***Large Scale Requirement Engineering***

***Reflective Report***

Yi Chen

Chenyi1508@gmail.com

BTH University

Sweden, Karlskrona

19940121-6131

***Abstract – This document is about the reflective report of Large Scale Requirement Engineering course. Two articles related to the methods or techniques in Large Scale Requirement Engineering have been selected and implemented in the report.***

**Article One [1]:** Pu Y N, Liu Q. A viewpoint-oriented requirements elicitation integrated with aspects[C]//Computer Science and Information Engineering, 2009 WRI World Congress on. IEEE, 2009, 7: 706-711.

1. **Motivation**

The motivations for selecting first article are as follows: 1. Requirement elicitation is one of the most important phase in requirement engineering. The success of large-scale software systems is based on how accurate the large number of requirements is elicited and analyzed in the early stage [1]. However, in large scale requirement engineering, different stakeholders have different responsibilities and concerns for the system which may causes some problems. The most common situation is that there are some conflicts among different stakeholders. Boehm pointed out that if these conflicts cannot be handled correctly, there would be 200 times cost in the test period [2]. Therefore, the authors of first article put forward a method to deal with requirement conflicts which can benefit to improve the accuracy in requirement elicitation and lower the cost in the test period. 2. I am interested in the requirement engineering and I have participated several courses related the requirement engineering. However, I did not pay much attention to the requirement conflicts which is especially important in large scale requirement engineering. The first article light the idea of providing an effective way to document requirements which can give support to solve the conflicts among different viewpoints. Therefore, the first article is an enlightening study material for me.

1. **Implementation plan**

There are many ways to elicit requirements such as interviews, survey, questionnaires and etc.

A huge number of requirements can be obtained by these methods. However, the requirements are document loosely, immaturely and optionally. It is rather important to document these requirements via templates and then the softer engineers can get a lot of help when solving the requirement conflicts.

The **first stage** is to format the requirements. There are five steps in this stage.

1. Identify and define viewpoints. The viewpoints related to the system should be identified first which includes the name, source stakeholder, objective, feature, non-functional requirement, use case and history.
2. Define objectives. The system goal in this viewpoint should be defined which has three elements: (1) Name: The identifier of objectives. (2) Viewpoints: viewpoints related this objective. (3) Description: a description about the objective.
3. Define features. Some features can be proposed in different viewpoints when after defined the objectives. Features can be regard as essential parts of objective. Several elements should be contained in the feature. (1) Name: the identifier to the feature. (2) Objectives: list the objectives related to the feature. (3) Description: a description of feature. (4) Weights: It is about the influence on corresponding objective. The scale is between 0 and 1.
4. Define use cases. The use cases from function part can be extended form features. And in large scale requirement engineering, a use case may belong to different features, objectives and even viewpoints. The use cases will be defined with following elements. (1) Name (2) Description. (3) Actors: the actors use the use case (4) features: feature that this use case belongs to. (5) Primary scenario: main scenario. (6) Secondary scenario: exceptions (7) NFR: none function requirements that related to this use case.
5. Define non-functional requirements. The conflicts may occur because one non-function requirement may relate to several use cases, features, objectives and viewpoints. So, a template of NFR which contains name, description, influence (with both features and use cases) and priority (score from 1 to 5, 1 means low, 5 means very high) should be identified.

The **second stage** is to resolve conflicts by better documentation. The crosscutting concerns are difficult to understand and maintain. So, it is rather important to use aspect-oriented requirements engineering to specify the crosscutting concerns in separate modules. This method identifies the crosscutting concern clearly and relate these conflicts to the “feature” attribute in the template which help to discuss and handle conflicts easily. There are three steps in in this stage.

1. Identify aspectual use cases (AUC). The use cases belong to several features can be regarded as aspectual use cases. If a UC crosscut several features, there should be some crosscuts in functional requirements. So, document the AUC can help us to reduce the conflicts and we can easily tell which use cases are included by which features or from which viewpoint.
2. Identify aspectual non-functional requirement (ANF). It is similar to AUC, if one NFR crosscuts several use cases, features, objectives, or viewpoints, it is regarded as ANF.
3. Identify and solve conflicts. A method called AORE [3] will be used in this step which uses a table to relate aspectual elements and viewpoints. The identified relation will be help to document and deal with conflicts.
4. **Execution**

Followed the implemented plan the following execution will be performed. I use an online library system to execution the plan. The system accepts one of the stakeholder--customer to log in/out, borrow books, return books, etc. Another stakeholder is administrator who can log in/out the system and manage the system, etc.

