Reflective Report

Yilei Liu
9206012669
yili14@student.bth.se

1. Article Selection:

I choose paper "Requirements Abstraction Model" written by Gorschek & Wohlin[1]. This paper detailed introduced what is RAM and the process of it. RAM is a tool to offer product manager handle and work with requirements on multiple levels of abstraction in a continuous product-center requirement engineering effort. There are four advantages of RAM. All the requirements can be compared to the product strategies to ensure it does not violate the goals set by the management. All the requirements are broken down to an abstract level that is goof for starting a development project. Work-up of requirements means that they are in the same abstract level and they can be compared and set against one another. All the requirements can get a richer understanding through the abstract process. RAM use the work-up rules to abstracts and breaks down requirements to several levels and reflecting the need for a development organization. Product Level requirements can be compared to product strategies and attempt to dismiss out-of-scope requirements an early phase. This model provides a detailed analysis of requirements and ensures the risk will not creeping after development effort was planned and started. RAM can be tailored to different organization and products.

I also choose paper "A case study evaluation of the guideline-supported QUPER model for elicitation of quality requirements" written by Svensson and Regnell[2]. This paper is a case study of the QUPER model. It presents the detailed practical guidelines of how to use QUPER and added the introduction of how to incorporate cost dependencies between QR. And then, they use the detailed guidelines to evaluate QUPER's applicability with real QR. It also discussed how to define the breakpoints and barriers and the different important level of the three levels. The QUPER model has three views that can help the companies understand better of the necessary requirements of their products. The benefit view: there are three breakpoints that show the main changes in the benefit level. The utility breakpoint marks the border of quality from useless to useful. The differentiation breakpoint marks the level of the quality is in a competitive position. The saturation breakpoint marks the product is excessive that higher quality level has no practical impact on benefit. The cost view includes cost barriers that present the non-liner relationship between quality and cost. A barrier occurs when the quality cost shift from a low level to a high level. The roadmap view combines the first two views by putting the breakpoints and carries on the same scale. The goal of this view is to compare the quality of our product and the competitor's and make some targets for the future releases. The challenge in the definition is difficult to define and specify the value for the differentiation and saturation breakpoints. As the purpose of developing QUPER is to support high-level decision-making in the quality requirements release planning while helping market-driven software products to plan their product release more effectively. Applications QUPER can help developers deliver competitive and high quality products to the market as soon as possible. QUPER is the only way to solve QR quality and cost constraints.

2. Implementation Plan:

Method 1——RAM:

RAM has three steps:

1. Specify(elicit)

The goal of this step is to get an overview of the requirements. There are four attributes to describe the requirements(attributes 1-4):

- Description: Describes the essence of the requirement.
- Reason/Benefit/Rationale: Describes why chose this requirement and the benefits for it.
- Restrictions/Risks: This attribute describes the risks that may exist.
- Title: The name of the requirement. Reflecting the essence of requirement.

2. Place(evaluate)

In this step, I will analysis which level a requirement is on. RAM consists four abstraction levels:

- Product Level(goal): this is the most abstract level. Requirements in this level are goal-like in nature.
- Feature Level(features): requirements on this level are the features that the product supports.
- Function Level(functions/actions): this is a repository for functional requirements and for non-functional requirements.
- Component Level(details consists of): this is the last level of abstraction.

There are two rules for the abstraction. No requirements may exist without having a connection to the Product Level and all the requirements have to be broken down to Function Level.

3. Abstraction(work-up)

The third step is to abstract and breakdown of a requirement. The work-up process also includes creating new requirements. Specifying additional requirements on adjacent abstraction levels until the work-up rules are satisfied. In this process, attributes 1-4 are specified for the new requirements.

Method 2——QUPER:

This section showed the practical guidelines of QUPER.

1) Identify candidate QR which considered relevant features, market segment, competitor and hardware platform capability.

2) For each selected QR, define a scale and a measurement unit to express the level of quality of QR.

3) Identify reference levels for each QR based on actual products.

4) Define market expectations to elicit quality breakpoints.

5) Estimate the cost in terms of the cost barriers.

6) Propose candidate requirements, discuss and decide actual requirements for the coming release, create roadmaps.

7) If cost dependencies between QR are considered important for cost estimations, identify which module needs to be changed if that QR will be improved beyond the “next” breakpoint.

The QUPER model has three views that can help the companies understand better of the necessary requirements of their products.

