# Contents

# Figures

# Tables

# About the Author

Hossein Hassani is a lecturer at the University of Kurdistan-Hawler since 2007. He joined UKH after nearly twenty years of experience in software industry. He has been teaching different modules such as Project Management, Advanced Database, Software Engineering, Object Oriented Programming, Management Information Systems, and Human Computer Interaction.

Furthermore, He has taught some other courses such as Fundamentals of Programming Languages at other universities. In addition, he has mentored a group of postgraduate students as teacher assistants of software engineering laboratory for undergraduate programs in software engineering.

During his experience in software industry, he has worked in different positions starting from a junior programmer, promoting systematically to a senior programmer, then a designer, an analyst, a team leader, a project manager and finally a senior consultant. Furthermore, during this period he kept his relation with higher education institutions and teaching activities through providing different seminars and intensive courses to the audience of experts in the field. This long path of experience alongside diversity of the projects in which he has been involved and committed have enabled him to have a deep understanding of software and information technology projects, as well asgiving him a holistic idea about the dynamic relations of computing with all aspects of humanity and social life. He shares his findings of this amazing journey with his students during lecturing and teaching.

He is interested in Information Accessibility, Computational Linguistics, and Software Quality Assurance.

# Preface

Every year thousands and thousands of students who are studying Computer Science and Information technology have to do their final year projects, one way or another. Most of these students are thinking about the title, methodology, tools, and other related issues regarding their final year projects. However, some do not paying enough attention until it becomes too late and then they are struggling with the probable not so interesting consequences, such as late delivery, low quality standards, and sometimes walking around with confusion.

Although the role of supervision is very important in the final year projects, however, because of the diversity in the supervisors' background and the limitations of the times that they can dedicate to their students, it would be useful to provide students with a practical guideline that they can follow in order to make the mission possible!

During several years of experience in supervising final year projects and dealing with students with the different levels of knowledge and skills that are required to accomplish their final year project, I have faced diverse cases, which showed me that this is necessary to prepare a practical guideline based on which students can conduct their projects. Having a practical guideline which is able to answer their general issues, students can utilize the meeting times with their supervisors to gain more in deep advice regarding their specific projects, instead of using it to resolve primitive issues.

Although the book mainly targets the undergraduate students, however, postgraduate students can use it as a general guideline to accomplish their projects as well.

The book has 12 chapters, which explainthe whole roadmap for accomplishing a final year project. In addition, three appendices have been provided to show samples of project plans, project diaries, and supervision meetings.

The author would be grateful if he receives feedbacks from colleagues who have dedicated some time to review the book and students who have used the book during their final year project. You can reach the author at hosseinh@ukh.ac and h.hossein@rgu.ac.uk.

Hossein Hassani
Hawler
October 2012

# 1 Introduction

The final year projects are considered as one of the core modules for computer science and information technology studiesat the undergraduate levels. Although the credit that this module carries is different from school to school, in different countries, however, all of them agree on the paramount role that the module plays in forming the students' mindset towards performing real life projects. In addition, the final year project aims to let the students to combine and utilize almost all core modules/courses they have studied during their undergraduate journey. This is the reason for paying so much attention to the final year project as a crucial assessment.

Regardless of different approaches in computer science and IT education at different universities, the main theme of the final year projects is to implement an idea using software development tools. Although the projects are mostly individual projects, however, there are cases in which group projects are considered as final year projects. Normally, students receive different training and knowledge in order to accomplish small projects through their assignments in different modules/courses, but these assignments, mainly, focus on specific topics such as database and/or programming coursework. However, when it comes to the final year project, it is considered to be more comprehensive and to cover and utilize most the knowledge that students have already been obtained.

In fact, when students start their final year projects it is assumed that they have received the fundamental knowledge that they need in order to sufficiently conduct their projects and other final year modules/courses should be considered supplementary to the project implementation. Despite the fact, when it comes to the action point, students face many issues, such as:

- What is a proper topic that I can choose?
- How can I conduct my project?
- What is the methodology and which one is good enough for my project?
- What are the stages that I should go through?
- What kind of tools should I use?
- How can I prepare my report?
- What kind of documents should I submit?

Although, some of these questions might have been answered within the guidelines that faculties/departments/schools provide students with, but in reality, many of students ask their lecturers/professors these questions again. The following chapters aim to answer these questions and to show students how to tackle their final year projects. The discussion would start by showing how to choose a project topic and continueon how to conduct the project.

# 2 How to Choose a Project

To choose a proper topic is the first step in doing your project. Usually, according to their procedures, the school/department,where you are studying, publishes a list of project topics at the beginning of every semester/year. Different schools, however, follow different approaches in publishing this list. Some publish the list, which only includes the project topics, some others allow every supervisor to publish their own list, some publish the supervisors list alongside their related projects, and yet some publish the list with a brief explanation of the problem area.

Regardless of whichever of these approaches is the case, almost all schools give an opportunity to the students to suggest their own ideas and topics for their final year projects. However, whether the schools accept the suggestion or not depends on the criteria that they consider in their proposal acceptance process. In the following sections, the project selection task will be discussed in detail. In this discussion two main cases, namely, to choose form a list or to suggest your own topic, will be explained. In addition, a method will be provided that enables you to choose your project objectively and to eliminate the subjectivity as much as possible. Moreover, although the topic selection is the main theme of this chapter, a section on choosing a supervisor has been provided for thesituations that it is applicable.

## 2.1 Choosing a Topicfrom the Project List

Like many other activities, usually, the first stepsof a project are very crucial. In your final year project, the first step is to choose your project topic. It is very important to dedicate a proper amount of time in order to make your decision. Bear in mind that you have to live with your project for a semester or in most cases for a year, so it is better to choose a topic, which is truly of your interest. Equally important, do not forget that you should be wise enough to choose a topic that you can accomplish. Therefore, you should be as honest and frank with yourself as possible.

Sometimes, the supervisor who is supposed to supervise you may know your background and advise you to do not take the project that you have selected and choose another one instead. In the majority of cases, it is better to take this kind of advice. Indeed, if you could not strongly defend your suggestion and to convince your supervisor that you would be able to accomplish the project, it would be much more wise and safe if you reconsider your proposal and change your topic.

Below you can find a general guideline that you can follow. Experience has shown that it works fine.

**Tips on choosing a project topic**

- Study the topics carefully and thoroughly. This is important because if you miss a topic that can be suitable for you, one of your classmates may take it. Although this is not a competition per se, but the projects assignment cannot avoid of applying some sort of prioritization procedures.
- Consider the harmony between the heart and the brain!
- Choose a topic that is attractive but at the same time you think that it is doable. This is important to understand that no matter how much the topic is attractive at the end of the day if you do not deliver its minimum requirement you cannot receive a pass.
- Choose among the topics of a supervisor(s) with whom you feel that you can comfortably communicate.
- Choose more than one topic to have more flexibility; three is a good number, then!

Before having further discussions on the case, it is worth it to notice that you can decide on your final project, intuitively, which means without investigating any fact and factor and simply by following your desires and feelings. However, you will be lucky if it comes out as a right decision. Nevertheless, if you do not want to take this risk,and then follow a systematic approach to make your decision. In order to show youhow to follow a systematic approach to select your final year project, let us consider an example.

Suppose that you have chosen three project topics, among 30 ones that are availableand you have prepared a short list of which, without any prioritization, as below:

1. An Speech to Text System
2. AnAsset Management System
3. A SMS Cryptographer

Assuming that all three are open to you to choose from, how can you prioritize this short-list? Obviously, the above selection shows a very broad boundary of desire. In fact, they have been deliberately selected to be so in order to show you the important parameters that you should take into account when you make your decision. Moreover, later in this section, you will find out that each one of thoseprojects needs different levels of skills and requires knowledge of different areas. Now, you have to decide how to prioritize this short list.

First, you should consider that these projects need different levels of programming and technical skills. In addition, they need different levels of knowledge ofother subject areas. While all of them need good programming skills, certainly higher level of programming challenges would be anticipated for the first and the third topics. On the other hand, the second topic needs good knowledge of business system analysis and database applications. Moreover, the third topic needs some knowledgeof mobile computing and applications. Again, this topic needs some background on cryptography, while the first topic needs a good understanding on the speech related technologies. Now, we go back to the question that we had at the beginning of this paragraph, "How can you make your final decision?"

In order to answer this question you have to measure each project's "suitability " degree to you. In order to do so a method will be introduced that helps in this measurement process. To keep this measurement method simple, we assign some measurable parameters to each project and we call them selection parameters. Again, for the sake of simplicity and applicability, we summarize these selection parameters into three categories, namely, Interesting, Background Knowledge, and Required Skills. The parameters will be described shortly. In addition, to measure each parameter, we assign each one two attributes, namely a Coefficient and a Percentage. As a result, we have a proper foundation based on which we can apply our method.

Now, to measure each project's "suitability", you should assign a percentage to the Degree parameters to show each parameter's value. The Coefficient is a fix number, from 1 to 5, which shows the weight of the parameter, or its importance, if you prefer. The Coefficient must remain the same for all projects that you have selected. Table 2-1 shows atemplate based on which you can quantify the situation of each project in the short list based on the mentioned parameters.

| Parameter / Projects | Interesting | | | Background Knowledge | | | Required Skills | | | Final Result |
|---|---|---|---|---|---|---|---|---|---|---|
| | Coif. 1–5 | Degree 0–100 | Result | Coif. 1–5 | Degree 0–100 | Result | Coif. 1–5 | Degree 0–100 | Result | |
| Speech to Text System | | | | | | | | | | |
| Asset Management System | | | | | | | | | | |
| SMS Cryptographer | | | | | | | | | | |

**Table 2-1** Project Prioritization Template

As an example, suppose we have set the Coefficient to be 3 for the Interesting, 4 for the Background Knowledge, and 3 for the Required Skills. In addition, suppose we have set our estimation for the Degree parameter as well. Table 2-2 shows this situation and the calculated result for each parameter per each project. The last column, Final Result, shows the outcome. According to this example, the prioritized list of projects would be as below: An Asset Management System

1. A SMS Cryptographer
2. An Speech to Text System
3. An Speech to Text System

| Parameter / Projects | Interesting | | | Background Knowledge | | | Required Skills | | | Final Result |
|---|---|---|---|---|---|---|---|---|---|---|
| | Coif. 1–5 | Degree 0–100 | Result | Coif. 1–5 | Degree 0–100 | Result | Coif. 1–5 | Degree 0–100 | Result | |
| Speech to Text System | 3 | 50 | 150 | 4 | 20 | 80 | 3 | 30 | 90 | 320 |
| Asset Management System | 3 | 60 | 180 | 4 | 40 | 160 | 3 | 60 | 180 | 520 |
| SMS Cryptographer | 3 | 70 | 210 | 4 | 30 | 120 | 3 | 40 | 120 | 450 |

**Table 2-2** Project Prioritization Sample

Consequently, the revised list should be considered as your guide in order to choose your project topic. However, sometimes we might not like the outcome! Well, this is the reason for the measurement process. Indeed, it prevents any subjective decision and shows you the factual and objective outcome. Nevertheless, the harmony between the heart and the brain, as it was mentioned earlier, is very important. But, instead of trying to change the result mechanically, be careful when you are providing the data for each parameter in the first place and do accept the result, afterwards. Anyway, if you could not fix a project topic this way, then read the next section to find out how you can propose your won idea.

## 2.2 Proposing Your Own Topic

There are situations that you cannot find a proper project in the school's proposed list. You may face this situation because of different reasons. For instance, it might be because you are simply not satisfied with the proposed topics, or it might be because all the topics, which are of your interest, have already been taken by your classmates. Whatever the reason is, you should either select a project that does not suit you or propose your own topic(s). The farmer choice would not be awise one unless you are forced to do so. If that is the case then you should apply the same method that was introduced to you in the previous section. However, in this new situation you are going to select among the closest topics that still are open to be selected.

If you have to propose your own idea, it would be better to think about more than one topic; again, three would be a magic number. This time, the situation can be more difficult than the normal circumstances that we discussed, earlier. In fact, you need to use your creativity talents to come up with proper ideas. However, the general guidelines below can help you to find your way:

**Tips on proposing your own topic**

- Be creative. Look around at the school, university, your hometown, province, state, and country and try to find a subject, which has not been automated.
- Look at new technologies, specifically those that are related to mobile computing and web technology. You can find many areas that these technologies can be applied for the first time. Clearly, this situation is more popular in developing countries and regions.
- Focus on the subjects that you like their status to be improved using automated systems and computers. These subjects can be selected from different origins such as education, health, globalization, tourism, global warming, life style, culture, entertainment (specifically gaming), etc.
- Choose three topics in order to provide your school and yourself proper flexibility with your proposals.
- Write a couple of paragraphs that state the problem area and why you think this topic is important to be considered as a final year project. The writing process provides you an opportunity to rethink about the topics and to establish the grounds based on which you can defend your proposals.

As soon as you prepared your short list, apply the method that we discussed, earlier, to prioritize the topics. Next, provide the list to the proper authority, who is supposed to decide on your topics. Normally, in this situation, you have to talk to your prospect supervisor. As it was mentioned in the previous section, although you need to defend your proposal strongly, however, it is very important to listen to your supervisor carefully, and to take her/his advice on your proposals seriously.

## 2.3 Choosing a Supervisor

Some schools allow you to choose your supervisor, but some do not. If you do not have the opportunity to choose your supervisor, then maybe you can simply skip this section, however, even if that is the case, you still can get some good advice regarding supervisors in this section. The supervisor's responsibility is explained by most of the guidelines that schools prepare and disseminateregarding the final year projects. Usually, these guidelines properly inform the students that their supervisors do not provide them with solutions but the general roadmap to the solution.

Moreover, the guidelines advise them about the main issues of the project. Nevertheless, students expect to haveat least some hints about their projects during different stages of the project and to receive some feedbacks, which are specifically addressing different aspects of the selected topic. However, students are told that the project is their responsibility. It means that they have to accept all the consequences of not attending to the supervisor meetings. Again, this is the student responsibility, to seek and follow their supervisor feedbacks on their project according to a preplanned timetable.

Nevertheless, supervisors can have a significant impact on your project. Many factors participate in this impact such as supervisor's experience in the project area, her/his experience on the supervision, the time she/he can dedicate to each student, her/his interest in the project topic, her/his personality and communication style. However, it is not always possible to choose your favorite supervisor. This is because sometimes the supervisor that can help you more does not propose the topics of your interest, or because all the topics that she/he has proposed have already been assigned to your classmates. Below you can find some general hints regarding supervisor selection.

**Tips on choosing a supervisor**

Choose among the topics of a supervisor with whom you feel you can comfortably communicate. However, you should consider the topics based on the topic selection, at the first step (refer to section 2.1 and 2.2).

- Choose a supervisor that has the knowledge of the area of the chosen topic.
- Consult with graduate students about their experiences on different supervisors.
- Choose more than one supervisor and prioritize your list.
- Talk to the prospect supervisors as soon as you can and register your name with them.
- If you have selected more than one supervisor, let them be informed that you have done so.

## 2.4 Summary

The first step to do the final year project is to select a proper topic. In this chapter, the importance of this step was discussed. In addition, possible situations for this step were identified. It was explained that either you can find a topic within the school'sproposed project list or you have to propose your own idea. Whichever the case is, a method to quantify the topic's "suitability" was presented. In addition, an example was presented to show you how to apply the mentioned method. Furthermore, the case of supervisor selection was discussed and some guidelines regarding it were presented.

# 3 Project Planning

The next step, after you selected your project topic, is project planning. Project planning subject is taught, one way or another, at different schools. Some address it during teaching thesoftware engineering module/course and some others teach it as an individual module/course. Studentsmight even have some experience of planning through their previousindividual/group assignments. Despite the fact, it is quite a common phenomenon among the students to undermine this step extensively, which cause them to submit incomplete projects or face late submission.

Actually, in software engineering and development, project planning is intertwined into the methodology in a way that sometimes it is difficult to separate these subjects from each other. However, to understand the whole roadmap of conducting a project, these two concepts would be discussed in two different chapters, though the relation between the two would be maintained and addressed wherever appropriate. Having this, in the current chapter we will focus on the planning step, then we will discuss the methodology and its influence, and impacts on the project planin the next chapter.

To understand how to prepare a plan for your project, let us considerthe situation that you are standing in. You are in the final year of your study; you have several modules to study, you have other assignments and you have to pass several other exams; and finally you have to do your project as well. Now, other duties that you have would normally be scheduled by the university and/or the school. In fact, you are obliged to stick with the timetables that the university/school enforces, and therefore, the degree of freedom is not totally under your control. However, the scheduling and controlling your project is at your disposal. As a result, this situation makes the project planning of paramount importance in the whole process.

Obviously, lack of planning and being careless about schedulingcan create an irreversible catastrophic circumstance that puts your entire project in a hazardous situation. The dangerous outcomes can be very different. The extent can be from low quality project delivery, at one extreme, to a total failure, at the other. Now, how can you prevent this unwanted scenario? How can you guarantee that your project would provide the requirements to a pass, as a minimum achievement? How can you turn it to a real success then? Indeed, taking simple steps can help you to overcome with this situation, properly.