**Stage one:** Format the requirements.

1. Define viewpoints. We have defined two viewpoints in the online library system. One is customer and another one is administrator. I take the customer viewpoint as an example.

Table 1. customer viewpoint

|  |  |
| --- | --- |
| VP | Description |
| Name | Customer viewpoint |
| Source stakeholder | Customer |
| Objective | Provide service |
| Features | Log in, borrow book, return book, etc. |
| UCs | Log in, borrow book, return book, etc. |
| NFRs | 24-hour service, etc. |

1. The object of customer viewpoint is “provide service” which means the customer can get the services from the online library system. On the meanwhile, the administrator can get the service such as log in/out query book from the system. So, we can find that different viewpoints may have a same objective.

Table 2. “Provide service” objective

|  |  |
| --- | --- |
| Objective | Description |
| Name | Provide service |
| Viewpoints | Customer, administrator |
| Description | Provide query service and log in/out |

1. Some features will be added to the objective. I take “can log in” feature as an example which is showed in table 3.

Table 3. “Can log in” feature

|  |  |
| --- | --- |
| Feature | Description |
| Name | Can log in |
| Objective | Provide service |
| Description | The system accepts correct users request to log in |
| weight | 1 |

1. Deign use case. There are two different viewpoints – customer and administrator which all should have the use case of log in. And one use case may include by many features. Table 4 take the “log in” use case as an example.

Table 4. “Log in” use case

|  |  |
| --- | --- |
| UC | Description |
| Name | Log in |
| Description | Provide 24-hours log in service for user |
| Actors | Customer, Administrator |
| Features | Can log in |
| Primary scenario | 1. Open the webpage 2. Enter user name and pin code, correct 3. Log in successfully |
| Secondary scenario | 1. If the user name or pin code is not match or correctly. Inform reenter the username and pin code. |
| NFR | Security, Performance |

1. Define NFR. We should document the NFRs to implement features. And one NFR may also related to several features. Table 5, 6 and 7 show NFR of 24- hours service, security and performance.

Table 5. NFR of “24-hource service”

|  |  |  |
| --- | --- | --- |
| NFR | | Description |
| Name | | 24 hours’ service |
| Description | | Online library system provides 24 hours’ service |
| Influence | Feature | Can register borrow book, query personal information, borrowing information, etc. |
| UCs |  |
| Priority | | 5 |

Table 6. “Security” NFR

|  |  |  |
| --- | --- | --- |
| NFR | | Description |
| Name | | Security |
| Description | | Require messages validation if the user name and pin code not match for five times. |
| Influence | Feature |  |
| UCs | Log in |
| Priority | | 5 |

Table 7. “Performance” NFR

|  |  |  |
| --- | --- | --- |
| NFR | | Description |
| Name | | Performance |
| Description | | Log in service should have less than 1 second response time. |
| Influence | Feature |  |
| UCs | Log in |
| Priority | | 5 |

**Stage two:** Identify and solve conflicts

1. Identify AUC. The “log in” use case is related to the both customer viewpoint and administrator viewpoint which crosscut two requirement domains. So, the “Log in” use case is regards as an AUC. And because the weight of the “can log in” feature is 1 so the “log in” use case impact of viewpoint is 1 as well.

Table 8. AUC for different viewpoints

|  |  |  |
| --- | --- | --- |
| AUC | Customer viewpoint | Administrator viewpoint |
| Log in | 1 | 1 |

1. Identify ANF. These three non-function requirements -- “24 hours services”, “security” and maintainability all relate to the customer viewpoints and administrator viewpoints. So, these three are all ANF. Besides, the weight of the feature “can log in” is 1 so these three NFR’s impact of viewpoints are all 1.

Table 9. ANF for different viewpoints

|  |  |  |
| --- | --- | --- |
| NFR | Customer | Administrator |
| 24 hours’ service | 1 | 1 |
| Security | 1 | 1 |
| Performance | 1 | 1 |

1. Identify and solve conflict. We can find that these two different stakeholders may have different desire in security and performance where conflicts may occur. So, we can relate the ANF to the stakeholders which help to document the conflict and negotiating on the conflicts.