The benefit view: there are three breakpoints that show the main changes in the benefit level. The utility breakpoint marks the border of quality from useless to useful. The differentiation breakpoint marks the level of the quality is in a competitive position. The saturation breakpoint marks the product is excessive that higher quality level has no practical impact on benefit. The cost view includes cost barriers that present the non-liner relationship between quality and cost. A barrier occurs when the quality cost shift from a low level to a high level. The roadmap view combines the first two views by putting the breakpoints and carries on the same scale. The goal of this view is to compare the quality of our product and the competitor's and make some targets for the future releases. The challenge in the definition is difficult to define and specify the value for the differentiation and saturation breakpoints.
As mentioned above, applying QUPER consists of seven steps. To in-depth understanding QUPER, I plan to apply QUPER on two quality requirements of a student educational systems. This system is for students and teachers. The target market now is Sweden schools that have international students. I will implement all seven steps by specifying the current level of competitors and determining the cost for each quality requirement.

3. Proof of Concept and Execution:

Method 1—RAM:

For this method, I have selected the requirements form Github, the course management system. As implement thousands of requirements is very difficult for me. To shown I have clearly understood this method, I will list 5 requirements as the example. The execution process is shown as following.

- Step one: Specify.

In this step, four attributes need to be specified: description, reason/benefit/ rational, and restrictions/risks and title. the following are the attributes for the above three requirements.

Requirement 1:

Description: All access to the system must take place via the systems own user interface, i.e. access from third-party products is not allowed.

Reason/Benefit/ Rational: This can control the look and user feel. The benefits are to avoid security issues and compatibility problems.

Restrictions/Risks: Access from the third party is a risk. More functions need to be fixed to reduce this risk.

Title: Restricted User Interface

Requirement 2:

Description: As a user, I want to have the personalized view in the system so that I am only presented with information that is relevant to me. Users in the system shall have a Personal Profile. Allow other users to find contact information.

Reason/Benefit/ Rational: This can facilitate the interaction between users.

Restrictions/Risks: Should provide a function to allow users upload their contact information.

Title: Personalize Views

Requirement 3:

Description: We are currently focusing on the Swedish market, but will extend this to the European market in the future. Long-term, we may wish to target other continents as well. The user interface language is Swedish or English. This will be extended to other languages.

Reason/Benefit/ Rational: The target market is Swedish. The students and teachers are almost understand Swedish and English.

Restrictions/Risks: Some of the users may don't know Swedish and English. Translate to another language may result in other problems.

Title: Market and User Interface Language

Requirement 4:

Description: As a user, I should be able to log into the system.

Reason/Benefit/ Rational: To distinguish users and keep user security.

Restrictions/Risks: It is difficult to handle shared meeting or sessions.

Title: User login

Requirement 5:

Description: As a system manager I want the product to prevent unauthorized use, so that I can control who is using the system.

Reason/Benefit/ Rational: This can keep system safety and protect user information.

Restrictions/Risks: User can not modify the information when without unauthorized.

Title: Secure Product

- Step two: Place.

RAM has four abstraction levels: Product level, Feature level, Function level, and component level. This step is to analysis which level the requirements are on and place it. The following are the descriptions for each level.

Product level: this is the most abstract level. Requirements in this level are goal-like in nature.

Feature level: requirements on this level are the features that the product supports.

Function level: this is a repository for functional requirements and for non-functional requirements.

Component level: this is the last level of abstraction.

To place the requirements into the right level, there are some steps and questions need to be done to help analysis the level the requirements are on. RQ1: Restricted User Interface. RQ2: Personalize Views. RQ3: Market and User Interface Language. RQ4: User login. RQ5: Secure Product.

1. Is the requirement function or not?

RQ2 is a function for personalizing views. So RQ 2 is placed on the function level.

RQ4 is a function for user. So RQ4 is placed on the function level.

2. Is the requirement include specific suggestions of how things solve?

RQ1 is a solution for own user interface and is the candidate for a component level.

RQ5 means only manager can control the system and is placed on the component level.

3. Is the requirement comparable to the product strategies?

RQ3 is abstract in nature. But RQ1 is only close to the product strategies and is not directly comparable to the product strategies.

4. Is the requirement describes “what the system should include or support”?

RQ3 means that the system only support Swedish and English and the target market is Sweden now. And it will be extended in the future. So RQ3 is placed on the Feature level.

- Step three: Abstraction(work-up)

This step is abstract and breakdown requirements and creates new requirements. There are two tules in this step:

Rule 1: Requirements can not exist without a connection to the Product level.

Rule 2: All the requirements need to be broken down to function level.

RQ1: Restricted User Interface.

Product level: Keep user information security. This is a created requirement.

Feature level: Own user interface.

Function level: User need to use username and password to access to their own interface. This is a created requirement.

Component level: Restricted User Interface. This is the original requirement.

RQ2: Personalize Views.

Product level: This is the usability of the system.