To illustrate, we begin with the assumption that you have already studied some project management techniques, therefore, you have the required knowledge of project planning and scheduling. If this is the case, then presumably, you already have familiarity with software development techniques, as well. Consequently, you are required to apply the knowledge that you have already obtained, efficiently. As a first action, if you are already familiar with the planning concepts and using Gantt charts as a scheduling tool then consider the following steps:

**Main phases of a project plan**

- Using one of the tools, such as Microsoft Project, Microsoft Visio, and OpenProj, etc. that can help you in preparing a Gantt chart, devise a first-cut plan including the following major steps:
  - Understanding the problem area
  - Literature Review
  - Requirements Management
  - Analysis
  - Design
  - Implementation
  - Test
  - Project Report
  - Presentation

If you are not familiar with the Gantt chart, which was mentioned in the previous step, and have not experienced the tools that you can use to createone at this stage,do not worry because these concepts would be presented in the following sections. However, although we are not going to discuss these concepts in a very detailed format, you can find general explanations that canhelp you plan and manage your projects, effectively.

## 3.1 Project Plan

Project plan in a simple format is a document that explains 4Ws+H, which means **W**hat is going to bedone, **W**hen it is going to be done, **W**ho is going to do it, **W**here this is going to be done, and **H**ow it is going to be done. This concept has been rephrased again as below:

**Questions about Project Plan**

- What are the main steps that the project goes through?
- What is the project timeline (i.e. what happens when)?
- What are the deliverables that the project should deliver?
- What are the quality criteria?
- What are the acceptance criteria?
- How the project would be conducted?

However, although the above topics are typical for all projects, every specific project has its own specialties, which makes its project planning unique.

Many students underestimate this activity. They think that this is an individual activity that is totally under their control; hence, there is no need for wasting time on this somehow "superficial" and "mechanical" activity. As it was stated before, if you do not know how you are going to do your project, it is highly likely to fail doing it.

At the first step, the plan seems to be very vague to you. At this stage, do not worry about the details. Simply, think about the major steps that you should take and make a list that depicts those major steps, which should be taken. Look at the following example to find out how it can be done. As you can realize these may slightly be different in your case, however, the main theme remains similar. This task – making a list by breaking down the whole job into specific manageable tasks – is called Work Breakdown and the result of which is called Work Breakdown Structure or simply (WBS).

**Work Breakdown Structure (WBS) sample**

- Preparing Problem Statement
- Understanding general requirements
- Meeting with my supervisor
- Literature Review
- Implementation
- Testing
- Writing Project report
- Preparing Presentation

Now, this list needs two important things, which should be addedto it in order to make it a plan: timing and resourcing. By timing, I mean, you have to say when the task is expected to be started and when it is expected to be accomplished (finished). By resourcing, I mean the human beings and any material/equipment that any specific task of the above list needs to have in order to be done. In fact, in your specific project, the main human resource of the project is you. However, sometimes you need other resources such as your supervisor. In terms of material/equipment you may need a computer (which you normally have one) and in some cases specific devices or software that should be obtained before the task is started.

Having these items ready, it is good to know that there are two major approaches with adding these details to your plan and applying the required changes: top-down approach and bottom-up approach. As the names imply, and as you might have heard about these approaches in other courses/modules, if we follow the former approach we start from top task in the list and will apply resources and timings one by one until we reach to the end of the list. Whereas if we follow the latter approach, we should start from the last task and assign resources and timings to task backwardly.

There are different reasons for having these two approaches, which discussing them is beyond the scope of this book. However, if you are interested in having more details you can consult thebibliography section at the end of the book. Nevertheless, in your case the bottom-up approach is the better choice than the top-down. The main reason for this is that the delivery time of your project is very restricted and normally bound to the specific university timetable for the assessment schedules. Hence, some major dates are not under your control and you are obliged to stick with a predefined university/school plan.

Let us apply the approach on the mentioned example. All you need is to have pen and paper, or a word processor, and a calendar. We assume that the project should be done within a semester. For the sake of simplicity, we assume the semester starts at October 1, and ends at January 27. Again, suppose you should submit your project by January 15 and you should be ready to present your project on January 22. Having this information, let us prepare a schedule based on which you can conduct your project.

Applying bottom-up approach implies that we should start from the last date, which is January 20 and set this date as the finish date for the last task, which is "Presentation". Then, we examine the mentioned task and we estimate a duration within which we think that the task is getting accomplished.We try to be as realistic as we can in this regard. However, you may ask how you should know that how long the task might take to get accomplished. In reality, there are different ways that help project plannerswith the time estimation process, which unfortunately most of them do not work for you and therefore we are not discussing them, here. In fact, you have to establish this version of your schedule by guessing the required time for accomplishing each task. Then, you have to refine it by consulting your supervisor, later.

Suppose that based on a rough estimation the presentation task needs you to spend 3 days in order to prepare it. Assuming the presentation day as January 22, you should have finished your presentation no later thanJanuary 21, which is exactly a day before your presentation day. This assumption sets the end date of the "Preparing Presentation" task on January 21 and therefore, the start date on January 19. However, for the time being assume that you are off at the weekends. Considering this, puts the start date of the task on January 17. By applying this technique, you can obtain the first version of your scheduling as you can see in Table 3-1.

| ID | Task | Duration | Start | End |
|---|---|---|---|---|
| 1 | Preparing Problem Statement | ? | ? | ? |
| 2 | Understanding general requirements | ? | ? | ? |
| 3 | Meeting with my supervisor | ? | ? | ? |
| 4 | Literature Review | ? | ? | ? |
| 5 | Implementation | ? | ? | ? |
| 6 | Testing | ? | ? | ? |
| 7 | Writing Project Report | ? | ? | ? |
| 8 | Preparing Presentation | 3 days | 01/17/13 | 01/21/13 |

**Table 3-1** Project Scheduling – Bottom Up – First Attempt

As you can see, we have created a table including the first cut tasks that we already recognized and calculated the duration, start date, and end date for the last task. We can calculate the start and end date for each task by applying the same technique. Let us do it for "Writing Project Report" task. Suppose that this task needs 10 days to be accomplished. However, later in this chapter you will see that this task should not be considered as a short-term task, rather a continuous task that is spanning throughout the entire project. Nevertheless, for the time being we put this issue aside.The result schedule will be asit appears in Table 3-2.

| ID | Task | Duration | Start | End |
|---|---|---|---|---|
| 1 | Preparing Problem Statement | ? | ? | ? |
| 2 | Understanding general requirements | ? | ? | ? |
| 3 | Meeting with my supervisor | ? | ? | ? |
| 4 | Literature Review | ? | ? | ? |
| 5 | Implementation | ? | ? | ? |
| 6 | Testing | ? | ? | ? |
| 7 | Writing Project Report | 10 days | 01/03/13 | 01/16/13 |
| 8 | Preparing Presentation | 3 days | 01/17/13 | 01/21/13 |

**Table 3-2** Project Scheduling – Bottom Up (Continued)

Finally, if we continue to apply this method we would end up to the first cut schedule for our sample project as you can find in Table 3-3.

| ID | Task | Duration | Start | End |
|---|---|---|---|---|
| 1 | Preparing Problem Statement | 3 days | 01/10/12 | 03/10/12 |
| 2 | Understanding general requirements | 12 days | 04/10/12 | 19/10/12 |
| 3 | Meeting with my supervisor | 3 days | 20/10/12 | 23/10/12 |
| 4 | Literature Review | 10 days | 24/10/12 | 06/11/12 |
| 5 | Implementation | 30 days | 07/11/12 | 18/12/12 |
| 6 | Testing | 10 days | 22/12/12 | 01/01/13 |
| 7 | Writing Project Report | 10 days | 01/03/13 | 01/16/13 |
| 8 | Preparing Presentation | 3 days | 01/17/13 | 01/21/13 |

**Table 3-3** Project Scheduling – Bottom Up (First Draft)

If you prepare a Gantt chart for Table 3-1, the outcome would become something such as what you can see in Figure 3-1.

| ID | Task |
|---|---|
| 1 | Preparing Problem Statement |
| 2 | Understanding general requirements |
| 3 | Meeting with my supervisor |
| 4 | Literature Review |
| 5 | Implementation |
| 6 | Testing |
| 7 | Writing Project Report |
| 8 | Preparing Presentation |

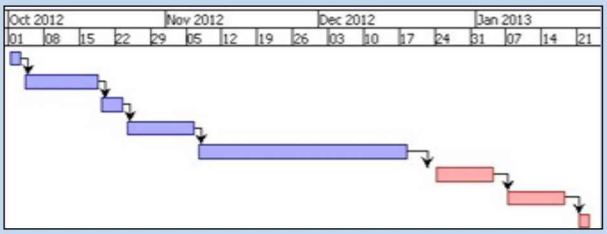


**Figure 3-1** Project Scheduling – Gantt chart (First Draft)

The arrows in Figure 3-1 shows how the defined tasks are related to each other in a way that each successive task starts when its predecessor finishes, which resembles a waterfall. However, in reality this is not the case. To illustrate, let us have a closer look at this Gantt chart. Clearly "Meeting with my supervisor" is not a task that you would do once and then you have done with it. Actually, this task is of a type that we can call it an "ongoing" task, which starts when you start your project and will finish right before you present your project.

Again, “Writing Project Report” task can begin very earlier, when you have done part of “Literature Review” task, for example, and can continue throughout the project up to the presentation time. During this time, you might send it to your supervisor, as a draft, in order to receive some feedbacks to improve the final version. Indeed, these instances show that this first cut project plan needs some revision to make it a plan that can be as applicable as the reality requiresit to be. Consequently, after preparing the first cut plan it should be reviewed and refined. This revision process would be discussed in the next section.

## 3.2 Refining the Project Plan

During the previous section, you prepared your first cut project plan. It was shown that this first cut version of the plan needed to be reviewed based on the characteristics of each task, in order to adapt it to the real situation. For example, the following tasks should start earlier:

- Implementation
- Testing
- Writing Project Report

Furthermore, you can do some tasks in parallel. Well, when we talk about parallelism do not forget that the concept should be understood in its context. Indeed, although it seems that these tasks are in parallel, but they are not. Because you have only one resource to perform these tasks, andit is you, yourself. Therefore, in this case, the situation resembles more to multitasking rather than parallelism. To understand it better, you can compare the situation with the multiprocessor and single processor concept that you might have seen in the Operating Systemscontext. A single processor in your computer does several jobs in a multitask format which seems to be done in parallel.

Below is a group of tasks that can be done in parallel:

- Meeting with my supervisor
- Literature Review

Again, this is another group:

- Implementation
- Testing
- Writing Project Report

However, do not forget that you cannot do the above tasks literally simultaneously. In fact, it means that if you have three tasks in parallel for a specific period, you are dividing your efforts and timing according to the nature of each task. Again, this means that you need to accomplish a portion of one task at a time, and then you perform another task, and then another one, until you cover all the tasks by swapping between them. Figure 3-2 shows the refined version of the Gantt chart of Figure 3-1. This new versions has been obtained through applying the techniques that were discussed in the current section.

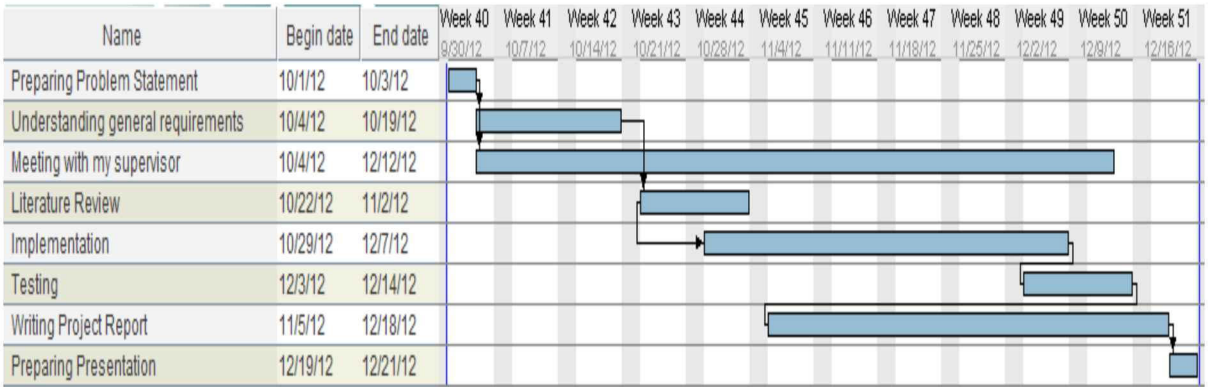


**Figure 3-2** Project Scheduling – Gantt chart (Revised)

In order to show that that the tool that you can use for scheduling does not have a great influence on the results, this new version has been prepared using GanttProject software. GanttProject is an open source software that can be used as one of the many tools that you can find to automate your project scheduling and control. However, bear in mind that using pen and paper, or simply a word processor, or a spreadsheet, you can still prepare, mange, and control your project. Obviously, this is the concept and techniques that is significant, not the tools.

Figure 3-2 shows how some tasks, which are taking place in parallel. In addition, it shows that it is not necessary to finish a task to start its redelated successor. You can observe this case for several tasks such as "Implementation", "Testing", and "Writing Project Report". Although managing several tasks simultaneously is not an easy job, however, you have been doing this for many years during your education, perhaps unconsciously. Actually, while you are studying several modules at the same time you are studying them in parallel. Similar methodis applied to the above parallel tasks, but this time it should be done completely consciously! The way that you can do it and control it would be discussed in the next section.

## 3.3 Controlling the Project

Project planning, as it was mentioned, is one of the early steps that you should take in order to pave the way to accomplish your project. However, having a good plan, solely, does not guarantee your success. Indeed, a good plan without other necessary activities cannot assure any project's success. To achieve your goals you need to control your project properly. To control your project means you have to evaluate your progress and to update your plan to reflect this progress, continuously. Moreover, you have to compare the progress obtained with the progress expected. The positive or negative deviation of the expected progress, or estimated progress if you like, can show if you are ahead or behindthe plan.

To illustrate, let us continue with the example that we had in the previous sections. Suppose that you are at the beginning of the project. You have to prepare the problem statement and you have considered that you should prepare it within 3 days. Again, suppose that based on your estimation it needs 10 hours to be done. Now, it is a good practice to specify how many hours you are going to dedicate to this task in each day of this task's 3 days period. However, if it seems too detailed and too cumbersome to you to do so, then simply evaluate the task in terms of percentage and evaluate your daily progress.Then write these evaluation and progress down on your schedule as a part of your Project Diary, which would be discussed in the next section. This evaluation allows you to examine your progress and to adapt your efforts based on its outcome.

For example, assume you have spent 4 hours on the task "Understanding general requirements" and you have had one meeting of the expected 10 meetings with your supervisor. To update your schedule means that you update the first task's progress by 40% and the second one by 10%. Figure 3-3 shows the resulted Gantt chart. There are different ways that you can show this progress and if you are interested in having more information on this, you can consult the bibliography at the end of the book. But, regardless of the presentation of the progress, this is important to show this progress, on way or another, preferably graphically, to understand where you are standing with regard to your project goals.

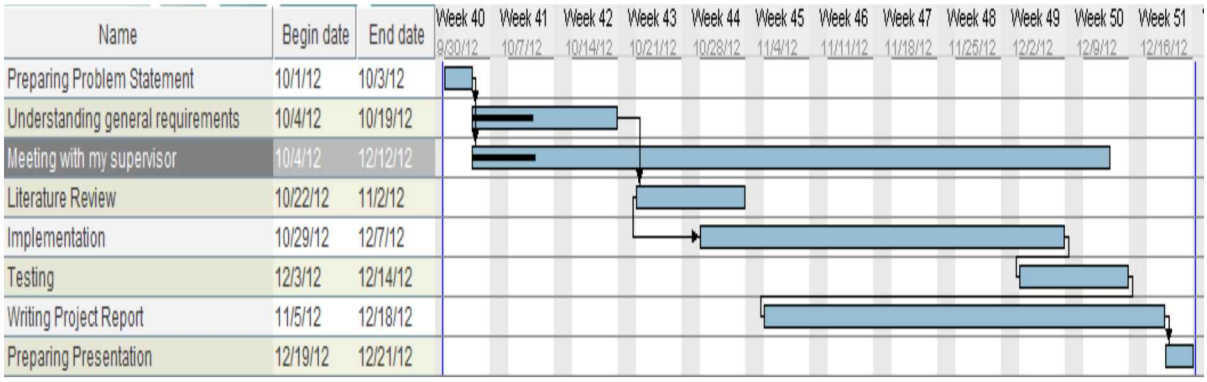


**Figure 3-3** Project Scheduling – Gantt chart showing tasks' progress

In addition, there is another key concept in the project control process that you should consider. This concept is the milestone concept. By definition, milestone is an event or a task that consumes no time in the project. OK, what does really it mean? Well, one can say that a "milestone" is a point or moment in a project that one expects a worthy event to happen which is the result of what have already been accomplished during the project. Usually, this event can be finalizing a document, receiving approval for something, and/or delivering some deliverable material, etc. Furthermore, this event would normally have a great influence on the other tasks of the project.

For instance, finalizing the literature review can be considered as a milestone, which means you have sent your literature review to your supervisor, then you have received her/his feedback, then you have applied changes, and finally you have received the supervisor's OK on this task. Figure 3-4 shows the Gantt chart of our hypothetical project on which this concept has been applied. If you have a close look at this figure, you can see that a new task "Literature Review Approved" has been inserted to the schedule below the "Literature Review" task. As you can see, a diamond-shape has been shown in front of the task on the Gantt chart's grid section, which is a usual way to show milestones.

