the RAM requirements support test decomposition.

Requirements Abstraction Model (RAM) provides a structure for organizing the requirements with different abstraction in the market-driven requirements engineering. Requirements in the RAM are placed at four different levels of abstraction i.e. product, feature, function and component levels. With its flexible nature it is possible to cater the RAM for different organizations. Since, requirements are a valuable source for designing high-level system tests, and in current development approaches designing tests from requirements start at early phases. So, organizations that are willing to adopt the RAM need to know how helpful the RAM requirements are for designing system and acceptance level tests.

A document analysis of the RAM requirements document has been performed to analyze how much information for a test can be extracted from each requirement. Twenty requirements, five from each level (product, feature, function and component) have been selected for the document analysis. A test case was declared to be in executable form if it holds good level of detail for its fields in the test case template. Based on a chosen criterion (level of detail extracted from a requirement) test cases were graded on a 5-1 scale.

Results of the document analysis show that as we move upward from the component level to the product level requirements in the RAM, chances of creating executable tests from the requirements decreases.

Component level requirements are in better testable form, and can help in creating executable tests at their own. Function level requirements hold good information but still cannot produce executable test cases without the help of other requirements in the RAM document. Like function level, feature level requirements cannot produce executable from their own information, but need other requirements to generate executable tests. Product level requirements are highly abstract. Executable tests cannot be generated from the product level requirements, even with the contribution of the other requirements. Eight out of twenty test cases have grade 3 which shows that most of the time test cases have been designed with the contribution of more than one requirement. Overall thirteen out of twenty test cases are in executable from (either from single requirement or from more than one requirement). Results show that most of the times, test cases can be designed from the RAM requirements, either from the single requirement or with the contribution from more than one requirement.

A general understanding about the RAM is that requirements abstraction increases as we move upward from the component level to product level. Results of the study strengthen this understanding, since results prove that it becomes harder to design executable test cases from the requirements as we move upward from the component level to product level.

With test decomposition it is possible to have better test prioritization, hence utilizing benefits like meeting time-to-market, better resource utilization etc. In order to enhance understanding with it, more work is required in this area. Test cases designed from the use case scenarios provide opportunity for creating test

decomposition. Moreover, tests designed from the RAM requirements also provide opportunity for creating test decomposition. It is possible to divide tests designed from the product level requirement into tests designed from the feature, function and component level requirements in the same chain.

## 6.2 Future Work

As it has been discussed earlier in chapter 4, very limited work has been done on test case decomposition. A systematic review of the literature in this area can enhance the understanding about the current state of the art research for test case decomposition.

In chapter 5, it was found that tests developed from the RAM requirements are suitable for creating test decomposition. The RAM requirements provide opportunity for dividable tests; this concept has been discussed earlier with example. Only one example has been used in this study. An exclusive study can be performed on this topic with more examples. The study will validate the opportunity discussed in chapter 5.

In this master thesis, a document analysis for analyzing the suitability of the RAM requirements for high-level system testing has been performed on a single RAM requirements document. As a future study it will be of great interest to perform document analysis of two (or more) different RAM requirements document. This will help in identifying any variation in the results of the document analysis of the RAM documents.

# REFERENCES


[1] Elfriede Dustin, *Effective Software Testing: 50 Specific Ways to Improve Your Testing.* Addison Wesley Professional, 2002.

[2] Ron Patton, *Software Testing, Second Edition.* Sams, 2005.

[3] G. Kotonya and I. Sommerville, *Requirements Engineering - Processes and Techniques.* John Wiley & Sons Ltd, 1998.

[4] Mogyorodi G. "Requirements-based testing: an overview," in *39th International Conference and Exhibition on Technology of Object-oriented Languages and Systems (TOOLS-39)*, July 2001, pp. 286-295.

[5] Tahat L.H. Vaysburg B. Korel B. and Bader A.J. "Requirement-based automated black-box test generation," in *25th Annual International Computer Software and Applications Conference (COMPSAC'01)*, October 2001, pp. 489-495.

[6] Hsia, P. Davis, A.M. Kung, D.C. "Status report: requirements engineering," *IEEE Software*, Vol. 10, pp. 75 – 79, November 1993.

[7] Muthu Ramachandran, "Requirements-driven software test: a process-oriented approach," *SIGSOFT Software Engineering Notes*, Vol. 21, pp. 66-70, July 1996.

[8] Johannes Ryser, Stefan Berner and Martin Glinz, "On the State of the Art in Requirements-based Validation and Test of Software," Universitat Zurich, Institut fur Informatik, Zurich, Berichte des Instituts fur Informatik 98, 1998.