Feature level: User can have their personalize view.

Function level: Personalize Views. This is the original requirement.

RQ3: Market and User Interface Language.

Product level: This is the usability of the system.

Feature level: Market and User Interface Language. This is the original requirement.

Function level: Use can choose the language when they are using the system. This is a created requirement.

Component level: The interface can adopt to the language text length. This is a created requirement.

RQ4: User login.

Product level: This is the usability of the system.

Feature level: User must login to view their own interface.

Function level: This is the original requirement.

Component level: User need to choose their own username and password.

RQ5: Secure Product.

Product level: This is the stability of the system.

Feature level: Only system manager can have the authorization to control the system.

Function level: System manager can control the users' function and modify user information. This is a created requirement.

Component level: This is the original requirement.

After these steps, the requirements are abstracted. After the abstraction process is completed, each requirement should be attached with other attributes, include requirements source, requirements owners, and the dependencies related to other requirements, status, date of creation, and modification records.

Method 2—QUPER:

To implement QUPER, an example is a student eduction system. This system is for students and teachers. The target market now is Sweden schools that have international students.

Quality performance (QUPER) model has several steps. I will follow these steps to performed QUPER.

Step 1: Identify candidate QR :

QR1: A competitor recently launched the educational system. The user login time is 3 seconds;

QR2: A competitor recently launched educational administration system, the speed of user uploads and downloads the file is 15M / s;
Step 2: Define a scale and a measurement unit:
QR1: Scale-time unit: Seconds;
QR2: Scale-time unit: M/s;
Step 3: Identify reference levels: For both QRs, two reference levels are defined according to the competing products. All data are referenced from the network and are both fictitious.
QR1 :
Feature: Student educational system login time.
Quality Requirement: Time to login to the system.
Definition: The number of seconds required to login to the system.
Reference levels:
Competitor A level: 3 seconds;
Competitor B level: 5 seconds;
Own system C level: 3.5 seconds;

QR2:
Feature: The speed of upload or download files.
Quality Requirement: Time to upload or download files from the system.
Definition: The speed of upload or download files from the system.
Reference levels:
Competitor A level: 15M/s;
Competitor B level: 5M/s;
Own system C level: 10M/s;

Step 4: Define market expectations and elicit quality breakpoints:
The utility breakpoint marks the border of quality from useless to useful. The differentiation breakpoint marks the level of the quality is in a competitive position. The saturation breakpoint marks the product is excessive that higher quality level has no practical impact on benefit.
QR1:
Utility: 8 seconds;
Differentiation:4 seconds;
Saturation: 1 seconds;

QR2:
Utility: 512KB/s;
Differentiation: 8M/s;
Saturation: 20M/s;

Step 5: Estimate the cost barriers:
For these two QRs, two cost barriers are defined. The first cost barrier is related to software change and updates. The second cost barrier is related to system repair.
QR1:
Qref(own product): 3.5 seconds;
Q11: 3 seconds;
Rationale: Software system updates, add new features.
C11: 2 weeks;
Q12: 2.5 seconds;
Rationale: System maintenance and functional repair.
C12: 4 weeks;

QR2:
Qref 2: 10M/s;
Q21: 15M/s;
Rationale: Software system updates, add new features.
C21: 3 weeks;
Q22: 20M/s;
Rationale: System maintenance and functional repair.
C22: 4 weeks;

Step 6: Propose candidate requirements target:
QR1:
GOOD:3 seconds;
Rationale: will beat most of the competitors;
STRETCH: 2 seconds;
Rationale: if system function updates.

QR2:
GOOD: 15M/s;
Rationale: will beat most of the competitors;
STRETCH: 25M/s;
Rationale: if system function updates.

Step 7: Identify cost dependencies between QRs:
Assume that QR1 and QR2 affect the same architecture part. So QR1 and QR2 are interdependent. By analyzing the cost estimates and importance of the two requirements, I decided to implement QR1 because it was more important than QR2, and requiring less cost and effort. However, in practical projects, the cost dependencies of these QRs have been identified by experts, and experts have chosen a range of requirements that are easier to achieve than other dependable requirements. And then estimate the workload for all selected requirements. If the required resources are higher than the available resources, the experts will change the requirements.

4. Lessons Learned:

Market-driven development is different from the bespoke development that the product is specific to the needs and wishes of one customer. MDRE refers to the situation where the product is offered to an open market with many customers, the development costs are divided among many buyers and the potential profit is rewarded to the producer. MDRE covers the classical RE activities like elicitation, specification and validation. It also covers the release management and market analysis which is specific to MDRE.