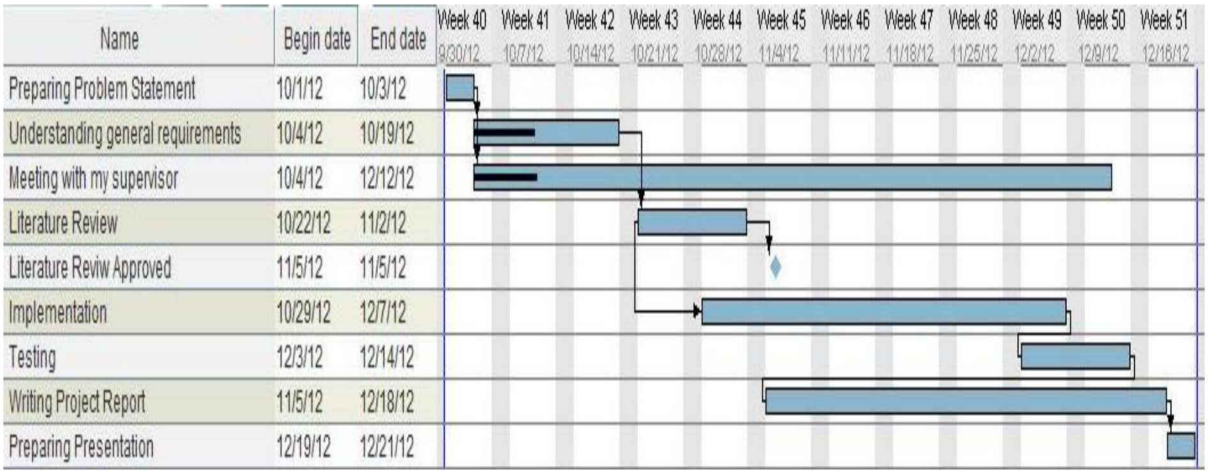


**Figure 3-4** Project Scheduling – Gantt chart with Milestone

## 3.4 Project Diary

Some of the schools/departments encourage students to keep a project diary for the project that they do. Indeed, in some cases they ask students to attach this diary as an attachment/appendix to their final reports on their projects. Regardless of whether this is an obligation to do or not, it would be a great habit and practice to keep a project diary for your project. There is no need to record everything in detail, but having simple notes on what you have done, when you have done it, and what have been the main problems/findings that you faced could be sufficient. As a rule of thumb, a short paragraph addressing the main items in bulleted format would be enough. Do not forget to mention the date and the specific task(s) that you are writing about and the overall time that have dedicated to each item.

| Item | Date | From (hh:mm) | To (hh:mm) | Description |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| …. | | | | |

**Table 3-4** Project Diary template

## 3.5 Summary

Project planning plays an important role in any project. Although your final year project is normally considered as an individual project, however, it is still crucial to have properly scheduled plan for it. Equally important, the final year project is usually considered as a fully individual activity. However, in this "individual activity" you need to deal with the other demands, such as other modules/courses that you have to study, your supervisor time, probablythe laboratory time and equipment that you might use. This situation requires you should to prepare an efficient program.

Using main steps, which are typical to information technologyand computing projects, you were advised to prepare your first-cut schedule. Afterwards, you were shown how to polish this draft version and to revise it to cover your specific situation. In addition, the concept of the parallel tasks (multitasks, actually!) and the way that you can manage them was discussed. Moreover, the concept of milestones and its importance was introduced. Milestone was defined as a point or a moment in a project that one expects a worthy event to happen. Again, you were advised to identify your project's milestones.

Finally, the project control process and the concept of keeping a project diary were explained. It was shown that project plan and its related schedule could not be useful if they are not updated, properly and regularly, in order to show the project's current situation in any specific moment. Keeping diary short and up to the point, whereas recording main items such as date, amount of effort, main findings/problems, and naming related tasks according to the schedule were presented as a good and helpful practice.

# 4 Methodology

Methodology seems to be a buzzword in the academic context. However, it is not as well understood and known in undergraduate as it is in postgraduate studies. Besides, because the term and its concept has widely been customized and adapted in software engineering and development, it becomes somehow difficult for students, especially in undergraduate level, to properly understand and apply it. Usually, final year students in computer science, software engineering and other divisions of computing have studied the concept of software development methodology in modules/courses such as software engineering, IT project management, system analysis and design, etc. However, I found it useful to review the concept again and to focus on this issue from another perspective.

## 4.1 Overview

Methodology has different meanings. As a branch of logic, it is about understanding the way that human beings knowledge is formed. It is also the combinations of best practices, procedures, rules and guidelines, in one word the methods, of the specific field of science and art by which professionals, specialists, and researchers can conduct their projects, research, development and study activities. Again, from another perspective, it is the study of those methods, which were just mentioned. By giving different meanings of methodology, I did not want to confuse you. It was just to let you have wider perspective of what the methodology can be in the different contexts. However, for our purpose the second meaning plus a trivial part of the first one is sufficient more than enough. It means that we are taking the methodology as the combination of methods by which we can conduct our project, and control it, on one hand, and the methods that we can apply in software development in order to deliver a software product, on the other hand. Moreover, we consider different methods and we argue why we have chosen a particular method.

Clearly, if your project is not involving software development, such as a research project for instance, then the methodology should be chosen and discussed based on a generally established approach for the scientific research. However, we are not going to discuss this concept in this book, instead, we will discuss and explain some of the major adapted approaches in software development context will be discussed and explained.

## 4.2 Software Development Methodology

Fortunately, we have left this perception behind that software development is a synonym to programming, long time ago. Nowadays, most of software developers, with or without a formal educational background in the field, have a clear idea that this activity is far beyond sole programming. They understand that it has more science and engineering in it than art. Indeed, the term “software engineering” has widely been accepted despite some trivial arguments against it that can be heard, here and there, from time to time.

My personal experience has brought me to this belief that software development, as a production activity, should be considered "naturally" and "mainly" an engineering process. However, like many other engineering activities it includes and actually, it needs some artistic endeavors and flavors. Moreover, in many cases, when the activity is not a simple reproduction or mass production of software, it becomes much more difficult to manage. This is what I have tried to teach my students in academy or share with my colleagues in the industry over the years.

The problem is, and surprisingly most of the software developers, at least verbally, confirm it as a problem, that because of the nature of software, which mainly comes from its intangibility, this engineering process can sometimes be jeopardized. There is a large tendency towards the ignorance of the engineering process in software development as the process it is very difficult to follow and control, especially when novice developers are involved. However, the complete argument on this subject is beyond of the scope of this book. Indeed, I assume that, hopefully, most of the readers of the book are already familiar with the concept and, again, hopefully, in line with the idea of taking software development as an engineering process.

Having what was mentioned, you could select and follow one of well known and well-practiced (software engineering) methods to accomplish your projects. Perhaps you have heard about different methods such as waterfall, spiral, prototyping, agile, Xtreme, object oriented, structured, Model View Controller (MVC), Rapid Application Development (RAD) and different frameworks such as Unified Process, Rational Unified Process (RUP), Microsoft SolutionFramework, and many more. How can we survive within this jungle of methods? How can we choose among them? To give you a straightforward answer is not easy though, it is not impossible!

It would not be a wrong claim that the notion that software development is an iterative activity has almost unanimously been accepted by developers. Whether by following a spiral method or non-spiral, you would usually build software during an evolutionary process. The process,at least most of the time,is iterative, and the final result can be obtained through accumulative increments, each one of which is supposed to satisfy one or several parts, but not all, of the requirements.

There is another decision that should be made before the development process starts. Which approach are you going to use to analyze, design, and implement your software? Structured, or object oriented? Some people may have a single and simple answer to this question. They would say, "Certainly, it is object oriented." If you ask about the reasons for this decision, you may receive several arguments which most of them are legitimate but not complete. For example, an argument can be about the dominance of object-oriented paradigm and its related technologies in the market and industry. Another one can mention the technical and theoretical capabilities of this method. Although, as it was mentioned,these arguments and other similar reasons are quite legitimate and understandable, yet the answer to that question, especially in your case, is not that straightforward. We will discuss this issue in the following section.

## 4.3 OO vs. Structured

OO or Structured? Which one is better? The second question is not appropriate. When we are in the process of choosing a method, we are not talking about these methods in general. We have to consider them in a specific context, which here is your project. The mentioned questions should be changed to "OO or Structured, which one is the most suitable approach to do my project?" The answer depends on several parameters and conditions. Below you can find these conditions in the format of some questions that can guide you through your decision.

**OO vs. Structured**

- How familiar are you with the method?
- How complex is your project?
- Which method is more suitable to analyze and design your system?
- What kind of development tools are you going to use?
- Is there any technical specification in your project definition about the methodology?
- Is there any preference or evaluation bonus on choosing a method?

Your answers to the above questions can provide you with a general view of which method is preferable in your case. However, as an engineering method we try to specify the situation in a more precise way, which you can find in the Table 4-1. For each parameter, you can provide a value between zero and five to indicate the measure of the parameter. Zero means the lowest rate and five the highest. This table shows an example. Clearly, you better to seek some advice from your supervisor in some cases, for example, to help you on the third question above.

Table 4-1 gives you an example. As you can realize, the results are too close to each other, however, in this example the OO is preferable. You may find some situations that the results are even. What should we do in these cases? Well, it is up to you! Do not forget about the harmony between the heart and the brain. Object-Oriented paradigm is not a fashion only. It is a very powerful approach, which stays in the software development as long as there are no other stronger competitors that they can give developers more than it has given. Therefore, as a person that should look at the career as well, developing a project with this method is a benefit. However, do not forget the approach that we took during the project selection. The aim at this step is to deliver a successful project.

| Methods / Parameters | OO | Structured |
|---|---|---|
| Familiarity with the method | 3 | 2 |
| Suitability for the project | 3 | 4 |
| Tools availability | 5 | 3 |
| Forced by project specification | 0 | 0 |
| Affects project evaluation | 0 | 0 |
| Result | 11 | 9 |

**Table 4-1** Choosing a method

Nevertheless, there are other concerns that should be taken into account. Are there only two options? Can we decide to take another approach? Can we get benefit of both object-oriented and structured methods? These questions would be addressed in the following section.

## 4.4 A Mixed-mode Approach

There are many undergraduate projects, which are not dealing with complex systems. They are not using complex objects and the functionality of the system is not difficult to understand. Most of these projects aim to equip the students to understand a general-purpose system and to utilize their knowledge in order to build a system from scratch. Many of these systems are using some kind of database, which users create and update them. In addition, these systems usually provide different kind of reports and outputs based on the expected functionality. Hence, the analysis and design of this type of projects would focus on the user interface and database design, none of which requires advanced object-oriented approach. However, one aspect of object-oriented, the Use Case Modeling, should be considered as a powerful tool that makes requirement management, analysis, and user interface design to be sound and flexible.

Using Use Case Modeling was introduced by Ivar Jacobson.Jacobson is one of the pioneers of object orientation, and a member of Gang of Three (including Grady Booch and JamesRumbaugh). Use Case Modeling has steadily become a powerful instrument in different steps of software development, since 1990s. This model has proven itself as a reliable, user-friendly, flexible, and powerful instrument to catch and model the software requirements. The independence of the model from other sections of the design gives the model the flexibility to be used with both object-oriented and structured approach in the software development. This is why when it is used with the traditional (structured) approach I call it "Mixed-mode " method, in order to show that this method has its roots from both main methods. Indeed, in this case, I have replaced the traditional Data Flow Diagrams (DFDs) with the Use Case Modeling.

In the rest of this book, my focus would be on this approach, because you can find extremely wide resources on each approach, separately. Hence, I am not repeating the material that you can find them in their origin formats. You can use the bibliography at the end of the book or simply search your university's library to find a huge pile of resources in this regard.

The mixed-mode approach uses the UseCase Modeling in four stages of your software development, which are Requirement Management, Analysis, User Interface Design and Testing. It also helps you during the Database Design and users guide compilation. As it is obvious, this method does not consider Class Modeling and Object Modeling so you can use development tools and programming environments in a traditional, structured method. But, it does not mean that you cannot use object-oriented aspects of your development environment.

Anyway, if you do not interested to bear with the complexity of object-oriented design and development, then this method eliminates this complexity and lets you implement your system in a more straightforward manner. Object oriented developers may not agree with me on this, but as a developer with more than 25 years of experience in both methods, I can tell you that the approach works properly and the development cycle can be shortened if the project falls in the categories, which are proper for the method. Among the projects that we discussed in section 2.1, only the first one, the Asset Management System is a good candidate for this approach. During the next chapters, you will find how this mix-mode approach can be implemented.

## 4.5 Summary

To do your project you need to follow amethod. Methodology discusses different methods, which are available in order to accomplish different projects. It is important to understand different methods that exist and applicable to software and information technology projects. Usually, this topic is addressed in different modules/courses such as software engineering, project management, and/or research methods. However, you have to have to make a trade-off between different methods and select the one that is more efficient regarding to your project.

Two main approaches to software development are structured (traditional) and object oriented. Both of these methods are supported by different tools and development environments. There are programming languages that support one or both methods. However, the processes of analysis and design are very different both in terms of their techniques and in terms of their outputs. Although object oriented method is a dominant method in the industry, however, structured method has still its own application in software development. Besides, there is a huge legacy of systems, which have been built, based on the structured approach. Therefore, this method is taught at universities and you should not ignore it as an option.

But, in this chapter, a mixed-mode approach was presented which utilizes the Use Case Modeling from the object-oriented method in different phases of software development and intertwines it with traditional Database Design in order to accomplish a project. In addition, a method was presented which helps you to choose a proper method. Moreover, eventhough several factors were presented in order to select a method, the success of the project was emphasized as the key decision factor in order to select the suitable method.

# 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.

# 6 Use Case Modeling and User Interface Design

Use Case Modeling is a powerful modeling technique that can be used in different stages of system and software development. Its understandability by technical and non-technical people alongside its simplicity, high flexibility, and readability has made it a first choice to capture and organize requirements. It can also be used in analysis, design, and testing processes. Use case modeling is supported by Unified Modeling Language (UML) as well.

Use case modeling plays a great role in capturing requirements, as well as in presenting the behavior of the system. It can be used both in understanding the current system, whether it is manual or automated, and to model a new system. Having both diagrammatical view and explanatory document, makes it an excellent tool for communication between developers and users. Again, like other subjects in this book I will explain the concept of Use Case Modeling and User Interface Design in the context of final year projects.

## 6.1 Use Case

A use case has two formats. The first one is a diagrammatic format, which includes a watermelon shape, a matchstick man, and a line between these two. It depicts a major functionality in the system and the way that a user, which is called an actor, would communicate to the system to acquire the mentioned function. The second format is a textual description, which explains what will be happening when the actor (user) requeststo use the "use case", in other words, what would be the case of using system by the actor.

Usingthe examples of the previous chapters, you can find out a Use Case, which corresponds to the first requirementof the sample 1 in the chapter 5 inFigure 6-1.

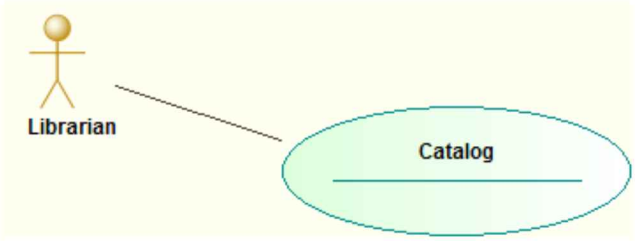


**Figure 6-1** Use Case sample (diagrammatic view)

These simple items, as you have learnt before, show that a Librarian communicates with a system to maintain the library catalog through the Catalog use case. Figure 6-1 cannot tell you how this happens though. This is not its responsibility to do so. In order to use case to be able to tell about the detail of its functionality, it should be accompanied by a textual explanation, which will be discussed in the next section.

## 6.2 Use case Model

Use case modeling is a dynamic activity. You would identify use cases and actors and you would put them on the context. As a first attempt on the sample 1 of our requirements in chapter 5, you may come up with the first-cut use case model as you can see in Figure 6-2.

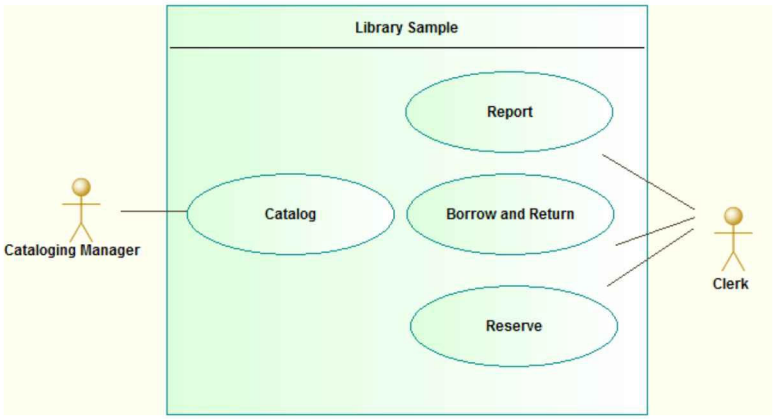


**Figure 6-2** Use Case Model (sample 1)

But, as you can see in section 6.5 this model may need some refinement. I would like to repeat that use case modeling, and perhaps all modeling activities, is a dynamic activity, which you would build your mode through consecutive steps until it fits your case.