[9] Johannes Ryser and Martin Glinz, "SCENT: A Method Employing Scenarios to Systematically Derive Test Cases for System Test," Universit.t Z.rich, Institut f.r Informatik, Z.rich, Berichte des Instituts f.r Informatik 2000/2003, 2000

[10] Ross Collard, "Test Design: Developing test cases from use cases," [online]. Available:http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=MAGAZINE&ObjectId=5081&tth=DYN&tt=siteemail&iDyn=2. [Accessed: July 24, 2007].

[11] Peter Zielczynski, "Traceability from Use Cases to Test Cases", [online]. Available: http://www.ibm.com/developerworks/rational/library/04/r-3217/. [Accessed: May 29, 2007].

[12] Johannes Ryser and Martin Glinz, "A Scenario-Based Approach to Validating and Testing Software Systems Using State charts", In *Proceedings of the Twelfth International Conference on Software & Systems Engineering and their Applications (ICSSEA '99)*, December 1999.

[13] IEEE, "IEEE-STD 610.12-1990: IEEE Standard Glossary of Software Engineering Terminology," IEEE, [online]. Available:http://ieeexplore.ieee.org/servlet/opac?punumber=2238. [Accessed: May 25, 2007].

[14] Aybüke Aurum and Claes Wohlin (Eds), *Engineering and Managing Software Requirements*. Springer, 2005.

[15] Hickey Ann M. Davis Alan M. and Kaiser Denali, “Requirements Elicitation Techniques: Analyzing the Gap between Technology Availability and Technology Use,” *Comparative Technology Transfer and Society,* Vol. 1, pp. 279-302, 2003.

[16] IEEE, “IEEE-STD 830-1993: IEEE Recommended Practice for Software Requirements Specifications,” IEEE, [online]. Available:http://ieeexplore.ieee.org/servlet/opac?punumber=3114. [Accessed: June 14], 2007.

[17] Elizabeth Hull, Kenneth Jackson and Jeremy Dick, *Requirements Engineering.* Springer, 2004.

[18] Tony Gorschek and Claes Wohlin, “Requirements Abstraction Model,” *Requirements Engineering*, Vol. 11, pp. 79-101, 2006.

[19] Gorschek T. Svahnberg M. Borg A. Loconsole A. Borstler J. Sandahl K. and Eriksson, M. “A controlled empirical evaluation of a requirements abstraction model,” *Information and Software Technology*, Vol. 49, pp. 790-80, 2007.

[20] Tony Gorschek, Claes Wohlin and Per Garre, “A Model for Technology Transfer in Practice,” *IEEE Software*, Vol. 23, pp. 88-96, 2006.

[21] Tony Gorschek, “Requirements Engineering Supporting Technical Product Management**,”** PhD dissertation, Blekinge Institute of Technology, Karlskrona, Sweden, 2006.

[22] Watkins, John, *Testing IT: An Off-the-Shelf Software Testing Handbook.* Cambridge University Press, 2001.

[23] Myers, Glenford J. *Art of Software Testing.* John Wiley & Sons Inc. 2004.

[24] S. R. Rakitin. *Software Verification and Validation for Practitioners and Managers*. Artech House Inc. 2001.

[25] Ilene Burnstein, *Practical Software Testing A Process-Oriented Approach.* Springer London, 2003.

[26] Pyhajarvi M. Rautiainen K. and Itkonen, J. “Increasing understanding of the modern testing perspective in software development projects**,”** in *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, January 2003.

[27] Wikipedia, “V-Model software development”, [online]. Available: http://en.wikipedia.org/wiki/V-Model_software_developmen. [Accessed: June 6, 2007].

[28] Wikipedia, “List of unit testing frameworks”, [online]. Available: http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks. [Accessed: June 9, 2007].

[29] Lee Copeland, *A Practitioner's Guide to Software Test Design.* Artech House, Inc. 2003.

[30] Rick Craig, “The Value of Requirements-Based Testing”, [online]. Available: http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectId=6455. [Accessed: June 20, 2007].

[31] Jim Heumann, “Generating test cases from Use Cases”, [online]. Available: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jun01/GeneratingTestCasesFromUseCasesJune01.pdf. [Accessed: July 24, 2007].

[32] Thomas Behrens, “Capturing business requirements using use cases”, [online]. Available:http://www.ibm.com/developerworks/rational/library/dec04/behrens/. [Accessed: August 1, 2007].

[33] IEEE, “IEEE-STD 829-1998: IEEE standard for software test documentation,” IEEE, [online]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=5976. [Accessed: August 08, 2007]

[34] Dr. Mikael Svahnberg, Course Management System RAM Requirements Document.

[35] Offutt A.J. Yiwei Xiong and Shaoying Liu, “Criteria for generating specification-based tests,” in *5th International Conference on Engineering of Complex Computer Systems (ICECCS '99),* Oct. 1999, pp. 119-129.

[36] Karlsson, L., Dahlstedt, Å.G., Natt och Dag, J., Regnell, B. and Persson, A, ” Requirements engineering challenges in market-driven software development - An interview study with practitioners”, I*nformation and Software Technology,* Vol.49, pp. 588-604, 2007.

