CSCI 5801
Assignment 4, Software Requirements Specification Reflection
Guanpin Zhong, 5362045
March 26, 2019

1. Elicitation Session

a. What did you learn from the elicitation session? What parts worked well? What parts did not work well?

From the elicitation session, I learn that it is really important to hear what requirements the users have. Sometimes users even do not know how the product should work, especially for some details. We are supposed to ask as detailed questions as possible in order to get some information about what exactly the users want.

The part that worked well was that lots of people asked about some actions they take during some processes. For example, the two users answered a question about who will actually send the offer to the appointed TAs, and they said it was payroll staff. That is really helpful as originally, I thought it should be CSE staff.

The part that did not work very well is that we were not able to ask all types of users. There were only one administrative staff and one payroll staff, but there was confusion about what the faculty members and prospective TAs want the application to do. The two people could say vague answers when they were not sure about them.

b. How might you prepare differently for a similar experience in the future, where you may only have one chance to meet and talk with the user(s)?

To be honest, our group did not prepare a lot before asking the users. I will make more preparations next time in order to make a more organized set of questions to ask the users. Even though we may only have one chance, organized questions covering different parts of the conducting situations, demands and other details will help a lot for us.

c. When is asking the same question twice bad/unhelpful? When is asking the same question twice good/helpful?

When an answer of a question is obvious and easy to answer, it is bad to ask it twice. It only wastes time and that is not helpful. When an answer of a question is very long and includes multiple parts, such as describing the whole process of considering the qualifications of TAs and assigning them to the most appropriate courses, it is good to ask the question twice. That way engineers or designers can remember more details.

2. Individual Requirements

a. How well did capturing the requirements go?

I spent much time to understand all the tasks the application is supposed to do and figure out specific demands from every type of users. I think it is important to pay attention to details and try to list all the actions thoroughly. It is really helpful to write down all the possibilities on the paper at first, and then reword them for consistency. After that I will put them in different areas which means for different users or functionalities. That way I am able to have a clear map of the useful requirements not only in my mind but also in records, so that for further assignments I will be able to keep track and make improvements continuously.

b. How did you approach user statements/requirements which were vague or ambiguous?

For those vague user requirements, I tried to write down all the possible cases I can think of and consider if they will fit the real situation carefully. Usually I chose the most proper one based on my TA experience and other information I got, and that will help to realize assignment goals much easily in the future.

c. How did you approach user statements/requirements which were contradictory?

When two user requirements are contradictory, I tried to put them into the whole environment, and compare the possible influences which changing them will lead to. I would choose the one resulting in less changes, as that means it is more proper in general.

d. Did you follow a template? Did the template help or hurt your efforts? Why?

Yes, I followed a template, which is the IEEE one. It helped a lot as it provided basic ideas of what items I was supposed to write. But actually, not all the items worked, and I picked some meaningful ones for my user requirements.

3. Team Requirements

a. Did you encounter any situations where you and your teammates disagreed on the definitions or process? What caused that ambiguity? What could you do next time to avoid it? If your team faced no disagreements, explain how you worked together to build the requirements to avoid them.

Yes, I encountered some situations where I and my teammates disagreed on the definitions or process. For example, we had different opinions about whether to add the system as an independent class in the conceptual diagram. We had seen different examples where some included the system itself as a class but some did not, so we were confused about that. In order to deal with that, I went to ask the TA, and we applied the suggestion from the TA, which is including a class named “system data” in the conceptual model. Next time if we have disagreements again, we will analyze carefully and asking the TAs for help when in need.

4. Final Spec & Testing

a. What kinds of assumptions will be necessary to ensure that meeting the specification (verification) means that you have also met the requirements (validation)?