## 6.3 Use Case Template

This explanation should follow a template. You can find many templates for use case and perhaps you have seen some up to this point of your study. Most of these templates have several items in common, which together forms their main characteristics. Some of these templates are too detailed and some are very simple. Below I will introduce a template, which I have found suitable for your final year project.

| **Use Case Document Template** | |
|---|---|
| **ID:** | Give an identification number that enables you to make the use case traceable. |
| **Name:** | The name that you have used in the use case model. |
| **Aim:** | The aim of use case. |
| **Main Actor:** | |
| **Pre-condition:** | What is the expected situation before the use case can be started? |
| **Main Scenario:** | Main scenario, which use case performs when it is started. |
| **Alternative Scenarios:** | Alternative scenarios, if there is any. |
| **Extension Points:** | Other use case to which this use case can extend. |
| **Post condition:** | What is the expected situation after the use case is finished. |

Here is my specific advice on how to use this template. Add your user inter face design to the above template where you intend to explain the use case scenarios. What does this mean? I will explain this in the next section.

## 6.4 User Interface and Use Case

Assume that you want to document the use case in Figure 6-1. It would be very helpful your customer/ supervisor, and for you/developer as well, to show a similar window (i.e. dialog box, greed, table, tabular sheets, etc.) that you are planning to use as user interface, when you are describing it. This way you would give a real sense of what is happening to you documents reader. In addition, you can role-play what is happening during the use case course. Furthermore, you can focus on the possible data entry as well. I can add more to these pros of combining user interface and embed it into use case. But, it is better for you to try it and see how it works.

In brief, you can design your user interface using the same tool that you are planning to use in the implementation phase and embed snapshots of which in your use case document. Otherwise, you can use other designing tools such as Microsoft Visio to design screen samples and embed them in the use case document, instead. Well, I prefer the former choice, anyway, but it is up to you to choose the one that you prefer.

## 6.5 Use Case Granularity

How many use cases you should have in your system? This question should be answered carefully. The answer depends on many factors such as the scope of the system, the complexity of the system, the relations with other system, and some other factors. However, some rules of thumb can help us to stay on the right track. Usually, in a project of the size of your final year project, you might have between 5 to 15 use cases.

If you have less than 5 use cases, then it seems that something is wrong. It can be the size of the system, which means that it is not a full system, rather it is a part of a system, or it can be from not understanding use case modeling properly. Whichever is the case, it would be better if you think about it again and to consult with your supervisor about it.

If you have more than 15 use case, I do not think can be because of your project size, rather it is because of your misunderstanding of the definition of use cases. Many students get confused about use cases. Sometimes experts do the same. Anyway, look at your use cases again and make sure that you have not taken a single step as a use case. Use case should fulfill part of a requirement, a requirement, or more than one requirement. But, in any case, it starts from a stable situation of the system and leaves system in a stable situation, when it finishes.

Nonetheless, there are situations that either you have less than 5 or more than 15 use cases and you cannot find any serious problems in your modeling. Well, here I should introduce you a concept, which is called "use case granularity". If your use cases number is less than 5, and you have a real system, then if it is helpful in managing the system, both from development perspective and from functionality perspective, then try to decompose one or more use cases into other use cases to increase the system granularity. If your use cases number is more than 15, then try to identify or define some subsystems to which you assign use case which are more related to each other. Normally, a subsystem is better not to include more than 10 use case.

Figure 6-3 shows the use case model of Figure 6-2 on which the granularity concept has been considered. However, this model still can be decomposed more and even to be redesigned to include at least three subsystems such as Catalog, Membership, and Borrowing, each of which can include a couple of use cases.

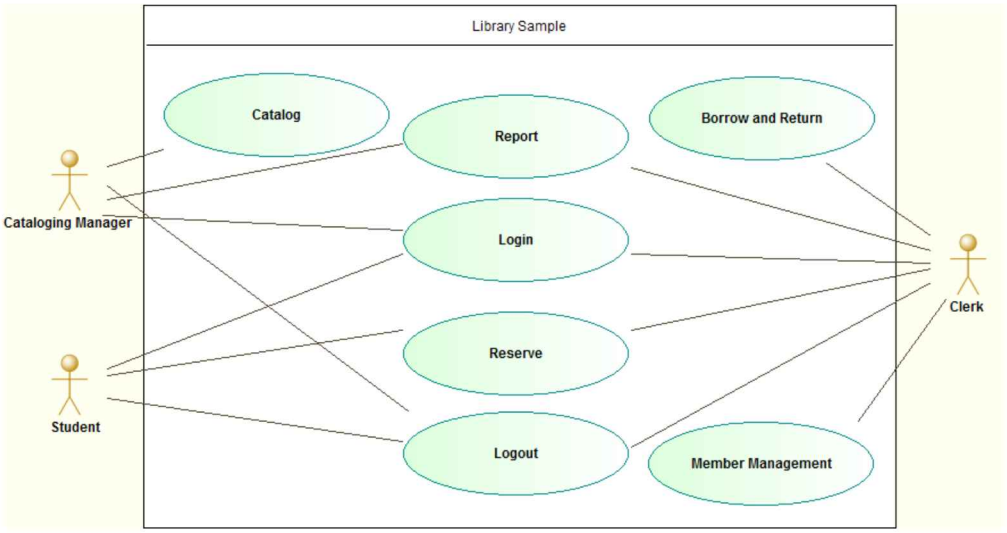


**Figure 6-3** Use Case Model (sample 1 – revised)

## 6.6 More applications of Use Case

As it was mentioned, the main purpose of use cases is to capture requirements and then to use them in the analysis in order to provide a more understandable view of the system, both for developers and customers. However, use case can be used in other stages of the system and software development. One case was user interface design, which was discussed earlier in this chapter. In addition, use case can be used in the testing process. Moreover, they can be used for users guide and system help preparation. The scenarios that you have already explained, and the screen snapshots that you have already embedded into use case explanations can act as a rough material for the mentioned purposes. Furthermore, use case documents can be used as rough material to prepare training documents, as well. Now, can you tell me how amazing this use case thing is? Thank you Ivar Jacobson, for introducing this excellent idea!

## 6.7 Summary

Use Case Modeling helps developers to capture, organize, and manage requirements. In addition, it is a useful tool in other development stages such as analysis, design, and testing. It can be used in users guide preparing and even in user training process.

Use case is the way that user/actor communicates with the system. Actor asks system to perform a function and system responds to the actor's request through a course of actions. Use case modeling has two perspectives, one is diagrammatic, and the other is textual. In order to simplify the use case explanation, a template was introduced. In addition, it was suggested to combine user interfaces with use case explanation in order to make it easier for all people who are involved in the system development process to understand the analysis and design results. Finally, the use case granularity concept was discussed and some rules of thumb to manage the use case granularity were provided.

# 7 Database Design

Database design plays a major role in application systems. Almost all courses/programs, which are related to software development and information technology, include at least one module on database systems. Like other sections of this book, the intent of the book is not to repeat the concepts that you might have studied thoroughly, rather to show you the way that you can adapt your knowledge about the subject and utilize it properly in your final year project.

## 7.1 Database Management Systems

Database Management System (DBMS) is a software system, which manages database construction, update, and access control, and maintenance. I assume that you are familiar with some DBMSs such as My SQL, MS SQL, MS Access, Oracle, etc. Some DBMSs are licensed software and some others are open source. You have to choose a DBMS as the database tool for your final year project if you are going to develop a database application.

Below you can find some tips on selecting a DBMS.

**Tips on DBMS section**

- Choose a DBMS that you have already your hands dirty with it.
- Unless you have strong reasons don not go for small/medium-scale databases such as MS Access.
- Choose among large-scale databases e.g. My SQL or MS SQL.

## 7.2 Relational vs. Object Oriented Database

The most common DBMSs in the market are Relational Database Management Systems (RDBMS). Their popularity, both in the market and education, make them the first choice for most of the final year projects. However, other types of databases, particularly Object Oriented Database Management Systems (OODBMS), have steadily been growing during the past decade. In fact, you can download and use some of them such as ObjectStore and combine them with your programming environments i.e. MS Visual Studio.

So, which one is preferable? RDBMS or OODBMS? My quick answer to this question is "if you are about to do an ordinary application and you do not have previous background on OODBMS, and particularly if your methodology is not object-oriented do not think about using an OODBMS. In other words, unless you have a strong reason and desire to use OODBMS, from one side, and it can give more facility and robustness to your system that is not achievable using RDBMS, choose one of the well-known RDBMSs, especially among those with which you have already had some experience".

Finally, some RDBMSs support Object Relational Models (ORM), which allow developers to map their object models into relational models. Although, the process of mapping your class model into an ERD might persuade you to use ORM, but you should be careful about the concepts behind each one. To clarify, you can use ORM only if the DBMS supports it. In this case, you should develop your data layer in your system architecture according to this decision. However, if you are not planning to use ORM, then you have to map your persistent classesinto an ERD, and then apply normalization techniques on the result to obtain your relational database design. In this case, you have to develop your data layer, accordingly, which might not be the same as you could have in the previous situation.

## 7.3 Data Modeling

Data modeling, which is also called conceptual database design, is a core activityin database applications. It is comparable with the importance and the role of foundations of constructions in civil engineering. I assume you have had at least one module/course about database design before you start your final year project. However, this familiarity cannot guaranty that your database design is proper. Below you can find some tips on data modeling.

**Tips on Data Modeling**

- Use an automated tool, preferably the one that your chosen DBMS provides, for your data model.
- Apply normalization rules and normalize your database design at least to the 3rd Normalization form or to BCNF.
- Use understandable names for relations/tables and attributes/fields.
- Use proper data types and apply DBMS provided restrictions and rules such as Not Null.
- Use proper naming for Foreign Keys and Relationships.
- Check your design by providing data through DBMS interface and control the expected outputs by running some Structured Query Language (SQL) commands on the provided data.

If you follow an ordinary (structured) methodology, then you would provide an Entity Relationship Diagram (ERD). You can use this diagramto provide a normalized data model, afterwards. However, if you follow object-oriented methodology then you should know how you could map your class model to an ERD. Most of IDEs and development tools are providing tools, add-ons, and facilities to help in the mapping process. If you are using one of these tools then your job is easier. Simply use the facilities that they provide. However, if you do not have an automated tool to do so, below you can find some rules of thumb for mapping a class model into an ERD, manually.

**Mapping a Class Diagram into an ERD**

- Indicate your persistent class, e.g. by using a persistent stereotype.
- Create an Entity for any persistent class, giving the name of the class.
- Map all data members of the class as attributes of the created Entity.
- Give the same, or equivalent, data type to the attributes.
- Make the class identity (ID) the Primary Key for the Entity.
- Map associations, aggregations, and compositions to relationships and use class IDs as Foreign Keys.
- Change generalization/specialization relations of the class model to superclass/subclass relations and apply ERD rules, accordingly.
- Map multiplicities of associations to the equivalent cardinalities in the ERD.

## 7.4 Database Design

The next step is to prepare a database design, which is also called physical database design, based on the data model. This modelis quite similar to your data model. However, data model is a DBMS independent model while database design should be prepared according to a specific DBMS. The reason is in database design you have to specify the attribute types, length, key types, attributes' default values, and other parameters, which may be slightly different from one DBMS to another, yet this slight difference can make your design to work on one DBMS, but not the other.

## 7.5 Using Stored-Procedures and Triggers

Stored-procedure is a specific procedure that is written in Standard Query Language (SQL) as a database item. Stored-procedures can accomplish database related requirements efficiently. Although they are part of database, however, many experts categorize them under business logic layer in a three-tier/multi-tier software architecture. Stored-procedures are independent of specific tables.

Trigger is a piece of SQL, which can perform a sequence of actions if an action such as Create, Update, and Delete happens on specific table.

If your project were a database application, and if your DBMS supported stored procedures and triggers, it would be a good practice to use these features. In this case, do not forget to document stored procedures and triggers properly and provide them as part of supplementary documents in your final report.

### 7.5.1 XML

Extensible Markup Language (XML) is a markup language that you have studied during modules related to database or web technology. I advise you to consider XML as a data container in your final year project if one or more of the following conditions are applicable. Be careful about the term "data container" that I used. Differentiation between a "data container" and a "database" is important. Several arguments and discussions can be found on whether you can use XML as a database or not. However, many DBMSs provide diverse number of utilities to facilitate using XML with databases. This phenomenon, in its essence, shows that the arguments against using skeletal format of XML as a database have been widely accepted. On the other hand, providing specific tools regarding securing the XML files and treating them as database, particularly for semi-structured or unstructured data is becoming very popular.

If you use XML in the context of DBMS, then you have to follow and apply specific rules that are pertinent to using XML in that context. You can find several resources on this in the bibliography section. Anyway, I will leave you in peace by avoiding more technical (even philosophical!) discussions on this matter, rather, I provide with some tips on using XML in your final year project.

**Tips on using XML**

- XML is mainly a technology to be used to organize/standardize data exchanges
- Consider using XML as a data container in your final year project if:
  - Your data is semi-structured or unstructured
  - You do not have heavy update on your database
  - Security is not a main concern
  - You do not deal with a large amount of data
- Ignore all the above and use XML if you are not obliged to use a traditional database! This is an opportunity for you to practice an amazing software technology.

## 7.6 Summary

Most of the final year projects are dealing with databases one way or another. However, some projectsmight not use any specific database or database systems. If your project includes a database or if it is a database application, then you have to design a proper database. Although recently Object Oriented Database Management Systems (OODBMS) have started to play a greater role than they were playing, however, Relation Database Management Systems (RDBMS) are still the most popular databases, both in the market and education. Therefore, whether you use an object-oriented or traditional approach, it is most probablethat you would use a RDBMS in order to implement your database.

If you follow a traditional approach, you are required to create an Entity Relationship Diagram (ERD) during your database design. However, if you follow an object-oriented approach you do not have this diagram, hence, some straightforward rules were presented in this chapter, using which you can create an ERD from your Class Model. Then, you can follow the normal process of normalization and database design as you would do it in the traditional approach. Finally, the importance of choosing the proper RDBMS was discussed and some guidelines were provided in order to help you in finding suitable RDBMS for your final year project.

# 8 Implementation

By implementation, here, I mean programming. I mention this because the word "implementation" is used for two different purposes in the software development context. It is used to address the programming phase of software development, as its widely used concept, however, it is used as installation and making the system ready for use as well, in some other contexts.

Programming phase of the software development is the phase that gives "birth" to your system. It is the time when you materialize what you have analyzed and then designed on paper or electronic diagrams. In fact, this is the time when your thoughts become real. Of course, the separation of phases in software development is not too strict. However, different phases are clearly distinguishable. When we talk about an implementation phase as the programming time, it does not mean that you are only doing programming at this time, rather it means that your main activity is programming, while you may still do a little design and even analysis activities.

However, it is crucial to every kind of software development and information technology projects that you have everything designed and thought about before you start the implementation phase. To make an analogy, it is similar to when a civil engineering team starts to build a construction, a house, a bridge, or a road. They cannot start the building without having all their blueprints ready. If they do, it is more likely to them to end up with a disaster or a catastrophe rather than what they would expect.

In order to make the implementation phase a success, the first step is to decide what kind of tools and what programming environment you are going to use. Indeed, you should have decided and documented this in the previous phases, as we discussed in the design phase. There is a jungle of programming languages and environments out there. How can we decide on which one to choose in this extremely variable environment? This chapter aims to help you on this issue.

## 8.1 Implementation Tools and Environment

Software implementation includes several activities of which programming and debugging are considered as the core activities. To program you need to use one, and sometimes more than one, programming language. You have to "write" your program, compile it, link different pieces of it together, integrate it with a database if it need be, execute it (run it), find its probable errors (debug it) and fix them, and finally make it deliverable to your customer (your school, department, or university).

There are many programming language alongside their related editors and compilers. In fact, you can find several different products even for a specificlanguage such as C++, Java or Python in the market. Some of these products are proprietary (licensed) software and some others are opensources. Moreover, most of the major licensed Integrated Development Environment (IDE) providers give some sort of special offer for students and academic institutions. However, the choices are so diverse that you should be very careful in choosing one. In the following sections, I will discuss this issue in order to make your decision easier regarding implementation.

### 8.1.1 Programming Languages

Since the first generation of programming languages were borne, their community has been growing to several hundreds. Many of these languages have died or became obsolete during the time. However, some have been able to continue their lives. Indeed, these resistant languages, have adapted themselves to the environment by evolving to the higher generations or because of some other scientific or practical reasons. For instance having large amount of software in these languages, which have been widely in use, is one of those reasons. You have studied different courses/modules covering the programming techniques. Up to this point you have done different assignments and coursework in programming, you have learnt at least one or two programming languages, and you have become familiar with some Integrated Development Environments (IDEs) that let you program efficiently.

### 8.1.2 Object Oriented vs. Structured Programming

Although, we are talking about programming language in the implementation phase, however, you are expected to decide on which tool and programming language you are going to use in your project during the previous phases. Like other development tools, which we discussed in the previous chapters, you should consider several factors while you are deciding on this issue.