[37] Richard Bender, “Requirements Based Testing Process Overview”, [online]. Available: http://www.benderrbt.com/Bender-Requirements Based Testing Process Overview.pdf. [Accessed: May 15, 2007].

[38] Aynur Abdurazik, Paul Ammann, Wei Ding and Jeff Offutt, “Evaluation of Three Specification-based Testing Criteria” in *Sixth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS '00)*, September 2000, pp. 179-187.

[39] Tony Gorschek, Per Garre Stig B. M. Larsson and Claes Wohlin, “Industry evaluation of the Requirements Abstraction Model”, *Requirements Engineering*, Vol. 12, pp. 163-190, 2007.

# APPENDIX 1

This appendix contains details about the Course Management System (CMS). CMS requirements document has been used for the document analysis performed in this study. Since this document contains 183 requirements, it is difficult to provide whole document here. So appendix contains only general description about the CMS.

---

Software Requirements Specification
Course Management System

Version 1.03
2005-12-08

## 1 Introduction

This document describes the requirements for Course Management System (CMS). CMS is an intranet solution for course management. The purpose of the product is to provide teachers and students with information regarding courses, such as news, schedule, necessary files for the course, discussion forums, and management of course participants.

### *1.1 Purpose of the Requirements Document*

This document, the Software Requirements Specification (SRS) describes the requirements on CMS. The intended audience for this document is customers, product managers, project managers, developers, and testers.

Customers will use the document to get an overview of what has been agreed upon. The SRS will help communicate that the development organization and the customers have a shared understanding of the system requirements.

Product managers and project managers will use the document to plan development projects and product releases.

Developers and Testers will use the document to understand what should be supported by the product. Developers will construct the product based on the SRS, and the testers will write test cases and test the product based on the SRS.

### *1.2 Scope of the Product*

The CMS is a web-based product that consists of a server, a database, and multiple web clients. These are referred to as CMS Server, CMS database, Web Client.

### *1.3 Definitions, Acronyms and Abbreviations*

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

### *1.4 References*

RFC 2119 http://www.faqs.org/rfcs/rfc2119.html

### *1.5 Overview of the Remainder of the Document*

The remainder of this document is organized as follows. Section 2 presents an overview of the product. Section 3 presents specific requirements on the product.

Page 3 of 27

# 2 General Description

## 2.1 Product perspective

The CMS product is to be used at Universities, where there is a need to provide information about courses, such as schedules, course news, and distribute documents and other files necessary to conduct a course. There product shall also support management of course participants. The product is a stand-alone, independent system.

## 2.2 Product Functions

The product is intended to provide a personalized user experience, so that users recognize themselves while working in the system, and so that information relevant for each user is presented. It shall be easy for the user to find relevant information in a quick and easy overview. The product shall also provide functionality that supports a teacher (course manager) to conduct a course, manage the course, and supports the participants in following course progress, course news, and accessing information related to the course as well as exchanging course related information. It shall also support management of course participators. Functions that the product shall include are: provide information about courses, provide news in a course, provide an archive of documents and other files used in the course, provide a course-specific calendar, provide a user-specific calendar, provide an archive of useful web links for a course, provide a course-specific discussion forum, and maintain personal profiles on all system users.

The product shall be usable in terms of accessibility, understandability and performance. All access to the product must take place via the system's own user interface, i.e. access from third party products is not allowed. This is to control the look and feel of the product, and to avoid security issues (i.e. prevent unauthorized use) and compatibility problems. Course participants shall be able to access the product from their home computers, which is an uncontrolled environment that may consist of any computer hardware and operating system. The product shall thus support common computer and operating system platforms available on the Swedish market.

## 2.3 User Characteristics

**Course Manager**. Course managers are teachers at the university. They have varying degrees of computer practice, but most of them use computers in their everyday work. The CMS product will be extensively used by the course managers.

**Course Tutor**. Course tutors are junior teachers or even senior students. They have considerable computer practice, often more than the course manager. The course tutors will have some experience of using the CMS product as course participants, but are in general novices on the product.

**Course Participants**. Course participants are students of all ages and backgrounds. Some have considerable computer experience, and some have only very basic computer skills. Much of the education will be assisted by CMS, so over time the course participators are

expected to become skilled in using the product – unless they only take an individual course.

**System Administrator**. The system administrator has considerable computer experience. The system administrator is expected to maintain the system over a long period of time, and will thus have considerable experience of those parts of the system that he or she comes into contact with.

## 2.4 General Constraints

The product is expected to support a large number of users accessing the system many times per day. A rough estimate yields 1 million accesses per day. Most of these accesses will occur during office hours and especially directly before and after lectures (8am, 10am, 12am, 1pm, 3pm, and 7pm).