***Large Scale Requirement Engineering***

***Reflective Report***

Mingyu Ren

renmingyu1994@gmail.com

BTH University

Sweden, Karlskrona

9403280648

**1. Article Selection**

**1.1 First Article**

**Article information:** 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.

**Article name:** Binary priority list for prioritizing software requirements

**Motivation for selection:**

The motivations for selecting this article are as follows:

1. Requirements prioritization is a very important part of requirement engineering domain. This article is an article about the requirement prioritization, which helps us understand and learn how to select incoming requirements. This is the first reason for selecting this article.

2. As we know that find a suitable prioritization method to prioritize requirements is very important for a software product manager, but many prioritization techniques can not suitable for many requirements. This article introduces a new method named BPL, that can deal with many requirements. This is the second reason why we chose this article.

3. For our own product, we need to deal with 208 requirements from the GitHub, we have already divided these 208 requirements into 10 types based on their functionality. But these 10 types of requirements are all high-level requirements, for each type of requirements, they all have a lot of low level and medium level requirements. BPL can deal with these kinds of requirements. This is the third reason why we chose this article.

**1.2 Second Article**

**Article information:** Hatton S. Choosing the right prioritization method[C]//Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on. IEEE, 2008: 517-526.

**Article name:** Choosing the right prioritization method.

**Motivation for selection:**

The motivations for selecting this article are as follows:

1. Requirements prioritization is an important step in the software development process. The article introduced several requirements priority method. We can learn a lot of requirements priority method from this article. This is the first reason for selecting this article.

2. As we know that there are many methods available for prioritizing software requirements. But chose a suitable method is difficult. There are many factors will influence the result of chose. This article examines the type of information available at different stages in a project and matches it to the properties of prioritization methods. This is the second reason why we chose this article.

3. The published year of the article is 2008, it is recent. And this article introduced a lot of requirement priority methods, and the author show the characteristic and principle of each method. It is valuable in the requirement domain. This is the third reason why we chose this article.

**2. Implementation Plan**

**2.1 The Plan for First Method**

Binary Priority List (BPL) is a binary search based technique for prioritizing requirements. As we know that in binary search, there are three main nodes, they are right node and left node and root node. The requirements in right node have higher prioritization. And requirements in left node have lower priority than root node requirements. Perform the BPL techniques are divided into the following steps：

1. Select one type of requirements, and list of all the sub-requirements included in this kind of requirements group.

2. Choose one requirements from the above list and let it become the root requirement.

3. Chose another requirement from the above list and compare the priority with the root requirement.

4. If the new chose requirement has a higher priority than the root requirement, compare it to the right node requirement. If the new chose requirement has a higher priority than the right node requirement, this requirement can be place as a node in the BPL.

5. If the new chose requirement has a lower priority than the root requirement, compare it to the left node requirement. If the new chose requirement has a higher priority than the left node requirement, this requirement can be place as a node in the BPL.

6. Select another new requirement from the above list and repeat step 2 to 5.

7. Traverse the list to get the prioritized order of all the requirements.

**2.2 The Plan for Second Method**

Because this article introduced a lot of requirement priority methods, and the author show the characteristic and principle of each method. So, at first, I would like to select a priority technique as my technique to perform our own project. I chose to use MoSCoW technique for my technique. Perform the MoSCoW techniques are divided into the following steps：

1. Classify the 208 requirements into 10 type of different types.

2. Deep understand these 10 type of requirements, we based on our project actual requirements and the description of each requirements to deeply understand the 10 large category requirements.

3. Divide these 10 types of requirements into 4 different levels. They are Must Have, Should Have, Could Have and Won’t Have.

4. Do the release planning, and first focus on the Must have requirement, and then focus on the Should have requirements, could have requirements and at last focus on the Won’t have requirements.

**3. Execution**

**3.1 The Execution for First Method**

In order to execute this course management system, at first, I will choose six low-level requirements from the requirements list. And use them to do the BPL prioritization as an example. The low-level requirements we selected will be shown below:

#27 Login/Logout: The user must login before being able to access the product. Unauthorized users should not have access to product functionality. This also enables us to customize information for user. Also, the user can logout.

#61 Incorrect Login: If the user enters an incorrect user id and/or password the login page shall be reloaded with information showing that an incorrect login has been attempted.

#89 Information provided to the user on restriction: Login page should provide the user with the information that the login will take place only via the credentials provided by the system admin.

#103 Automatic logout: The system must automatically logout if the user is not using the system for a long period or if the tab/browser is closed.

#137 Log in time reset: As a teacher assistant I would like the system to remember me for long time and not ask for log in information often so that I am not disturbed when I use the system for long periods of time.

#197 Account automatically sign out: As an old engineer student, I would like that my account automatically sign out when I have been leaving the system for several minutes.

Then I chose a requirement as a root requirement, I choose #27 as the root requirement.

Next, I take other requirements and make prioritization for the BPL.

(1) Take #61, I assume the prioritization of #61 is higher than #27, because If the user enters an incorrect user id /or password, he (she) could also login this system, it is show the system have a very poor security.

From the BPL:

(2) Take #89, I assume the prioritization of #89 is less important than #61, but is more important than #27.