Are you going to use an objected-oriented or structured programming environment and tools for your implementation/programming phase? It seems odd to many students and sometimes even to my colleagues when I ask this question. The prejudgment is so strong in this case nowadays. In most of the cases, the answer would be “of course, object-oriented”. When you ask for a reason, they reply with a sort of surprised faces. The response includes several items, the theme of most of which would go around the current available tools, the importance of the approach, the dominance of it in the market and the obsolescence of the traditional or structured programming. Obviously, except for the last item, i.e. “the obsolescence of the traditional programming”, nobody can argue and deny the correctness of the others. But, you can hardly find afactual discussion about a specific project and its requirements, which justify the decision.

### 8.1.3 Open Source

Open source has become a strong reality in the software development field, despite many years of discussions and arguments between the proprietary (licensed) software and open source communities. In fact, it sometimes has become a part of economic and political decisions of governments. The discussion of idea and philosophy of opensource and its pros and cons is beyond the scope of this book. In the context of this book, these arguments and their results do not play a great role and we are taking about the open source tools as the licensed ones.

However, some differences might be of more importance when you are choosing open source software. For instance, choosing a stable version and preferably from a well-known provider and resource. I will come back to this issue, shortly.

### 8.1.4 Integrated Environment Development

Using Integrated Environment Development (IDEs) was started during mid-1970s and has been gradually evolved to extremely comprehensive tools and environment since then.This has enableddevelopers to perform their activities in a more efficient way. However, although programming without using IDEs can be a very cumbersome job, from one side, using IDEs needs proper training and familiarity with the tool, on the other side. Overall, using IDEs gives you almost all the tools that you need to develop your systems, such as project creators, smart editors, compilers, debuggers, testers, database integrators, configuration management tools, and more, in one package.

Again, it is a good practice to answer some questions about the programming approach. Below you can find some questions of which some decision-making factors have been extracted in order to help you to make your decisions with your eyes open.

**Tips on IDEs**

- How familiar are you with this programming language?
- How familiar are you with the IDE?
- How efficient is this programming language to be used for the projects similar to your final year project?
- Are there some experts in this IDE and language that you can ask for help in case of facing difficulties?
- Is there any technical specification in your project definition about the implementation?
- Is the IDE is easily available (i.e. in open source form or by having academic special offers, etc.)?
- Is there any preference or evaluation bonus on choosing a specific tool?

Your answers to the above questions can help to decide which IDE and implementation tool best suits your final year project. However, as our practice in the other cases of this guideline, I am tryingto help you to specify the situation in a more precise way, which you can find in the Table 8-1. For each parameter, you can provide a value between zero and five to indicate the measure of the parameter. Zero means the lowest rate and five the highest. This table shows an example. Clearly, you better to seek some advice from your supervisor and your fellow students or alumnus in some cases, for example, to help you on the third question above, which affect the third parameter in the Table 8-1.

Table 8-1 gives you an example. To avoid any bias in choosing current available tools I have decided to use hypothetical names as tools-1, tools-2, and tools-3 in this case. You can replace them with whatever is real in your specific case. For example, you can use Microsoft Visual Studio, Eclipse, Interactive Ruby, XAMPP, WAMP, etc. As you can realize, the results are might be close to each other, however, in this example the tools-2 is preferable. You may find some situations that the results are even or too close to each other. What should we do in these cases? Well, it is up to you! Do not forget about the harmony between the heart and the brain. However, having a look into the current market and its expectations from developers, your ability and skills on how to use the tools can be considered as an additional decision making parameter. It means that as a person you should look at your career as well. It means that developing a project with a specific tool, which is highlydemanded in the market, can be a good point on your CV. However, do not forget that the aim of your project at this step is to deliver a successful product.

| Implementation Tools / Parameters | Tool – 1 | Tool – 2 | Tool – 3 |
|---|---|---|---|
| Familiarity with the method | 2 | 4 | 5 |
| Suitability for the project | 3 | 4 | 2 |
| Tools availability | 5 | 4 | 4 |
| Forced by project specification | 0 | 0 | 0 |
| Affects project evaluation | 0 | 0 | 0 |
| Result | 10 | 12 | 11 |

**Table 8-1** Choosing an implementaion tool

Do we have to restrict ourselves with three tools in all cases? No, definitely we do not. Nevertheless, making it too complex leads us astray of our main path. There are some hints that can be used as general guidelines. Equally important, you should choose among those tools, which are in line with your methodology. In other words, this is not a good decision, if not impossible, to choose among non-object-oriented implementation tools while we have decided to not using object-oriented approach. Similarly, it would not be proper, and sometimes would not be possible, to choose a non-object-oriented implementation tools when we have decided to follow an object-oriented approach.

## 8.2 Customization

Nowadays, many open source software allow you to download a complete solution framework, which are ready-made for some kind of problems and provide you with facilities that let you change and customize it in order to make your own system. Whether your school/department allows you to do so or not is an issue. However, if you are allowed to use this kind of framework then you should consider the following issues.

**Tips on Customization**

- Ask the authorities and consult your department to check if you are allowed to take a customization project as your final year project.
- Even you are allowed change your mind if it is possible, otherwise:
  - Add some requirements that the ready-made solution does not provide.
  - Make these requirements "Must have" requirements.
  - Adapt some other parts through localization (see section 8.3) process.

## 8.3 Localization

In the software technology context, localization is a processthrough which developers make specific software, which has been developed for a particular sociocultural environment, ready to be used in a new environment. This process can target different aspects of software, i.e. language, culture, and process workflow. Sometimes this concept comes alongside internalization, which means making local software ready to be used in a more diverse context and sociocultural environment. This process is important in software industry and needs different skills and knowledge, especially about the target market. Nevertheless, I do not advise students to consider it as a choice for their final year projects.

Nonetheless, some students are interested in this kind of projects, particularly, when they target some ready-made open source software that they can adapt it for local usage. If this is the case, then make sure that you can show some implementation activities, which are measurable in a way that your project can be evaluated as a product on your own. Explain your reasons to take this solution for your project, properly, and establish a profound background for your choice. Furthermore, try to understand the software architecture properly in a way that shows that you are in control of the product. Moreover, add some features based on the requirement specifications to augment your implementation activities and to go beyond the linguistic requirements. For example, changing interfaces based on local culture, changing workflows according to the local process, etc.

## 8.4 Summary

The implementation phase is the stage that you make your system alive. It is comparable to building a construction in civil engineering or making a car in mechanical engineering. This is the time that you make your system real based your blueprints which you have prepared during the previous stages. In order to do so, you need to select and utilize an implementation tool. Your implementation tool should be adaptable with your methodology and should be able to support your design material. In other words, when you have decided to follow an object-oriented method, you should choose an implementation tool among object-oriented ones and if you have decided to go for non-object-oriented methods, it is better to choose among non-object-oriented development environments. However, these are only guidelines and you have to make your decision based on some analysis.

In order to make your decision easier, some guideline questions were presented in this chapter. Then, similar to other cases in this book, based on these questions severalfactors were extracted and organized into a table. Finally, this table used to help to make your decision on the implementation issue.

As opens sources are becoming more and more important, there are technical and practical concerns that should be considered by developers. These concerns were discussed in different sections. You should be careful on customizing and localizing open sourcesoftware that allow you to build some sort of products, quickly. Although, they might serve the success of your project, however, the danger of producing a shallow product and preventing you from practicing on the fundamental concepts is something very serious that you should be aware it.

# 9 Testing

Despite this fact that testingphase is technically situated after the implementation phase, however, it practically starts during the earlier stages. Normally it starts during analysis, and remains as a continuous activity throughout the development process. Clearly, the emphasis would grow stronger when the development enters the testing phase and the nature of testing activities would change during the development course. Therefore, it is important to understand the testing process in its entirety and not to restrict it as an activity after building the system. Again, like other topics in this book, I assume that you have some backgrounds on the testing concepts and issues. Therefore, the discussion would be arranged around your final year project and the way that you can adapt the testing process in it.

## 9.1 Testing Process

Testing is a process; it means that it normally should receive some inputs, processes them, and provides some outputs. It includes several activities, which should be carried out in different stages of software development. However, as it was mentioned, the core activities would be carried out during the testing phase. Normally, several different specialties are needed for the testing process to be accomplished. But, again, we are not discussing the process outside of the context of the final year projects. Consequently, you are expected to play most of the roles that are required in this process.

In fact, you should design test cases, provide test scenarios, collect and arrange test data, perform the test, and document the result. These activities have been shown by using a UML activity diagram as you can find in Figure 9-1. If you compare this simplified testing process with similar testing process diagrams, regardless of the way that they are presented, you can realize that some detailed activities and loop-backs in the process have been removed. It does not mean that this has unreasonably diluted the testing process.

Again, I would like to emphasize that many parameters such as size, human resources, time, and the complexity level of the system affects the testing process. For example, “regression test” is an important concept in the testing process. Simply, regression test means that you have to repeat the tests that you have performed earlier if you change your system. This idea has been implemented in many testing tools, however, to my opinion, it is not necessary for the majority of final year projects.

As another example, we can talk about “automated testing”. Simply, it means that either you use automated tools to test your system, or you create necessary programs that performing the tests, automatically. Isautomating the testing process necessary in your final year project? Well, in most of the cases the answer would be “No”. However, you have to look at the context and the development environment. If the development tool that you have selected provides you with efficient testing tool, certainly, it would be a good idea to use it. However, you should be careful to not spend more than the scheduled time that your project allows you for testing just because the “automated testing” is a fashion that you should follow.

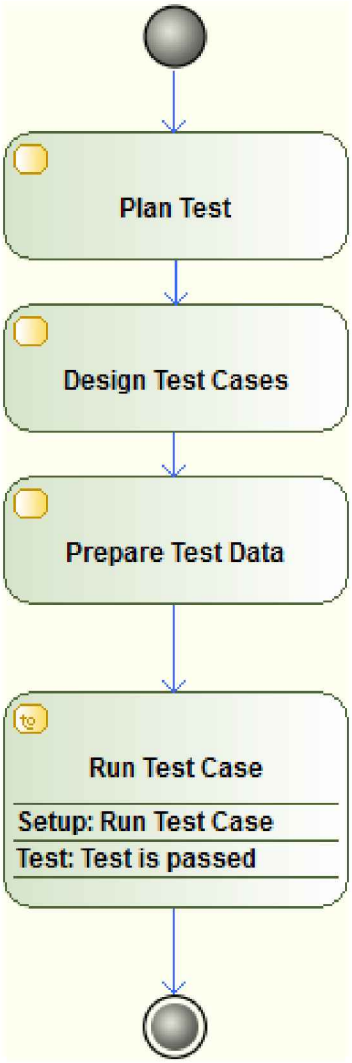


**Figure 9-1** Testing Process Activity Diagram

Below we are going to discuss the test classification at the first step, and then we will discuss test cases and test data preparation. In addition, the test result documentation would be discussed.

## 9.2 Test Categories

Throughout the years, test has been classified into several categories. Below, thesecategories have been summarized in order to provide you with the general topics of testing.

**Software Test Categories**

Software testing can be categorized from different perspectives as below. However, these categories are neither complete in terms of its diversity nor rigid in terms of classification. They have been provided to let students have a general view on the testing categories.

- From system perspective:
  - System Test
  - User Test
    - Interface Test
    - Acceptance Test
- From software perspective:
  - Component Test
  - Unit Test
  - Integration Test
- From non-functional requirement perspective:
  - Performance Test
  - Security Test
  - Reliability Test
  - Scalability Test
- From test data perspective:
  - Conformance Test
  - Destructive Test
  - Non-destructive Test

Usually, you should not focus on all the above categories.In fact, unless specific requirements are asking you to perform other tests, considering some of them would properly suffice your testing process. These selective tests are Interface Test, Component Test, Unit Test, Integration Test, and Security Test. In addition, in some cases you may combine these tests together. For instance, you can embed the Interface Test within the Unit Test.

## 9.3 Test Case

A test case is a document that explains how a software product would be tested against a specific requirement in order to show that the requirement has been fulfilled by the system or if the system has failed to fulfill it. For large projects, you have to have formal documents for test cases. Different templates have been suggested by the experts in the field. In some contexts, you may find "Test Suite" as well. Test suite s is documents that usually include several related test cases. There is no need for you to worry about test suites in your final project. However, you can read more about test suites and test cases by consulting the related books in the bibliography section.

But, does it necessary for you to be so formal in your project? The blunt answer is "No". You have to be as formal as a project of the size of a final year project should be. Well, how much is this amount of formality? To illustrate, I suggest you to prepare two tables to document your test cases. The tables can be formed as you can see in Table 9-1 and Table 9-2. These tables can be prepared using a word processor or a spreadsheet. They help you to manage your test even if you ask a third person as a user to perform a user test or acceptance test, for example.

| **Test Cases** | | | | | |
|---|---|---|---|---|---|
| ID | Test Case Name | Test Subject | Test Sequence | Test Data | Req/UC ID |
| | | | | | |

**Table 9-1** Test Case template

### 9.3.1 Test Scenario

Test scenario is a term that if you look it up in different texts, you would get different meanings, some of which is quite similar to what I said about the test case. To reduce the confusion, I would say the relation of a test case to a test scenario is similar to the relation of a use case to use case scenario. To simplify, test case explains the general information about how a test should be done, while, test scenario explains how you would apply those general explanations in specific context, and with specific data. To illustrate, a test casewhich aims to test a catalog information (a requirement with ID = 1 in the sample 1 of chapter 5), has been shown in Table 9-3. This test case can have several scenarios. For instance, the first scenario provides data to catalog a book, and the other one to catalog a journal, yet the third one tests recording a CD in the catalog. However, some experts may disagree with me on this issue.

### 9.3.2 Test Data

Test data is the data based on which you perform the test. Test should be prepared in way that covers different cases. You have to expect the behavior of the system based on different inputs. It means that you provide test data and you predict the system response if the mentioned data entered. This way you can test the system and see whether it is successful in performing its functions or fails to do it. Below you can find some hints on test data preparation. Clearly, this simplified guideline should be taken as a version that fit your final year project. In commercial projects, test data preparation is a highly professional activity and needs more effort to be accomplished. You can consult the bibliography section at the end of the book form more information on this issue.

**Tips on Test Data**

- Provide different test data for different test scenarios.
- If data is numeric, it must cover:
  - Lower bound (e.g. if this is a money attribute with 8 digits and no fractions it must take 0, 1, 2, for instance)
  - Middle (e.g. 55,870,20 and 11,659,450)
  - Upper bound (e.g. 88,999,999 and 99,999,999)
  - Negatives (e.g. -1)
- If data is alphanumeric, it must cover:
  - Maximum expected length
  - Mixed alphanumeric samples
  - Empty (blank)
- If data is an image, it must cover:
  - Images larger than the specified area
  - Images which exactly fit the are
  - Images which are too small to be used

### 9.3.3 Test Results

Table 9-1 documents the test cases. You should provide a unique ID by which you can identify the test case throughout your project and its related documents. You should give a name, which makes it easier to understand the test case. In addition, you should specify your test subject by specifying the subject type, i.e. screen, module, subsystem, system and the name/address of the test subject. In the Test Sequence field, you should explain the sequence through which the test should be performed. It is a good practice to have this sequence in a bulleted format to allow yourself or a tester to follow it in a stepwise manner. In the Test Data field, you can provide the data that you intend to use for the test activity. If the place was not spacious enough to hold what you want to document, then, you can put the address of related document(s) that you used for this purpose, instead. However, in most of the cases they are sufficient for their purposes. The Requirement/Use Case ID is the same ID that you have used in the Requirement Specification document in Table 5-1and Table 5-2. To make a cross-reference and trace test cases back to the requirements.

| Test Results | | | |
|---|---|---|---|
| Test Case ID | Attempt | Scenario | Result |
| | | | Pass/Fail – Comments |

**Table 9-2** Test Result template

Table 9-2 documents the test results. The ID is the same ID that you have used in Table 9-1 to make a cross-reference between the results and the test cases. The Attempt refers to the repetition attempts in case a test case has failed and the activity has repeated. In this situation, you would have more than one row for a specific ID, clearly, with different results. The Result field holds the result as Pass or Fail. In the latter case, you can explain the result or attach a screen capture image by providing a link to the image file, if you prefer to do so.

Table 9-3 shows an example based on the samples that were used in the previous chapters

| Test Cases | | | | | |
|---|---|---|---|---|---|
| ID | Test Case Name | Test Subject | Test Sequence | Test Data | Req/ UC ID |
| 1 | Test Catalog | Screen | Click on "New Catalog" button.<br><br>Enter all required data.<br><br>Click on "Save" button.<br><br>Click on "Yes" button of the message box. | **First scenario**:<br>Material Type: Book<br>Author: Pressman, R.S.<br>Title: Software Engineering:<br>A Practitioner's Approach<br>Year: 2010<br>Edition: 7th<br>Publisher: McGraw-Hill<br>ISBN-10: 0072496681<br>ISBN-13: 978-0073655789<br>Number of Copies: 20<br>Classification: Dewey<br>***Expected Result:*** Material Saved<br>**Second scenario**: Same data except for the following part:<br>ISBN-10: 000123<br>***Expected Result:*** System gives a warning that the ISBN-10 format is not correct. | 1 |

**Table 9-3** Test Cases sample

If you look at the test data column of Table 9-3, you can see that the data has been organized under different scenarios. Two scenarios were given as an example. After conducting the test, the result has been documented as you can see in Table 9-4.

| Test Results | | | |
|---|---|---|---|
| Test Case ID | Attempt | Scenario | Result |
| 1 | 1 | 1 | Pass |
| 1 | 1 | 2 | Fail – System crash – See screen shot at this address |

**Table 9-4** Test Results Sample

## 9.4 Object Oriented Test

Although the foundations of software testing process are independent of the software implementation, but the techniques used in different development paradigms are different. This difference becomes significant in some cases such as the difference between testing of traditionally implemented software versus object-oriented software. Detailed discussion on this issue is beyond the scope of this book and you can consult the related books in the bibliography section. However, I advise you to consider the followings as the minimum testing objectives if you are going to use object-oriented method.

