5 Requirements Management

It is a well-known fact that for a long while lack of proper requirement management has been one of the main factors for the failure of software and information technology projects. This factor still plays a great role, not as much in the failureas previously it was doing, but in the success of the projects. Simply,understanding software and information technology projects means that the project owner and the project developer have both agreed on certain requirements that should be met when the project is finished. In reality, two main sides of the project, the project owner and developers, would agree on some sort of contract within which they precisely articulate what the owner want and what the developer should do. The similar situation would apply to your final year project. The school/department is one side, as the project owner, and you are the other side, as the project developer.

To manage the requirements of your project you have to articulate the project needs in a formal and precise way. I assume that you have received proper education on the requirements management and requirements categories. In addition, I assume you have learned different techniques on how to catch the requirements, how to put them under specific category, and how to use them during the further stages of software development. Therefore, I am not intended in repeating those subjects in detail. Instead, in the following sections I would briefly explain a customized approach to the requirements management that suits the final year projects more properly.

5.1 Requirement Specification

To simplify, we can define a requirement as a brief explanation, mostly in one sentence, by which we explain something that we want a system to be able to accomplish. This explanation should be as precise and free of ambiguity as possible. A system may have several to several hundred requirements. However, when the number of requirements rises, usually, the system should be broken down into the smaller subsystems, each of which aims to address and fulfill a subset of the requirements list. As we discussed in chapter 3, "Requirement Management" is normally a major task of our project plan.

Although requirement specification is very important in all kind of projects, but in your final year project, it becomes even more crucial. The reason is in other projects, sometimes, there are some ways of compensation and/or boosting the development process. It can happen by asking for more time, or more budgets, or more human resources. Unfortunately, none of these can be applied to your final year project. You are totally on your own; academic period cannot be extended (forget about cumbersome process of appeals for a week or so extension!) and no one can be added to your project. Therefore, it is highly crucial for you to know whatyou are going to do. Below you can find some examples of good and poor requirement specifications.

Requirement specification examples – poorly specified

Sample 1:

- The project aims to develop a system for the university library, which fulfills the following requirements:
    - To keep and maintain all library records.
    - To keep track the borrowing and returning material.
    - To keep and maintain library members information.
    - To provide different reports.

Sample 2:

- The project aims to develop a speech recognition system for a specific language. The system should be able to do the following:
    - Understand different speeches.
    - Show the spoken sentences in a word processor.

None of the above samples talks about the level of the functionalities. Indeed, they are vague and interpretable. For instance, there is no specificationfor the period and numbers of borrow and return logs. Again, "To provide different reports", in the first sample, is a very broad requirement that whatever you prepare as a report, one still can say that there is a report that you have not prepared. In case of the second sample, the first requirement is too broad. It is not clear that what "different speeches" means. Again, the second requirement does not specify any word processors. Should it be Microsoft Word, NeoOffice, OpenOffice, or Apple i-Work word processor? To overcome with these problems you should work on the requirements to make them as specific and accurate as possible. This way the requirements can be quantifiable, manageable, testable, and traceable. You can find the revised version of the above samples on the next page.

Requirement specification examples – revised format

Sample 1:

- The project aims to develop a system for the library, which fulfills the following requirements:
    - To create and maintain library catalog based on Dewey decimal system.
    - To record borrowing and returning material for at least on academic year.
    - To keep and maintain library members information. The members should be classified base on their position i.e. undergraduate, Master students, PhD students, academic staff, and administration staff.
    - To provide reports on late returns, each member's status, library catalog, and a specific material status.
    - To maintain and handle a reservation list in first asked / first served manner.
    - To notify members on late submission via member's email, automatically.
    - To notify members on the availability of their reservation upon returning reserved material via member's email, automatically.
    - To notify members on late submission via members mobile phone, automatically.
    - To notify members on the availability of their reservation upon returning reserved material via member's mobile phone, automatically.

Sample 2:

- The project aims to develop a speech recognition system for a specific language. The system should be able to do the following:
    - Understand dialect A and B of the language.
    - Understand a mature female and male voice.
    - Show the spoken sentences in OpenOffice word processor.
    - Understand academic vocabulary with an error rate less than 20 words per 500 words page.

5.2 Requirement Types

Several models and approaches have been designed and introduced for the requirement management during the past decades in software industry. Almost all of these models are trying to help developers and users to understand the nature of requirements and to categorize which in such a way that guides the project to produce a system, which maintains an agreed (standard) level of quality. One well-known model for classifying requirements is called FURPS+, which stands for:

- **F**unctionality
- **U**sability
- **R**eliability
- **P**erformance
- **S**upportability

In addition, the "+" has been added to extend the classification to cover other areas such as Design, Implementation, and Physical requirements. In the real-life projects, it is important to consider all of these categories and to articulate each one, deliberately, in your project plan and other related project documents. However, neither in real-life nor in your final year project all of these subjects weigh the same weight. In fact, in different projects the weight of each class might change according to the nature of the system. In your final year project, though, the Functionality would play the major role. Hence, you should pay more attention to this class of the requirements. The rest is up to you and your supervisor. To my opinion, if your system is a general-purpose one, then you can classify all the rest as Non-functional requirements. However, be aware that there are final year projects, which may have the Performance as there heavyweight class within the requirement classifications.