There can be some assumptions necessary to ensure that meeting the specification means that we have also met the requirements. They are like some details not specified in requirements documents, but they are important in order to make specifications and requirements consistent. For example, let us say there is a word limit for “Enter personal and academic details”, and the assumption will be the prospective TA enters no more words than the limit. If the content he wants to enter is way more than the limit, he cannot indicate the complete details of his personal and academic information, which will influence the whole process of assigning TAs in the future. Also, the budget to be entered into the system is supposed to be the same as what is obtained from the dean office. Otherwise it is impossible to assign a right number of TAs in the end, as the available budget set in the system does not match the real amount. This is another instance of assumptions that is necessary to make when trying to ensure meeting the specification means that we have also met the requirements.

b. How did you translate the requirements to specification?

When translating the requirements to specification, we changed the language making it more professional. Since the target readers for the system specifications should be engineers, and they can understand the meanings by more scientific words, while general users without

engineering backgrounds can only read normal English expressions. What is more, we tried to combine some user requirements into one big title of specifications before translating the language, as some small points are connected together tightly, and it would be easy to learn if they are listed together.

c. What was the most difficult aspect of writing your test cases?

Actually, the biggest problem for us was that we thought too carefully. When we continued writing test cases according to our specifications, we often found that it was possible to add some more original cases in order to make the current test cases more practical. For instance, when we considered test cases that verify some data was correct, we thought adding a test case verifying manually inputting data is kind of necessary. That way we kept adding more cases, and the result was we had too many cases. After the professor's explanation on Monday's lecture, we realized that some cases are not so useful, so we deleted many cases in the end.

d. Did writing your test cases help you find issues in your requirements? If so, what did you find and how did test case writing help you find/fix it? If not, what kinds of work effort (man-hours/financial) is going to be necessary to implement your tests? Explain.

To be honest, there were not obvious issues in our requirements when we go through the test cases. From my point of view, the test cases are necessary to be more detailed and cover more things than the original user requirements. Test cases are designed to verify if all the basic actions the users can do are already included in the requirements. Additionally, we also should have some cases that help test some edge error cases, which are even not considered very specific in the user requirements, such as Assigning more than 2 courses to TA should be an error.

e. How brittle are your tests? What kinds of changes could break them (provide concrete examples)?

I think our tests are pretty reasonable and connected with each other in some way, building a solid foundation in general. However, many cases depend on some basic cases, and changes of basic cases may affect them obviously. For example, the login test case is the first test case and it is depended on by almost all other cases. Whenever it is not verified at first or we make any changes to it, a large amount of cases will be influenced, which is not something easy to handle.

5. Misc.

a. What were the most difficult parts of writing the SRS (including Assignments 1-3 in your thoughts)?

I think the most difficult parts of writing the SRS is the first assignment. I had no determined idea about how to create some requirements, and there was so much information from the users to consider. I was supposed to understand all of them and assign them clearly to different types. I think the biggest challenge is usually the very first step, no matter what problems we are trying to solve. After we come up with the first version or general shape of the solution, the improvements in need will be easier afterwards. In our project, the diagrams and system specifications for the second and third assignments are relatively easy to realized, as we could change and improve the finished complete user requirements to get them.

b. What additional experience or training would be helpful for next time?

Well, since there are two other group assignments, I think it is important to learn from the experience we get from the past two times and try to make better communications and work together more efficiently. In general, we have good experience when work as a team, and we can share ideas during the discussions. Four types of thoughts gather to produce more possibilities. I will say we can cooperate better in the future and make our project more qualified.

c. How sure are you that the specification you wrote meets the requirements? If you are confident that it does, what specifically gives you that confidence? If you are not confident, what would you do to fix that, and how do you know when to stop?

I am pretty sure the specification we wrote meets the requirements in general, as we literally translated them one point by one point. Based on keeping all the principal meanings, we reworded them and made them specific enough for the system.

d. What are the greatest risks in your SRS? What are the likely things that will go wrong? What makes them risky?

To be honest I do not think there are obvious risks in my SRS. All of them resulted from team members' carefully thinking and continuous improvements, which cost a large number of hours. We tried our best to match different parts and make the overall document consistent. If I have to say something that might have risks, it would be details. I believe it is the details that make things different. Details can make things perfect, and they can also reveal some disadvantages to some degree. There may still be some details in some parts that we did not pay enough attention to, such as test cases. If we fix them, our project will be even better.