### 9.4.1 Class Test

Prepare test data to test your implemented classes. Focus on business layer, but do not forget to test your data layer and presentation layer classes as well.

### 9.4.2 Package Test

If you have combined your classes into packages that participate in the system as a subsystem, prepare test cases that test the input/output of these packages and their communication with other parts of system.

### 9.4.3 Interface Test

By "interface", here, I mean Interface classes, which you have studied in your object-oriented programming modules/courses. You should be careful about these interfaces and the way that they perform their functions. You should prepare test cases that checks these interface and not only make you sure that they do what they are expected to do, but also they do not do anything else which leads to the breaching of object-oriented rules.

## 9.5 Validation and Verification

Validation and Verification (V&V) is a concept in software testing and quality control. By validation, you have to answer this question, "Are you developing the right system?" whereas by verification, you have to answer this question, "Are you developing the system right?" As you can realize, the first question is validates the system, which you are doing it through the testing process. But, the second question is about the process that you have followed to develop the system. Although you should answer this question and you should be able to show evidence of the positive answer to this question, however, this is your supervisor that can help you through her/his feedbacks by letting you know that you are on the right track. Furthermore, your supervisor or evaluation panel would judge on this issue when you deliver your final project. You can find more on this issue by consulting bibliography section of this book.

Finally, I did not discuss the debugging process in this book as I assume that you have received enough lessons about it in different programming modules/courses, and practiced it during laboratory exercises. However, although debugging is considered as a test activity in many contexts and it is specially categorized under the validation process, it should not be confused with the testing process. To my view, debugging is part of programming. It is so interconnected to the programming tasks that you cannot assume at as a separate task. On the other hand, as it was mentioned, test is a planned process that might be performed independent of programming.

## 9.6 Summary

Evaluation and testing is an unavoidable part of any software and information technology projects. You might have studied different topics on testing through modules/courses such as programming, software engineering, project management and such. Although, you might have received a great deal on the importance of testing, however, it is quite normal that you might have a tendency to underestimate it.

Despite my suggestion on the simplification of software development process in your final year project, I would like you to pay as much attention as you can to the testing process. This is what that gives your project a strong flavor of both engineering and scientific approach. Although general guidelines of testing, verification and validation process exist that you can follow, however, the testing could differ according to the methodology that you have chosen to follow. For example, if you have decided to use object-oriented programming you should either know or learn about this method's specific testing techniques.

Finally, you should appreciate the role of use cases, especially if they are well prepared, in designing test scenarios. This is one of the best practices that I have found during developing several projects. I have realized that how useful the use cases can be in the design of test scenarios and in the process of conformance test and user test. Therefore, my advice is to utilize the use cases, which you have documented in the analysis phase, in the testing stage. You can find more on this in the related subjects in the bibliography section at the end of this book.

# 10 Report Writing

Presumably, you think that writing the report of your final year project is the last step that you should do. Although this assumption is partially correct, however, it should not lead you to postpone the report writing activity to the last minute of your schedule. In fact, you have to start to write your report from the early stages of your project. If you have a look at Figure 3-2 again, you can find out that the report-writing task has been considered as an ongoing task after literature review finished. This means that your report is gradually completed alongside other activities that you would perform during the development process. Clearly, as you can move towards the end of your project, the time and effort that you dedicate to this task would increase.

Most of the schools/departments provide you with a template based on which you would prepare your final report. These templates may differ in detail or even the topics that they expect. However, there are commonalities among majority of them. Below I will suggest a general structure based on which I will explain the expected contents of each section.

## 10.1 Report Structure

You can find a template for the report structure below. However, depending on your specific project you may revise it.

**Report Structure**

- Cover Page
- Executive Summary/Abstract
- Acknowledgement
- Table of Contents
- List of Figures
- List of Tables
- Introduction
- Literature Review
- Methodology
- Solutions
- Requirement Management
- Analysis
- Design
    - User Interface Design
    - Database Design
    - Software Design
    - Implementation
- Testing
- Conclusion and Future works
- Bibliography/References
- Appendices

Each item of the above structure would be briefly explained in the following subsections.

### 10.1.1 Cover Page

Cover page is the identity of your final year report. I have seen many carelessly prepared cover pages and have penalized the providers (my students are well aware of this)! Unless there is a specific template in your department, make sure that you have provided at least the following information on your final year project report.

**Cover Page information**

- University Name and Logo
- Department
- Course/Program
- Your name
- Student ID
- Project Name
- Course/Module title and code
- Supervisor

### 10.1.2 Table of Contents

Prepare Table of Contents (TOC), List of Figures, and List of Tables. Most of word processors, e.g. MS Word help you in automatically prepare and update these lists. Utilizing these facilities prevents your document from wrong page references.

### 10.1.3 Executive Summary/Abstract

Executive Summary (ES) is a one-page explanation that explains to the reader the problem, how it has been tackled, and what the outcomes are. Although it is placed as the first page after the cover page, however, it is normally written at the end of project, when everything is clear! As the name is applying, it is a short summary for the executive management of your project, here the supervisor or the evaluation panel, to let them understand in a glance what the project is about, how it has been done and what the conclusion is.

If the project is a research-based project, usually, Executive Summary is replaced by "Abstract" title. However, the main theme remains similar, largely.

**Executive Summary**

- It is placed as the first page after the cover page
- It is normally written at the end of project, when everything is clear!
- It is a short summary to show:
  - what the project is about,
  - how it has been done,
  - and what the conclusion is.
- It is NOT an explanation on the report structure!

I have seen many reports within which the students have confused between Executive Summary and Introduction. Be careful on this issue and look the hints on these two to understand their differences.

### 10.1.4 Introduction

This chapter of your report provides a background of the project, states, in general terms, what the report is about and how it has been structured. Introduction can be divided to subsections such as Overview, Background, and Report Architecture. As you can see below, introduction has a clear-cut difference with the Executive Summary/Abstract. However, I have seen many reports with a transposition of Introduction and ES.

### 10.1.5 Literature Review

Although this section mainly applies to the research-based projects, however, it is a good idea to have it in the normal projects as well. In this chapter, you should acknowledge the previous works on the area of the project in order to show their main outcomes, strengths, and spaces for improvement. Consequently, you can show how your project has been built upon the previous ideas and works and how it would contribute to add more to this background.

### 10.1.6 Methodology

This chapter depicts how you have applied the subjects that we discussed in in chapter 4. Students, sometimes, refer only to the tools that they have used and the subjects that they have studied as their methodology. However, you should not write this chapter, aftermath! Methodology is something that you have to decide on during the early stages of your project. As a result, you can write this chapter as soon as you made your decision. This was one reason, among many, that I mentioned that report writing was an ongoing activity and should not be postponed to the last minute.

### 10.1.7 Solutions

This chapter can be considered as a subsection to the methodology chapter or as an individual one. Whichever way you choose, you have to clearly state that why you have taken this specific approach to develop the system. If there are more than one solution to your problem, name these solutions and talk about their pros and cons and then explain the reasons behind your decision to take one not the others. For example, you have to explain that why you have used a web-based system and not a desktop one.

### 10.1.8 Requirement Management

In this chapter, show the project requirements as we discussed in chapter 5. Use the classification that was used or other classifications that you have studied. But, this is important to depict the requirements clearly. Do not forget to pay proper attention to the functional requirements. Try to write this chapter when you finished the activity or at least draft its contents, otherwise, you may loss lots of data when you start writing it at the end of the project.

### 10.1.9 Analysis

Depending on your methodology, this chapter would contain use case modeling or data flow analysis. Whichever is the case, show the analysis steps and try to use proper figures (e.g. UML diagrams) to show the general view of the system. If you have restrictions on the word count of your report, then provide the main analysis here and move the details to the appendices of your report.

### 10.1.10 Design

Depending on your methodology, this chapter would contain different materials. However, structuring it as three subsections can be a good idea.

#### 10.1.10.1 Software Design

In this subsection, you should provide your software design. I did not discuss software design in details as it is beyond the scope of this book, and you can find enormous resources about the subject (see the bibliography for instance). However, you should show the system architecture, decompositions, and modularity in this section. If you followed an OO method, then, you may provide package model, class model, and interaction model in this chapter, as well.

#### 10.1.10.2 User Interface Design

You should introduce the User Interfaces (UI) in this chapter or show a sample of which and explain the way that you have designed it, then, leave the details to the appendices. You can use the approach that I advised you in chapter 6 to combine the UI design with use case description.

#### 10.1.10.3 Database Design

If your project is a database application, I advise you to be as generous as you can with the explanation of your data model and database design. In this subsection, you can provide the ERD and then the normalized version of your database design. Providing the details of your tables can give your project, more flavor and would show your profession. However, you can move some samples of triggers and stored procedures, if you have any, to the appendices section.

### 10.1.11 Implementation

Implementation chapter should focus on the codes. In addition, you should present the general architecture of implementation. Provide samples of codes. Explain algorithms used. If you have a considerable amount of code, then either provides them through a supplementary media or as one the appendices.

### 10.1.12 Testing

Testing chapter should show how the test process has been implemented, in general, and how the verification/validation techniques have been applied, in particular. Samples of test cases and test data explanation should be provided here. Importantly, test results, the outcome not the detail, should be provided as well. You can provide details of the test process as one of the appendices.

### 10.1.13 Conclusion and Future Works

Although it is usually a short chapter, however, conclusion and future works chapter is the distillation of your report. It should show the main outcome of your project.You should critically discuss the outcomes of your project, and point out your opinion on the results and outcomes. Try to be neutral at this stage and look at the outcomes as an outsider.

### 10.1.14 Bibliography/References

To clarify, bibliography is the list of resources that you have consulted with for your project but not directly cited, while, references is the list of resources that you have cited.

Most of departments have their suggestion for using bibliographical/citation style. However, in some cases you might not receive any specific suggestion. In this case, choose a style that you are already familiar with such as APA or Harvard citations and bibliography style. Fortunately, most of word processors, e.g. MS Word, can provide you with tools, which support several common citations and bibliographic styles.

### 10.1.15 Appendices

Well, the name is pretty much reflecting the content, and hence, there is no need to talk more about it. In addition, in the previous sections I addressed those parts of report that can be extended more by moving material to appendices section, wherever it was appropriate.

## 10.2 Proofreading

You have to follow academic writing techniques in your report writing. You have received proper training on this subject during your study at the university. However, most of the time, we forget to check our writing according to these techniques. This is important for both native speakers and those who use English as the second (perhaps third or fourth) language (like me, myself!). Unfortunately, most of the time students are careless and negligence about this activity. Below you can find some hints about proofreading.

**Proofreading**

- Use spell checkers.
- Do NOT trust spell checkers!
- Print your report; forget about being "green" at this stage.
- Find a quiet place.
- Read your report word by word, loudly.
- Correct errors with a distinguishable pen/marker.
- However, if you are so interested to remain "green", some word processors and some digital gadgets allow you to perform proofreading, digitally. Therefore, there is no way to escape of proofreading and no excuse that justify our errors (please let me know about mine in this book!).

## 10.3 Summary

Your final year project would be presented through three main items, namely, software, presentation, and project report. Project report is a key document that shows how you have accomplished your project. It presents your understanding of the project, explains the project plan, provides the results of each stage of your project, describes the outcome, and presents the evaluation and your conclusion on your final year project. In fact the project report is one of the main deliverable items based on which your project would be evaluated and marketed.

Usually, faculties/schools/departments are providing a template based on which you have to prepare your final report. However, in this chapter, a general outline for a typical project report was presented. It is crucial to pay attention to all the details that you have to present, however, your focus should be on those chapters of the report, which carry the higher marks. Moreover, being careful on the academic writing style, citation, referencing, proofreading, and general report structure such as cover page, table of contents, pagination, headers and footers are important as well. They show how professional you are in documenting your project.

# 11 Supporting Documents

To support your final report with more evidence, it is a good idea to have an appendix through which you can provide some extra documents.These documents should aim in helping your supervisor to have a better figure on what you have done during your final year project. This appendix may include several items. Certainly there is no restriction in it, however, below some specific parts have been suggested, which I think can be considered as the minimum.

## 11.1 Codes

Depending on your department request, you might or might not include your codes in your report. In some cases when you are using open source systems they provide you with a framework and its related source of codes, such as those which are used for building portals, for example. It is not necessary, if not applicable, to print all the codes that are not yours. Rather, it is better to concentrate on the customized parts and print some samples of which. In addition, there are non-open source projects for which enormous codes are created, whether automatically or manually. For this type of projects as well, some samples of the code would suffice. Below you can find some tips on how you should prepare your codes for your final report.

**Tips on presenting software code in your report**

- Include those parts of code that is your work.
- Make sure that you have properly commented the codes.
- Provide a title that represents the main functionality of the code.
- If you have customized an open source, make sure that you have properly referenced its origin.
- Do not include parts of code, which are automatically generated by the tool you have used for the implementation.

## 11.2 Test Documents

Provide your test cases, test data, and test results as evidence of your testing process. Categorize test documents based on the guidelines, which were provided in chapter 9.

## 11.3 Project Diary

As it was mentioned in section 3.4, it is a good practice to keep a diary for your final year project. Do not forget to provide this document as a supplementary document to show what has happened during your long journey through your final year project.

## 11.4 Electronic Documents

Obviously, in almost the entire computing projects your documents and products are mainly in electronic format. Your final year project is exceptional in this regard. According to your department procedures for the final year project, you might be asked to deliver your project on a physical medium, and electronic medium, or both. Regardless of the format, paying attention to the preparation of these documents and organizing them properly have a great impact on your project evaluation.

In the following subsections, I will provide you with some useful tips on preparing these documents.

### 11.4.1 Physical Medium

If you are asked to prepare a physical medium on which you should provide your final year project, make sure that you have paid utmost attention in preparing this item. Sometime, because of the procedures, if your physical medium could not operate properly it would be considered as incomplete project. This may cause you a severe penalty in your project evaluation. Below you can find some helpful hints on preparing a physical medium for your final year project.

**Tips on Electronic companion of your report / physical medium**

- Always support your report with a digital media (e.g. CD/DVD), which includes:
  - The system
  - The designs, codes, test cases and test data
  - Sample data/database (if it is applicable)
  - Installation guides
  - The final report (in pdf, MS Word, etc. format)
  - The presentation
  - Other supportive documents (screen shots, DB SQL files, etc.)
- Label this media properly using special markers. (I have received CDs, which were not readable because they have been written on with ballpoint pencils!)
- Include in this label at least:
  - Your name
  - Student ID
  - Project Name
  - Course title and code
  - Department
  - Supervisor
- Academic year/semester.
- Use a proper cover for the media that can be attached to your final report.
- Check your media by copying it on your computer to see that everything is OK.
- Organize all the above under properly structured folders and subfolders with proper naming style.

A templatefor the folder structure for your final year project can be found below.

**Folder Structure Template**

- FirstNameLastName – Student ID-Project Abbreviation
    - Report
    - System
        - Analysis
        - Design
            - Software Design
            - User Interface Design
            - DB Design
        - Code
        - Test
            - Test Case
            - Test Data
            - Test Result
        - Installation
            - Installation Data

### 11.4.2 Non-Physical Medium

Nowadays you are usually asked to post your project to an electronic postbox. In this case, follow the procedures of the electronic postbox, if there is any, otherwise find below some tips about how to provide your documents via this medium.

**Tips on using Non-Physical Medium for the project submission**

- Prepare a folder based on the above template.
- Make a compressed file out of which (e.g. zip, rar, etc.).
- Check your file by uncompressing it in another location.
- If it is successful, then upload your file to the electronic postbox, otherwise fix the problem and repeat this step.
- Do not forget to save any feedbacks or receipts showing that your document has been received, as evidence.

## 11.5 Summary

Your project report needs to be supported by several documents. One reason is that you may have some restrictions on the word count for the report, which is imposed by the university/faculty/school/department rules and regulations. In this case, supporting documents are normally not counted as the report word count. Moreover, putting these documents may distract your reader's attention (i.e. your supervisor or other examiners) from the main concepts and subjects that you would like to present throughout your report. For example, if you have prepared a user's guide or installation guide, it is better to organize them as supported documents in the appendices. Again, the project plan, project diary, and/or possible questionnaires that you might have, can be provided as supporting documents. In addition, test scenarios, test data, test results, program codes and such can be provided as supporting documents as well.

# 12 Presentation

Presentation is the way that you are going to sell your products. You might face with different situations. Some departments ask students to present the outcome of their projects in front of a panel of academic staff in a way that is expected for a viva in postgraduates. However, many departments are not asking for this kind of presentations. As this book aims to help postgraduates in their final project as well, I decided to dedicate a chapter to this activity. As undergraduate students, even if you are not requested to present your final year project, the guide can be useful in other situations when you have to do so.