Table 10. Conflicting ANFs between stakeholders

|  |  |  |
| --- | --- | --- |
| ANF | Customer | Administrator |
| Reliability | 1 | 1 |
| Performance | 1 | 1 |

1. **Lessons learned**

In LSRE, there are many stakeholders in the system. One use case may have the different viewpoint from the different stakeholders where conflicts may occur. I learned that the aspect-oriented approach can help to identify the conflicts effectively since these conflicts are all come from AUC or ANF. Besides, the stakeholders related to the conflicts should be found which can help to solve the conflicts easily by later negotiation. The templates author proposed are really help to document the requirements and analyze the conflicts in the early stage of LSRE. It mainly focuses finding the conflicts on the viewpoints from different stakeholders. We can use these templates to analyze and document the requirements accurately which can help to reduce the cost in the test period and benefit to the project in the long run. All in all, the aspect-oriented method can analyze conflicts and document the requirements effectively and efficiently in LSRE.

1. **Reflection**

Rashid [3] also propose aspect-oriented requirement engineering which can identify the crosscutting requirement at the early stage of LSRE and analyze their impacts in the later stage. It is similar to the first authors idea and it is proved that finding the conflict at the early stage can save the cost in the feature stage. Besides, VORD [4] has put forward a method that consist of viewpoints identification, requirement specification of viewpoints, and conflicts identification and resolution which is also relate to find the conflict among different viewpoint in the early stage. In [5] Moreira indicate that functional requirements should be described by use case, and non-functional requirements should also be described by using templates. Besides, they propose a set of models to deal with the crosscutting part of requirements. So, it is obvious that the crosscutting part of requirements have drew much attention in the field of LSRE. And it is suggested that not only the function requirement but also the non-functional requirement should be documented by use cases or specific templates which will benefit to find the conflicts from different viewpoints and reduce cost for large scale system development.

**Article Two [6]:** Bebensee T, van de Weerd I, Brinkkemper S. Binary priority list for prioritizing software requirements[C]//International working conference on requirements engineering: foundation for software quality. Springer Berlin Heidelberg, 2010: 67-78.

1. **Motivation**

The motivations for selecting second article are as follows:1. The requirement prioritization is a key stage in requirement analysis. It is because the resources are limited in LSRE, the product managers have to make a selection of incoming requirements that should be implemented [5]. So, it is rather important to find a suitable prioritization method to prioritize requirements. However, many prioritization techniques are not suitable for prioritizing larger amount of requirement. The second paper introduces a prioritization method called Binary Priority List (BPL) which can deal with this situation. It uses binary search based technique to prioritize requirement which can provide a lot of suggestion for product manager. 2. In the previous assignment, I have made a release planning for a course management system which has 208 requirements. In that case, I divided those requirements into 12 types based on the dependencies and their functionality. A. Upload and download file B. Non-functional requirement C. Communicate D. log in and log out E. Notify F. Edit or delete G. View H. Search I. Other J. User interface K. Evaluate L. Register. These 12 types are all high-level requirements and there are a medium number of low-level requirements in each high-level requirement. And then, I used the MosCow method to prioritize the low-level requirements. But now, I read the second article which related to BPL. And BPL is recommended to apply for a medium number of low-level requirements which seems suitable for the scenario I mentioned above. So, I want to learn and try to apply BPL to the course management system.

1. **Implementation plan**

Binary Priority List (BPL) is similar to the Binary Search Tree algorithm. All in all, the requirements in right node have higher and requirements in left node have lower priority than their root node requirements. The steps to perform the BPL techniques are as follows:

1. Make a list of all the requirements need to prioritize
2. Choose one requirements from list and take it as the root requirement
3. Take another requirement and compare the priority with the root requirement.
4. If the requirement has a lower priority than the root requirements, compare it to the left node of requirement and so forth. If the requirement has a higher priority than the root requirement, compare it to the right node of requirement and so forth. Perform the previous action until the requirement can be place as a node in the BPL.
5. Repeat step 2 to 4 for all requirements.
6. Traverse the list to get the prioritized order of all the requirements.
7. **Execution**

As I mention in part 1, I would like to execute the plan on the course management system. And I plan to chooses six low-level requirements from the “view” type to do the BPL prioritization as an example.

# 35 Basic Contents of Personal Profile: The Personal Profile shall contain first name, last name, social security number. Most basic information in order to be able to find information about user in e.g. other systems.