5.3 Functional Requirements

Functional requirements are those features of the system by which the system fulfills the core responsibility that is expected to accomplish. You will see some examples of functional requirements for the proposed samples in this chapter in the following sections. Although this is the focus area in almost all software projects, however, it becomes more crucial in many final year projects. The reason is, although non-functional requirements are plying a great role in software systems, however, you are expected to show that your final year project product is able to do its core responsibility as its main concern, and then you have to show that you have considered non-functional requirements as well.

5.4 Requirement Prioritization

The main purpose of the requirement prioritization is to let developers and customers to make an agreement on the sequences of the implementation process. What are the core functionalities without which the system cannot be utilized? What is the sequence of dependency between different features? It is quite usual, even necessary, to ask these questions and many more during the requirement management phase. There are many reasons behind this. For example, these questions and their proper answers help developers to plan for the iterative and incremental development. They let customers to identify their core functionalities.They help both sides to focus on the core and not to assign their resources to the surface problems.

Several models and techniques have been produced to help developers to prioritize project requirements. There are even automated tools that help developers in this regard. However, like other concepts of development in this book, I am not going to repeat what you can find elsewhere in the related resources. You can refer to the bibliography section at the end of the book or simply have a search in your university library or even much simple than that to go online and you can find more than what you want about the concept. I want to discuss the issue in the context of your final year project. Notwithstanding, the approach is applicable to many other small-scale real-lifeprojects as well.

Prioritize the functional requirements based on two factors, their importance and their sequence. For the importance, you can use the following labels:

Requirements Priorities

Must have – label with this those requirements that together create the core functionalities of your system without which your system would not provide the minimum capability that your customer expects.

Should have – label with this those requirements that although are not part of the core functionalities of your system, however, they can be very useful and you would try to implement them if the project schedule allows you to do so.

Nice to have – label with this those requirements that they do not play a main role in the functionality of the system, at least at the scope of this project; however, they would give more facilities to the users if the schedule of the project allows them to be implemented.

In addition, assign a sequence number to the requirements. You can use two different styles, either give a prefix letter to each category i.e. M to "Must have", S to "Should have", and N to "Nice to have" and start over from one in each category or simply start from one and step forward. Which method you select does not matter, what matters is to choose a sequence that shows the route of your development. A good practice would be to organize the related requirements in such a way that shows the system's overall structure. Table 5-1and Table 5-2 show the above requirement specifications for sample 1 and sample 2, respectively, on which this method has been applied.

ID	Requirement	Importance
1	To create and maintain library catalog based on Dewey decimal system.	Must
2	To record borrowing and returning material for at least on academic year.	Must
3	To keep and maintain library members information. The members should be classified base on their position i.e. undergraduate, Master students, PhD students, academic staff, and administration staff.	Must
4	To provide reports on late returns, each member's status, library catalog, and a specific material status.	Must
5	To maintain and handle a reservation list in first asked / first served manner.	Should
6	To notify members on late submission via member's email, automatically.	Should
7	To notify members on the availability of their reservation upon returning reserved material via member's email, automatically.	should
8	To notify members on late submission via members mobile phone, automatically.	Nice
9	To notify members on the availability of their reservation upon returning reserved material via member's mobile phone, automatically.	Nice

Table 5-1 Requirement Prioritization (sample 1)

ID	Requirement	Importance
1	Understand dialect A and B of the language.	Must
2	Understand a mature female and male voice.	Must
3	Show the spoken sentences in OpenOffice word processor.	Must
4	Understand academic vocabulary with anerror-rate less than 20 words per 500 words page.	Must

Table 5-2 Requirement Prioritization (sample 2)

As you can see, Table 5-1 includes requirements with different level of importance of whichthe first 4 is your obligation, and you can do the rest according to the rules that were discussed. But, in the second sample, Table 5-2, all requirements must be met.

5.5 Summary

A key parameter to your success in your final year project is to understand what you are going to do. In other words, you have to specify the project's requirements. This importance has been reflected in the Software Development Life Cycle (SDLC) by defining one of the early stages of development cycles asRequirement Management. However, this is not enough for your success that you have understood the project specification, rather, you have to precisely articulate this understanding and to get an approval on it from your supervisor.

Furthermore, you can determine the type of requirements by simply categorizing them under categories, such as Functional Requirements, and Non-functional requirements or you can use a bit more detailed approach by using other models such as FURPS+ model in order to make your requirement specification more accurate and precise. However, the sizes of a final year projects in the majority of cases are not suggesting that that requirement specification should be deeply detailed. Instead, an optimized approach based on the two main categories should suffice.

Finally, the prioritization of requirements can be done by following a simple classification as "Must have", "Should have", and "Nice to have" to specify the degree of the importance of each requirement. "Must Have" labeled requirements are your obligatory, and you can implement as many requirements of the second and third category as the schedule lets you to do so. Otherwise, you can talk about them under the "Extendable Features" or "Further Research" sections in your project report.