Presentation is an activity, which takes place in a short period, usually between 10 to 20 minutes. Therefore, its structure, preparation, contents, and delivery should be very well designed and implemented. Presentation providesyou with a unique opportunity within which you are able to introduce your capabilities on the subject, understanding of the chosen topics, and presenting the results and outcomes of your project. In this chapter, I will give some specific explanation on how you can prepare yourself for this important event. Do not forget that this activity must have been planned and scheduled in your project plan and you should be very careful on the timing of the event.

## 12.1 Presentation Structure

Presentation should be organized based on the allowed timing. Below I will provide two scenarios. In the first scenario, I assume that you have 20 minutes, which has been dived to three sections; 5 minutes for preparing the environment, 10 minutes for your presentation, and 5 minutes for question and answers. In the second scenario, I assume that you have 30 minutes of which 5 minutes are for preparation, 15 minutes for your presentation, and 10 minutes for questions and answers.

It is important to use presentation tools such Microsoft PowerPoint or Apple i-Work as they help to organize your presentation in an efficient manner. In addition, they can provide you with some predefined templates, which make your job easier to do. However, if your department requires you to prepare your presentation based on a departmental template then you are obliged to follow their template and structure.

Below you can find two proposed structures for the presentation. In these samples, presentation timing, its agenda, proposed title for each slide, and proposed timing of each section/slide have been shown. These suggestions should not be taken for granted. They are not rigid suggestions. Presentation is a very constructive activity and you should show your creativity in composing it. Therefore, take these samples as a guideline and be creative as much as you can.

The first structure has been proposed for 20 minutes as below:

**Presentation Structure 1 (for 20 minutes)**

Presentation time: 20 minutes as below:

Preparation: 5 minutes

Presentation: 10 minutes

Questions/Answers: 5 minutes

Number of slides: 12

Presentation Structure:

- Slide 1: Project information (15 seconds)
- Slide 2: Agenda/Topics (15 seconds)
- Slide 3: Problem Statement/Project Definition (30 seconds)
- Slide 4: Main Requirements (30 seconds)
- Slide 5: Background/Literature Review (1.5 minute)
- Slide 6: Methodology (30 seconds)
- Slide 7: Analysis (1 minute)
- Slide 8: Design (2 minute)
- Slide 9: Implementation (1 minute)
- Slide 10: Findings and Evaluation/ Links to live system or other documents (1.5 minute)
- Slide 11: Conclusion and Future Works (1 minute)
- Slide 12: Thank you and Q/A

The second presentation structure has been proposed for 30 minutes as below:

**Presentation Structure 2 (for 30 minutes)**

Presentation time: 30 minutes as below:

Preparation: 5 minutes

Presentation: 15 minutes

Questions/Answers: 10 minutes

Number of slides: 12

Presentation Structure:

- Slide 1: Project information (15 seconds)
- Slide 2: Agenda/Topics (15 seconds)
- Slide 3: Problem Statement/Project Definition (1 minute)
- Slide 4: Main Requirements (1 minute)
- Slide 5: Background/Literature Review (1.5 minute)
- Slide 6: Methodology (1 minute)
- Slide 7: Analysis (1.5 minute)
- Slide 8: Design (2.5 minute)
- Slide 9: Implementation (3 minutes)
- Slide 10: Findings and Evaluation/ Links to live system or other documents (2 minute)
- Slide 11: Conclusion and Future Works (1 minute)
- Slide 12: Thank you and Q/A

## 12.2 Preparation

Preparation has two stages. The main stage should be finished at lease several days before the actual presentation. Below you can find some useful tips that you can consider with regard to the presentation preparation.

Make sure that you have scheduled this activity in your project plan. In addition, it is a good practice if you ask your supervisor to review your presentation before you go live. You might be asked to present a quick show on your product, if the product is software. In this case, you have to check with your supervisor to arrange the presentation venueto be ready for you in terms of required hardware and software.

**Presentation Preparation Tips**

- Schedule your presentation as an compound activity in your project plan as below:
  - Presentation
    - Composition
    - Rehearse
    - Checked with supervisor (as milestone)
    - Check presentation venue and environment (as milestone)
    - Actual presentation (as milestone)
- Set the dates for the latter two tasks at least 3 days (preferably one week) before the actual presentation.
- Rehearse in front of your family/classmates/friends/previous graduates.
- Make at least two backups of your presentation.
- Have one of your friends/classmates laptop ready to use for your presentation in case of any failure.
- Be ready on campus at least 30 minutes before your presentation.

## 12.3 Rehearsal

Presentation is a skill. You have to practice it in order to become a skillful presenter. I assume that many of you have had at least one presentation. Presentation is similar to a play, and therefore, you have to rehearse it. There are different ways of rehearsal. The best method can be in front of some similar audience and in the same venue that the presentation is expected to be performed. However, this is not something that is always possible. Consequently, you have to rehearse it at home alone or in front of your family, possibly your siblings, or in dormitory in front of your roommates, in campus in front your classmates, or in front of your supervisor. If none of these is the case, then try it in front of a mirror! Have a clock in front of you; look at your gestures; or if you are using presentation software utilize its automatic slide movement.

## 12.4 Presenting

This is the time when your show is going live. Below you can find some tips about it.

**Presenting Tips**

- Take a deep breath and relax
- Say good morning/afternoon/evening
- Introduce yourself
- Introduce your project
- Introduce your supervisor
- Talk in a loud, but calm voice
- Have an eye contact with all audience
- If there is a question while you are presenting, if you have a short answer then answer it otherwise excuse yourself and kindly ask the person to postpone her/his question to the end of the presentation
- In any situation do not get irritated
- Have a gentle smile on your face during the presentation!

## 12.5 Summary

Presentation is the time that you sell your product. A good presentation requires some specific skills, which, hopefully, you have gained and practiced during your study. Some good rules of thumb help you to provide an attractive presentation. Moreover, you can find useful resources in the bibliography section at the end of this book. In addition, there are many helpful resources easily available on the Internet that can guide you in this regard. However, an important step is to rehearse your presentation in front some audience, your classmates and university fellows for example. This rehearse step can give you more confidence while when you are going to provide your real show. Moreover, it can reveal some flaws and problems in your presentation.

# Last Word

Well, that was it, for the time being. I hope this brief can help you, as students in computing, to plan and conduct your final year project smoother and easier. However, I would like to warn you about a serious case that I have seen as a proof of Murphy's Law in this context. I have seen some anxious and worried students who have come to my office or have sent me an email showing an unimaginably disparatesituation. They have wanted to know that if I, by any chance, have saved their reports, documents, models, and so on, which they had sent me the other time, somewhere! I do not ask them the reason for this, actually, I can tell them immediately that they have lost their documents and they could not have recovered it. That is why they are asking this. If these are my students, probably in software engineering, or database, or programming, then, I am about to telling them "my friend, did not I tell you in the class to take practical hints, seriously? Did not I repeat several times that taking the backups is not just a pieceof advice to give it to our users? Did not I tell you that rather it is much more important for us?" But, then I think loudly, "does it help to give a lecture about what should not have happened, aftermath?"

Folks, I hope that you take the backup issue, especially in your final year project case, as seriously as it deserves. I do not talk any more to this. Take it as the last word.

# Bibliography

AB, M., 2006. *MySQL administrator.* Indianapolis, USA: MySQL Press.

Ambler, S.W. & Sadalage, P.J., 2007. *Refactoring databases.* Upper Saddle River, USA: Addison-Wesley.

Anderson, R.J. & Long, C., 2008. *Security engineering.* 2 ed. Indianapolis, USA: Wiley Publishing Incorporated.

Balter, A. & Kanouse, G., 2006. *Microsoft SQL server 2005 express in 24 hours .* Indianapolis, USA: Sams Publishing.

Bardzell, J. & Sharp, W., 2003. *Macromedia dreamweaver MX dynamic applications.* Berkeley, USA: Macromedia Press.

Barzdins, J. & Caplinskas, A., 2001. *Databases and information systems.* Dordrecht, The Netherlands: Kluwer Academic Publishers.

Beekman, G. & Quinn, M.J., 2006. *Tomorrow's technology and you: introductory.* 8th ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Bell, D. & Parr, M., 2004. *C# for students.* Harlow, England: Addison-Wesley.

Bennett, S., McRobb, S., Farmer, R. & Mosman, K., 2006. *Object-oriented systems analysis and design using UML.* 3rd ed. ed. London, England: McGraw Hill.

Bentley, L.D., Whitten, J.L. & Randolph, G., 2007. *Systems analysis and design for the global enterprise.* 7th ed. ed. Boston, USA: McGraw-Hill Irwin.

Bernard, S.A., 2005. *An introduction to enterprise architecture.* 2nd ed. ed. USA: Authorhouse.

Bishop, M., 2004. *Introduction to computer security.* Boston, USA: Addison-Wesley.

Bittner, K., Spence, I. & Jacobson, I., 2007. *Use case modeling.* Boston, USA: Addison-Wesley.

Blanchard, B.S., Fabrycky, W.J. & Svendsen, E., 2006. *Systems engineering and analysis.* 4th ed. ed. s.l.: Prentice Hall.

Bloch, J. & Steele, G., 2007. *Effective Java.* Boston, USA: Addison-Wesley.

Blokdijk, G., 2008. [*IT*] *Risk management guide.* USA: Gerard Blokdijk.

Bon, J.v., International, I. & Wilkinson, J., 2007. *Foundations of IT service management.* 3rd ed. ed. Zaltbommel, The Netherlands: Van Haren Publishing.

Bon, J.v., Pieper, M., Veen, A.v.d. & Verheiken, T., 2007. *Introduction to ITIL.* London, England: The Stationery Office .

Booch, G., Rumbaugh, J. & Jacobson, I., 2003. *The unified modeling language user guide.* Boston, USA: Addison-Wesley.

Bradley, J.C. & Millspaugh, A.C., 2006. *Programming in Visual Basic .Net.* Boston, USA: McGraw-Hill Irwin.

Brooks, F.P., 1995. *The mythical man-month.* Anniversary ed. Reading, USA: Addison-Wesley.

Brookshear, J.G. & Triebel, L., 2005. *Computer science.* 9th ed. ed. Boston, USA: Pearson Addison Wesley.

Bussler, C. & Shan, M.-C., 2006. *Technologies for e-services.* Berlin, Germany: Springer.

Bustamante, M.L., 2007. *Learning WCF.* Sebastopol, USA: O'Reilly.

Chakrabarti, S. et al., 2009. *Data mining.* Burlington, USA : Morgan Kaufmann Publishers.

Chandra, B., 2005. *Object oriented programming using C++* . 2nd ed. ed. Harrow, England: Alpha Science International Limited.

Chatfield, C., 2010. *Microsoft Project 2010.* Redmond, USA: Microsoft Press.

Chatfield, C. & Johnson, T., 2003. *Microsoft Office Project 2003.* Washington, USA : Microsoft Press.

Chipman, M., Baron, A. & Clapp, C., 2000. *Microsoft access developer.* Indianapolis, USA: Sams Publishing.

Chorafas, D.N., 2002. *Enterpirse architecture and new generation information systems.* Boca Raton, USA: St Lucie Press.

Clements, J.P. & Gido, J., 2006. *Effective Project Management.* Newyork: THOMSON-SOUTH-WESTERN.

Commerce, O.o. G., 2007. *The official introduction to the ITIL service lifecycle.* London, England: The Stationery Office.

Connolly, T. & Begg, C., 2010. *Database Systems: A Practical Approach to design, Implementation and Management.* 5 ed. NY: Addison Wesley.

Connolly, T.M. & Begg, C., 2005. *Database systems.* 4th ed. Harlow, England: Addison-Wesley.

Connolly, T.M., Begg, C.E. & Hirsch, M., 2010. *Database systems.* 5th ed. ed. Boston, USA: Addison-Wesley.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. & Stein, C., 2007. *Introduction to algorithms.* 2nd ed. ed. Cambridge, USA: MIT Press.

Cyganski, D., Orr, J.A. & Vaz, R.F., 2000. *Information technology.* Upper Saddle River, USA: Prentice Hall.

Daft, 2008. *The Leadership Experience.* 4 ed. s.l.: Thomson Learning.

Daley, B., 2006. *Computers are your future: complete.* 9th ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Daniel, Y.L., 2011. *Introduction to Java Programming, Comprehensive.* 8 ed. s.l.: Prentice Hall.

Darie, C., Ruvalcaba, Z. & Laidlaw, G., 2006. *Build your own ASP.NET 2.0 web site using C# & VB.* 2nd ed. ed. Collingwood, Australia: Sitepoint Pty Limited.

Date, C.J., 2004. *An introduction to database systems.* 8th International ed. ed. Boston, USA: Pearson Addison Wesley.

Dawson, C.W., 2005. *Projects in computing and information systems.* Harlow, England: Addison-Wesley.

Dawson, C.W., 2009. *Projects in computing and information systems.* 2nd ed. ed. Harlow, England: Addison-Wesley.

Dawson, M. & Lee, M., 2010. *C++ projects.* Boston, USA: Course Technology, Cengage Learning.

Deitel, H.M. a. D.P.J., 2012. *C++ How to Program.* 8 ed. London: Prentice Hall.

Deitel, H.M., Deitel, P.J. & Cappello, J., 2006. *Visual C# 2005.* 2nd International ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Deitel, H.M., Deitel, P.J., Hoey, T.R. & Yaeger, C.H., 2004. *Simply C#.* Upper Saddle River, USA: Pearson Prentice Hall.

Deitel, H.M., Deitel, P.J. & Nieto, T.R., 2003. *Simply Visual Basic Net.* Upper Saddle River, USA: Prentice Hall.

Deitel, P., 2012. *C++.* 8th International ed. ed. Boston, USA: Prentice Hall.

Deitel, P. et al., 2010. *iPhone for programmers.* Upper Saddle River, USA: Prentice Hall.

Deitel, P., Deitel, H., Hort & Hirsch, M., 2011. *Visual C# 2010.* 4th International ed. ed. Boston, USA: Pearson.

Deitel, P.J. & Deitel, H.M., 2009. *Visual C# 2008.* 3rd International ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Deitel, P.J., Deitel, H.M. & Horton, M.J., 2010. *C++.* 7th International ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Deitel, P.J., Deitel, H.M. & Snyder, C., 2008. *C++ how to program.* 6th ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Dix, A., Finlay, J., Abowd, G. & Beale, R., 2003. *Human-Computer Interaction.* 3 ed. s.l.: Prentice Hall Europe.

Dix, A., Finlay, J., Abowd, G.D. & Beale, R., 2004. *Human-computer interaction.* 3rd ed. ed. Harlow, England: Pearson Prentice Hall.

Dorsch, W., Lee, R.Y. & Wu, C., 2005. *Software engineering research, management and applications.* Berlin, Germany: Springer Verlag.

DuBois, P. & Clapp, C., 2003. *MySQL.* 2nd ed. ed. Indianapolis, USA: Sams Publishing.

DuBois, P., Hinz, S. & Lentz, A., 2005. *MySQL language reference.* Indianapolis, USA: MySQL Press.

Easwarakumar, K.S., 2003. *Object-oriented data structures using C++.* New Delhi, India: Vikas Publishing House PVT Limited.

Elliott, G., 2004. *Global business information technology.* Harlow, England: Pearson Addison Wesley.

Englander, I., 2003. *The architecture of computer hardware and systems software.* 3rd ed. ed. Hoboken, USA: John Wiley & Sons.

Englander, I., 2010. *The architecture of computer hardware, system software, and networking.* 4th International ed. ed. Hoboken, USA: John Wiley & Sons.

Evans, E.G. & Ward, P. L., 2007. *Management Basics for Information Professionals.* 2 ed. NJ: Neal-Schuman.

Fowler, M., 2003. *Patterns of Enterprise Application Architecture.* s.l.: The Addison-Wesley Signature Serie.

Fowler, M., 2004. *UML distilled.* 3rd ed. ed. Boston, USA: Addison-Wesley.

Fowler, M., 2005. *Analysis patterns.* Boston, USA: Addison-Wesley.

Frederick, G.R. & Lal, R., 2009. *Beginning smartphone web developmen.* New York, USA: Apress.

Galitz, W.O., 2007. *The essential guide to user interface design.* 3rd ed. ed. Indianapolis, USA: Wiley Publishing Incorporated.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J., 2007. *Design patterns.* Boston, USA: Addison-Wesley.

Garmany, J. & Burleson, D.K., 2004. *Oracle application server 10g adminstration handbook.* New York, USA: McGraw-Hill/Osborne.

Gary, F.C. & Larson, W.E., 2008. *Project Management, the managerial process.* 4 ed. Singapore: McGraw Hill Higher Education.

George, J.F., Batra, D., Valacich, J.S. & Hoffer, J.A., 2007. *Object-oriented systems analysis and design.* 2nd ed. Upper Saddle River, USA: Pearson Prentice Hall.

George, J.F., Batra, D., Valacich, J.S. & Hoffer, J.A., 2007. *Object-oriented systems analysis and design.* 2nd ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Giokoetxea, A., 2007. *Enterprise architectures and digital administration.* Singapore, Singapore: World Scientific.

Gittleman, A., 2005. *C# .NET illuminated.* Sudbury, USA: Jones and Bartlett Publishers.

Giudici, P. & Figini, S., 2009. *Applied data mining for business and industry.* 2nd ed. ed. Chichester, England: John Wiley & Sons.