RAM:
By completing the entire RAM process, I learned that under the MDRE environment, in the product life cycle, the requirement flow is continuous. There are many sources of requirements, and there are different levels of complexity for them. Inaccurate treatment of requirements can lead to the inability to meet market needs while wasting effort and money. In the early stages of development, there are a lot of dependencies between requirements that need to be dealt with carefully. Under the MDRE environment, the requirements flow is continuous and comes from multiple stakeholders, so the requirements are difficult to deal with. The RAM model provided by this article is a good solution to this challenge. RAM provides four levels of abstraction, each level has a very clear description, and there are corresponding links between the various levels. This helps to combine

requirements and product strategy while helping developers understand every requirement. During the process of implementation, I understand the reasons, benefits and possible risks for each requirement. Additional requirements from existing requirements are also clearly described and added to different abstraction levels at the same time. I have a clear understanding of how to handle requirements in MDRE, handle dependencies between requirements, and allocate requirements to different releases.

QUPER:
As can be seen from this article, organizations often face the challenge of identifying the right market requirements for market needs and competitor products. Market-oriented software products should be effective in planning the product release so that products can have more competitive quality level compare to competitors and release to the market as soon as possible. Therefore, the quality requirement is the main part of the company's competition. Through the implementation of QUPER, I clearly know how to develop quality requirements and how to compare the quality requirements to competitors. At the same time, I also understand the importance of cost, benefit and roadmap in the QUPER model, and the importance of cost dependency in deciding different release plan.

5. Reflections:

RAM:
In order to compare my implementation experience, I read other RAM related articles.
Because I only listed five requirements, so my understanding may not be entirely accurate. In [3], the authors point out that the opportunity to create new requirements is reduced when moving from component level to product level. I agree with this view. Because most component level requirements are suggested by the developer and are very detailed. So it's hard to link them with the product strategy. I agree with most of the views raised by the author. But because I do not have the large-scale application of RAM, so some conclusions have not been verified.
In [4], the author collects comments on the implementation of RAM by doing case studies in two companies. The results show that through the use of RAM, the two companies have greatly improved product quality. But the application of RAM also needs more effort and cost. But with the increasing of accuracy and quality, the application of RAM will get more benefits. RAM is independent of the tool, but the expansion of the application also requires some tools to support.

QUPER:
For QUPER, it is important to know that breakpoints will change over time. Regnell said that in mature markets, utility and saturation are relatively stable over time, but there are still differences happen. The QUPER way of thinking is very important. As it forces you to know where your statement in the market, what you want to achieve and the cost you need to achieve your goals. QUPER's three views are a good indication of why these goals are set.

In paper [5], except to introduce the implementing process of QUPER, the article also focused on the cost dependence. This helps me to better handle the cost dependencies between QRs in my implementation process.

In paper [6], the authors used case study to evaluate the QUPER model. The results show that QUPER is related to early decision-making processes. Breakpoints, competitor analysis, cost barriers and the definition of their own product quality levels provide a good understanding of the level of requirements required for future release planing. This article through the case analysis defined two main challenges of QUPER. It is difficult to define the values of the differentiation breakpoints and saturation breakpoints and it is difficult to define the cost barriers and their value. Through my practice, I agree with the author's viewpoints. In my practice, breakpoints and cost barriers are indeed very difficult to determine. But because I have only implemented two quality requirements, it is difficult to draw more conclusions. But through this article and my practice, I realized that defining these values is indeed very difficult.

Reference

[1] T. Gorschek and C. Wohlin, “Requirements Abstraction Model,” Requirements Eng, vol. 11, no. 1, pp. 79–101, Mar. 2006.

[2] R. B. Svensson and B. Regnell, “A Case Study Evaluation of the Guideline-Supported QUPER Model for Elicitation of Quality Requirements,” in Requirements Engineering: Foundation for Software Quality, 2015, pp. 230–246.

[3] N.Muhammad, “Suitability oftheRequirements Abstraction Model(RAM)Requirements for High Level System Testing,” no. October, 2007.

[4] T. Gorschek, P. Garre, S. B. M. Larsson, and C. Wohlin, “Industry evaluation of the requirements abstraction model,” Requir. Eng., vol. 12, no. 3, pp. 163–90, Apr. 2007.

[5] R. B. Svensson, T. Olsson, and B.Regnell, “Introducing support for release planning of quality requirements - an industrial evaluation of the QUPER model,” in 2008 Second International Workshop on Software Product Management, 9 Sept. 2008, pp. 1-9.

[6] R. Berntsson Svensson, Y. Sprockel, B. Regnell, and S. Brinkkemper, “Setting quality targets for coming releases with QUPER: an industrial case study,” Requir. Eng., vol. 17, no. 4, pp. 283–98, Nov. 2012.