#47 Access to View Course Info: Any user of the system shall be able to view the course info. out me", link to personal homepage.

#49 View Course Links: It shall be possible to view links attached to a course

# 97 Teacher timetable and meeting: It provides user to access the timetable of teacher and book an appointment with teacher/lecturer.

# 132 Hand in overview: As an teacher assistant I would like to have overview of the hand in that are related only to me so that I don't correct wrong hand in ]

# 159 Student information view: As a teacher, i would like to view students' information who attend my class

The first step is to make a list of all the requirement that need to prioritization which I have done above.

The second step is to choose a requirement as a root. I randomly choose #159 as the root requirement.

The third step is to take other requirements and make prioritization for the BPL.

1. Take #32. I assume the prioritization of #32 is higher than #159 since #35 provide the personal information for teacher in #159 to check.

Form the BPL:

1. Take #47. I assume that #47 is more important than #159 and #32 since it is the core requirement for course management system.

Form the BPL:

1. Take #49. I assume the #49 has low prioritization than #159.

Form the BPL:

1. Take #97 I assume that it is less important than #159 and #49.

Form the BPL:

1. Take #132 I assume it is less important than #159 but more important than #49.

Form the BPL:

The final step is to traverse the list to get the prioritized order of all the requirements: #47>#32>#159>#132>#49>#97.

1. **Lessons learned**

I apply a new prioritization method – BPL to the release planning scenario at this time. As far as I am concerned, BPL is time consuming but it is more effective than MosCow in theme-based or type-based prioritization. Because there would be continuous stream of incoming requirements in LSRE. If a new incoming requirement is required to add in the next release so some requirements should be move out of the release because of the resource limitation. If we have used the MosCow to prioritize the requirements, it is hard to determine which requirement should be removed because all the requirements in the same hierarchical priority group share the same priority. On the other hand, we can easily remove the lowest priority requirements from the release plan if we have used the BPL method to prioritize the requirements. Another thing I have learned is that BPL is not suitable to apply to the large-scale requirement system directly since prioritize hundreds of requirements by BPL is really time consuming even with the help of BPL tools support. But BPL works well if we can divide the requirement into several types which have medium number of low-level requirements.

1. **Reflection**

There are some other articles that in favor of using BPL in requirement prioritization. For example, Racheva [7] propose to use Binary Priority List method for prioritizing requirements in medium or large scale of requirement system and this method is proved easy to use (The author refer the BPL as “Binary search tree technique” in the article). Besides, there are many other prioritization methods that can be used in the LSRE such as spanning tree matrix, MosCow, planning game, Wiegers’s matrix approach [8] and AHP. These different prioritization methods all have its own advantages and disadvantages. It is impossible to tell which prioritization method is the best or the worst. The thing for sure is that the most suitable prioritization method for the requirement scenario is always the best.

**Reference**

[1] Pu Y N, Liu Q. A viewpoint-oriented requirements elicitation integrated with aspects[C]//Computer Science and Information Engineering, 2009 WRI World Congress on. IEEE, 2009, 7: 706-711. [1]

[2] Boehm. Software Engineering Economics. Prentice Hall PTR, 1981.

[3] A. Rashid, P. Sawyer, A. Moreira, J. Araújo. Early Aspects: a Model for Aspect-Oriented Requirements Engineering. Requirements Engineering 2002 (RE'02), Essen, Germany, 9-13 September 2002.

[4] G. Kotonya, I. Sommerville. Requirements engineering with viewpoints. BCS/IEE Software Eng. J., 11(1): 5-18

[5] A. Moreira, J. Araújo, I. Brito. Crosscutting Quality Attributes for Requirements Engineering. 14th International Conference on Software Engineering and Knowledge Engineering (SEKE 2002), ACM Press, Italy, July 2002.

[6] Bebensee T, van de Weerd I, Brinkkemper S. Binary priority list for prioritizing software requirements[C]//International working conference on requirements engineering: foundation for software quality. Springer Berlin Heidelberg, 2010: 67-78.

[7] Racheva, Z., Daneva, M., Buglione, L.: Supporting the Dynamic Reprioritization of Requirements in Agile Development of Software Products. In: Proceedings of the Second International Workshop on Software Product Management 2008, Barcelona, pp. 49–58  (2008)

[8] Wiegers, K.: First things first: prioritizing requirements. Software Developmen 7(9),  48–53 (1999)