From the BPL:

(3) Take #103, I assume the prioritization of #103 is less important than #27.

From the BPL:

(4) Take #137, I assume the prioritization of #137 is less important than #27, but more important than #103.   
From the BPL:

(5) Take #197, I assume the prioritization of #197 is more important than #137.

From the BPL:

The final step is to traverse the list to get the prioritized order of all the requirements: #61>#89>#27>#197>#137>#103.

**3.2 The Execution for Second Method**

For executing MoSCoW method, I classify the requirements into 10 different types at first. Then I give labels to each requirement for "Must have", "Should have", "Could have" and "Won't have" according to my own understanding for each type of requirements. At last, executing the release planning and follow the order of Must->Should->Could->Won't.

The 10 different types of requirement will be shown below:  
R1. User Interface  
R2. Log in and log out  
R3. Manage personal information  
R4. Search function   
R5. Upload and download files  
R6. Notify  
R7. Evaluation functions  
R8. Course management  
R9. Communicate with other users  
R10. Others

Labels to each type of requirement for "Must have", "Should have", "Could have", "Won’t have" will be shown below table:

|  |  |  |
| --- | --- | --- |
| ID | Requirements name | Type of MoSCoW |
| No.1 | User Interface | Must have |
| No.2 | Log in and log out | Must have |
| No.3 | Manage personal information | Could have |
| No.4 | Search function | Should have |
| No.5 | Upload and download files | Must have |
| No.6 | Notify | Won’t have |
| No.7 | Evaluation functions | Could have |
| No.8 | Course management | Could have |
| No.9 | Communicate with other users | Should have |
| No.10 | Others | Could have |

**4. Lessons Learned**

**4.1 The lessons learned from the first article**

I select a new prioritization method named BPL to do the release planning process. Compare BPL method to MosCow, I have found although BPL is time consuming, it is more effective than MosCow method. Because BPL not only could deal with high level requirements but also could deal with medium and low level requirements. BPL has a bigger advantage is that it can easily remove the lowest priority requirements from the release plan if we deal with the incoming requirements. Compare with it, MosCow become more difficult to remove the lowest priority requirements from the release plan when there are some incoming requirements. Because the MosCow method put all the requirements in a same hierarchical priority group and they share a same priority. Another thing we learned from this article is that BPL is not suitable to apply to the large-scale requirement system directly. The large-scale requirement system means there are more than one thousand requirements. Because BPL technique deal with all kinds of requirements and time consuming is the biggest problem for BPL technique. I think a good method to deal with this problem is divide the requirement into several types which have medium number of low-level requirements and then use BPL technique again and again.

**4.2 The lessons learned from the second article**

From this article, I known that MoSCoW can deal with requirement prioritization problem. It is not a very difficult method, because it only need to divide the requirements into 4 different levels. And when performing release planning, the requirements are executed in the order of Must Have, Should Have, Could Have and Won’t Have. In a word, this method is easy to perform, handle and category the requirements. However, one of the drawbacks of this method is that we choose which requirement belong to which level is based on our own understanding of the requirement. This may cause classification errors because of we misunderstand the requirement. So, I think we can execute MoSCOW to quick inspired requirement prioritization and help us to deep understand the requirements. But we cannot use this method to deal with a large of requirements in the real project. Another thing I have found is that if we have used the MosCow to prioritize the requirements, it is hard to determine which requirement should be removed.

**5. Reflection**

**5.1 The reflection from the first article**

There are some other articles that talk about the BPL method. For example, Racheva [3] presents results of a literature review on agile requirements prioritization methods. This article refers the activity of continuous requirements reprioritization forms the very core of today's agile approaches. The BPL method is proved easy to use for prioritizing requirements in medium or large scale of requirement system. And from the literature [4] we also found many other prioritization methods, such as planning game, spanning tree matrix, MosCow, Wiegers’s matrix approach. From this paper, I know that different prioritization methods will have different advantages and disadvantages. We cannot select a best or worst prioritization method. We need to select a suitable prioritization method according to the real situation.

**5.2 The reflection from the second article**

From my implementation, I have found that MoSCoW can deal with requirement prioritization problem and it is an easy method that could be used in requirement prioritization. This method will divide the requirements into four different levels, they are Must Have, Should Have, Could Have and Won’t Have. There is another article that talk about MoSCoW method. The article [5] refers that if a project only has highest level priority-"MUST have" requirements, the scope can't be varied. And MoSCoW could be judged the option by several criterias, they are business value, relevant risk, difficulty of implementation, relationship to other requirements, stakeholder agreement and so on. From the article [5], I also known that when we doing the requirement prioritization in LSRE, we need to make a list of healthy number of ''Should have'' and ''Could have'' requirements. And try to give them some flexibility.

**6. Reference**

[1] 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.

[2] Hatton S. Choosing the right prioritisation method[C]//Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on. IEEE, 2008: 517-526.

[3] 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).

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

[5] Waters K. Prioritization using moscow[J]. Agile Planning, 2009, 12.

[6] Hatton S. Choosing the right prioritization method[C]//19th Australian Conference on Software Engineering (aswec 2008). IEEE, 2008: 517-526.

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