Goikoetxea, A., 2007. *Enterprise Architectures and Digital Administration Planning, Design and Assessment.* NJ: World Scientific.

Gold-Bernstein, B. & Ruth, W., 2005. *Enterprise integration.* Boston, USA: Addison-Wesley.

Goldreich, O., Rosenberg, A.L. & Selman, A.L., 2006. *Theoretical computer science.* Berlin, Germany: Springer.

Goldstein, N., 2010. *iPhone application development.* Hoboken, USA: Wiley Publishing Incorporated.

Gollman, D., 2006. *Computer security.* 2nd ed. ed. Chichester, England: John Wiley & Sons.

Gomaa, H., 2005. *Designing software product lines with UML.* Boston, USA: Addison-Wesley.

Goodrich, M.T. & Tamassia, R., 2002. *Algorithm design.* Hoboken, USA: John Wiley & Sons.

Gouge, I., 2003. *Shaping the IT organization.* London, England: Springer-Verlag.

Gray, C.F. & Larson, E.W., 2000. *Project Management, The Managerial Process.* Newyork: McGraw Hill-International Edition.

Grembergen, W.v. & Has, S.d., 2008. *Implementing information technology governance.* Hershey, USA: IGI Publishing.

Gross, C. & Moodie, M., 2008. *Beginning VB 2008.* New York, USA: Apress.

Haag, S., Baltzan, P. & Phillips, A., 2008. *Business driven technology.* Boston, USA: McGraw-Hill Irwin.

Haag, S. & Cummings, M., 2008. *Information systems essentials.* 2nd ed. Boston, USA: McGraw-Hill Irwin.

Haag, S. & Cummings, M., 2008. *Management information systems for the information age.* 7th ed. ed. Boston, USA: McGraw-Hill Irwin.

Haimes, Y.Y., 2004. *Risk modeling, assessment, and management.* 2nd ed. ed. Hoboken, USA: Wiley Interscience.

Harold, E.R., 2004. *Effective XML.* Boston, USA: Addison-Wesley.

Harris, P.E., 2007. *Prince2 Planning and control using Microsoft project.* Doncaster Heights, Australia: Paul E. Harris.

Harris, S., Ross, J. & Long, C., 2006. *Beginning algorithms.* Indianapolis, USA: Wiley Publishing Incorporated.

Hart, M. & Jesse, S., 2004. *Oracle database 10g.* New York, USA: McGraw-Hill/Osborne.

Haselden, K., Baker, B. & Gettman, K., 2006. *Microsoft SQL server 2005 integration services.* Indianapolis, USA: Sams Publishing.

Hoffer, J.A., Prescott, M.B. & McFadden, F.R., 2007. *Modern database management.* 8th ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Hoffer, J.A., Prescott, M.B. & Topi, H., 2009. *Modern database management.* Upper Saddle River, USA: 9th International ed.

Hughes, B. & Cotterell, M., 2006. *Software project management.* 4th ed. London, England: McGraw-Hill.

Hughes, B. & Cotterell, M., 2009. *Software project management.* 5th ed. ed. London, England: McGraw-Hill Higher Education.

Hunter, M.G., Tan, F.B. & Snavely, J., 2009. *Handbook of research on information management and the global landscape.* Hershey, USA: Information Science Reference.

Institute, I.G., 2005. *Governanave of the Extended Enterprise-Bridging Business and IT Strategies.* Hoboken, USA: John Wiley & Sons

Ithurralde, I. & Ramkaran, A., 2005. *This is IT 2.* 2nd ed. ed. Abingdon, England: Hodder & Stoughton.

Jackson, R., Burd, S.D. & Satzinger, J.W., 2004. *Object-oriented Analysis and Design with the Unified Process.* s.l.: Course Technology Inc.

Jacobson, I., 2002. *Use Case Modleing.* 3 ed. Newyork: Adisson Wesley.

Jacobson, R. et al., 2006. *Microsoft SQL server 2005 analysis services step by step.* Washington, USA: Microsoft Press.

Janakiraman, V.S. & Sarukesi, K., 2009. *Decision support systems.* New Delhi, India: PHI Learning Private Limited.

Jashapara, A., 2004. *Knowledge Management An Integrated Approach.* Essex: Pearson Education.

Jennex, M.E. et al., 2008. *Knowledge management.* Hershey, USA: Information Science Reference.

Jessup, L.M. & Valacich, J.S., 2006. *Information systems today.* 2nd ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Josuttis, N.M., 2007. *SOA in practice.* Sebastopol, USA: O'Reilly.

Kendall, K.E. & Kendall, J.E., 2008. *Systems analysis and design.* 7th International ed. Upper Saddle River, USA: Pearson Prentice Hall.

Kendall & Kendnall, C., 2010. *Systems Analysis and Design.* 8 ed. NJ: Pearson Education Inc.

Kifer, M., Bernstein, A. & Lewis, P.M., 2006. *Database systems.* 2nd International ed. Boston, USA: Pearson Addison-Wesley.

Kizza, J.M., 2007. *Ethical and social issues in the information age.* 3rd ed. London, England: Springer Verlag.

Kleinberg, J., Tardos, E. & Suarez-Rivas, M., 2006. *Algorithm design.* Boston, USA: Pearson Addison-Wesley.

Know, D. & Carey, D.W., 2004. *Effective Oracle database 10g security by design.* New York, USA: McGraw-Hill/Osborne.

Kochan, S.G., 2009. *Programming in Objective-C 2.0.* 2nd ed. ed. Upper Saddle River, USA: Addison-Wesley.

Koffman, E.B., Wolfgang, P.A.T. & Santor, K., 2006. *Objects, abstraction, data structures and design using C++.* Danvers, USA: John Wiley & Sons.

Kolling, M., Hirsch, M. & Horton, M., 2010. *Introduction to programming with Greenfoot.* Boston, USA: Pearson.

Kroenke, D.M., 2006. *Database processing.* 10th ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Kroenke, D.M., 2009. *Database Processing: Fundamentals, Design and Implementation.* 10 ed. NY: Prentice Hall.

Kroenke, D.M. & Auer, D.J., 2007. *Database concepts.* 3rd ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Kroenke, D.M. & Auer, D.J., 2008. *Database Concepts.* 3 ed. s.l.: Pearson Education Inc..

Kroenke, D.M. & Auer, D.J., 2010. *Database Concepts.* 4 ed. s.l.: Pearson Education Inc.

Kroenke, D.M., Auer, D.J. & Horan, B., 2010. *Database concepts.* 4th ed. ed. Boston, USA: Pearson.

Kroenke, D.M., Svendsen, E. & Horan, B., 2012. *MIS essentials.* 2nd International ed. ed. Boston, USA: Pearson.

Lankhorst, M., 2005. *Enterprise architecture at work.* Berlin, Germany: Springer.

Larman, C., 2002. *Applying UML and patterns.* 2nd ed. ed. Upper Saddle River, USA: Prentice Hall PTR.

Larman, C., 2005. *Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design and Iterative Development.* 3 ed. NJ: Prentice Hall.

Larman, C., 2007. *Applying UML and patterns.* 3rd ed. ed. Upper Saddle River, USA: Prentice Hall PTR.

Laudon, J. & Laudon, K., 2007. *Management Information Systems: Managing the Digital Firm.* 10 ed. s.l.: Prentice Hall (Pearson Education).

Laudon, K.C. & Laudon, J.P., 2009. *Essentials of management information systems.* 8th International ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Laudon, K.C. & Laudon, J.P., 2010. *Management information systems.* 11th Global ed. ed. Upper Saddle River, USA: Pearson.

Laudon, K.C., Laudon, J.P. & Horan, B., 2007. *Management information systems.* 10th ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Law, A.M., 2007. *Simulation modeling and analysis.* 4th ed. ed. Boston, USA: McGraw Hill.

Lejk, M. & Deeks, D., 2002. *An introduction to systems analysis techniques* . 2nd ed. ed. Harlow, England: Pearson Addison Wesley.

Lewis, J. & Loftus, W., 2007. *Java software solutions.* 5th ed. ed. Boston, USA: Pearson Addison-Wesley.

Liang, Y.D., 2007. *Introduction to Java programming: comprehensive version.* 6th ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Liang, Y.D., 2007. *Introduction to Java programming: fundamentals first.* 6th ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.

Liang, Y.D., 2011. *Introduction to Java programming: comprehensive.* Boston, USA: Pearson.

Liberty, J., Hurwitz, D. & Osborn, J., 2006. *Programming ASP.NET.* 3rd ed. ed. Bejing, China: O'Reilly.

Liberty, J. & MacDonald, B., 2008. *Learning C# 3.0.* Sebastopol, USA: O'Reilly.

Lippman, S.B., 1991. *C++ Primer.* s.l.: Addison-Wesley.

Lisin, M. & Joseph, J., 2006. *Microsoft SQL server 2005 reporting services.* Indianapolis, USA: Sams Publishing.

Liskov, B. & Guttag, J., 2007. *Program development in Java.* Boston, USA: Addison-Wesley.

Loney, K., 2004. *Oracle database 10g: the complete reference.* New York, USA: McGraw-Hill/Osborne.

Loney, K. & Bryla, B., 2005. *Oracle database 10g DBA handbook.* New York, USA: McGraw-Hill/Osborne.

Long, L., Long, N. & McPherson, J., 2005. *Computers.* 12th International ed. Upper Saddle River, USA: Pearson Prentice Hall.

Löwgren, J. & Stolterman, E., 2005. *Thoughtful interaction design.* Cambridge, USA: The MIT Press.

Lowy, J., 2009. *Programming WCF services.* Sebastopol, USA: O'Reilly.

Luftman, J.N., 2005. *Managing the Information Technology Resource: Leadership in the Information Age.* 1 ed. s.l.: s.n.

Macdonald, 2007. *Beginning ASP.NET 3.5 in C#: From Novice to Professional.* 2 ed. s.l.: s.n.

MacDonald, M., 2002. *User interfaces in C#.* Berkeley, USA: Apress.

MacLennan, J., Tang, Z. & Crivat, B., 2009. *Data mining with Microsoft SQL server 2008.* Indianapolis, USA: Wiley Publishing Incorporated.

Main, M., 2006. *Data structure and other objects using Java.* 3rd International ed. ed. Boston, USA: Pearson Addison Wesley.

Malaga, R.A., 2005. *Information systems technology.* Upper Saddle River, USA: Pearson Prentice Hall.

Mannino, M.V., 2007. *Database Design, Application Development and Administration.* 3 ed. s.l.:Pearson Education Inc.

Mannino, M.V., 2007. *Database design, application development, and administration.* 3rd ed. ed. Boston, USA: McGraw-Hill Irwin.

Mannino, M.V., 2009. *Datasbase design, application development, and administration.* USA: Michael V. Mannino.

Marakas, G.M., 2006. *Systems analysis and design.* 2nd ed. Boston, USA: McGraw-Hill Irwin.

Mayhew, D., 1992. *Principles and Guidelines in Software User Interface Design.* s.l.: Prentice Hall.

McFedries, P. & Clapp, C., 2005. *Business solutions.* Indianapolis, USA: Que Publishing.

McGovern, J., Sims, O., Jain, A. & Little, M., 2006. *Enterprise service oriented architectures.* Dordrecht, The Netherlands: Springer.

Mehlhorn, K. & Sanders, P., 2008. *Algorithms and data structures.* Berlin, Germany: Springer.

Meier, R., 2010. *Professional Android 2 application development.* Indianapolis, USA: Wiley Publishing Incorporated.

Merkow, M.S. & Briethaupt, J., 2006. *Information security.* Upper Saddle River, USA: Pearson Prentice Hall.

Mills, B., 2006. *Theoretical introduction to programming.* London, England: Springer Verlag.

Monk, E.F. & Wagner, B.J., 2009. *Concepts in enterprise resource planning.* 3rd ed. Boston, USA: Course Technology, Cengage Learning.

Monk, E. & Wagner, B., 2007. *Concepts in Enterprise Resource Planning.* 2 ed. s.l.: Thomson Course Technology.

Myerson, J.M., 2001. *Enterprise systems integration.* 2nd ed. Boca Raton, USA: Auerbach Publications.

Nielsen, J., 1994. *Usability Engineering.* s.l.: AP Professional.

Nielsen, J., 2000. *Designing Web Usability.* s.l.: New Riders Publishing.

O'Brien, J.A. & Marakas, G.M., 2008. *Management information systems.* 8th ed. Boston, USA: McGraw-Hill Irwin.

O'Brien, J.A. & Marakas, G.M., 2009. *Management information systems.* 9th International ed. Boston, USA: McGraw-Hill Irwin.

O'Brien, J.A. & Marakas, G.M., 2010. *Introduction to information systems.* 15th International ed. New York, USA: McGraw-Hill Irwin.

O'Leary, T.J., O'Leary, L.I. & Gordon, B., 2007. *Computing essentials 2007: complete.* Boston, USA: McGraw-Hill Irwin.

O'Leary, T.J., O'Leary, L.I. & Gordon, B., 2007. *Computing essentials: introductory.* Boston, USA: McGraw-Hill Irwin.

Olson, D.L., 2004. *Managerial issues of enterprise resource planning systems.* Boston, USA: McGraw-Hill Irwin.

Ostrowski, C. & Brown, B.D., 2005. *Oracle application server 10g web development.* New York, USA: McGraw-Hill/Osborne.

Oualline, S. & Denn, R., 2003. *Practical C++ programming.* 2nd ed. Sebastopol, USA: O'Reilly.

Oz, E. & Hennessy, K., 2009. *Management information systems.* 6th International ed. Boston, USA: Course Technology, Cengage Learning.

Panko, R.R., 2007. *Business data networks and telecommunications.* 6th ed. Upper Saddle River, USA: Pearson Prentice Hall.

Pearlson, K.E., Saunders, C.S. & Dodds, K., 2006. *Managing and Using Information Systems, A Strategic Approach.* 3rd ed. Hoboken, USA: John Wiley & Sons.

Peck, W., 2002. *Web menus with beauty and brains.* Indianapolis, USA: Hungry Minds.

Peck, W., 2003. *Great web typography.* New York, USA: Wiley Publishing Incorporated.

Perlson, K.E. & Saunders, C.S., 2006. *Managing and Using Information Systems – A Strategic Approach.* NY: Wiley.

Pfleeger, C.P. & Pfleeger, S.L., 2007. *Security in computing.* 4th ed. Upper Saddle River, USA: Prentice Hall.

Pidd, M., 2002. *Tools for thinking.* Chichester, England: Wiley.

Pohl, I. & Shanklin, C.J., 1993. *Object-oriented programming using C++.* Redwood City, USA: The Benjamin/Cummings Publishing Company Incorporated.

Powell, G. & McCullough-Dieter, C., 2007. *Oracle 10g database administrator.* Australia: Thomson Course Technology.

Power, D.J., 2002. *Decision support systems.* Westport, USA: Quorum Books.

Pratt, P.J., Adamski, J.J. & DiMassa, C., 2008. *Concepts of database management.* 6th ed. Boston, USA: Cengage Learning.

Pressman, R.S., 2010. *Software Engineering: A Practitioner's Approach.* 7th ed. s.l.: McGraw-Hill.

Price, J., 2004. *Oracle database 10g SQL.* New York, USA: McGraw-Hill/Osborne.

Quinn, M. J., 2006. *Ethics for the information age.* 2nd ed. Boston, USA: Pearson Addison-Wesley.

Rainer, R.K. & Turbanm, E., 2009. *Introduction to information systems.* 2nd International ed. Hoboken, USA: John Wiley & Sons.

Robertson, L.A., Doube, W. & Styles, K., 2004. *Simple program design.* 4th ed. Boston, USA: Thomson Course Technology.

Rob, P. & Coronel, C., 2002. *Database systems.* 5th ed. Boston, USA: Thomson Course Technology.

Rob, P. & Coronel, C., 2009. *Database systems.* 8th ed. Australia: Thomson Course Technology.

Rob, P. & Coronel, C., 2007. *Database Systems: Design, implementation and management.* 7 ed. MA: Course Technology.

Ross, J.W., Weill, P. & Robertson, D.C., 2006. *Enterprise architecture as strategy.* Boston, USA: Harvard Business School Press.

Ross, J.W., Weill, P. & Robertson, D.C., 2006. *Enterprise Architecture As Strategy.* NJ: Harvard Business School Press.

Royce, W., 2005. *Software project management.* Boston, USA: Addison-Wesley.

Ruby, S. et al., 2010. *Agile web development with Rails* . 3rd ed. Ragleigh, USA: The Pragmatic Bookshelf.

Satzinger, J.W., Jackson, R.B. & Burd, S.D., 2005. *Object-Oriented Analysis and Design – with the Unified Process.* Boston, USA: Thomson Course Technology.

Satzinger, J.W., Jackson, R.B. & Burd, S.D., 2009. *Systems analysis and design in a changing world* . 5th International Student Ed. ed. Boston, USA: Course Technology, Cengage Learning.

Savitch, W., 2012. *Java: An Introduction to Problem Solving and Programming.* 7th International ed. Boston, USA: Addison-Wesley.

Savitch, W., Mock, K. & Hirsch, M., 2009. *Problem solving with C++.* Boston, USA: Pearson Addison-Wesley.

Savitch, W., Mock, K., Horton, M. & Hirsch, M., 2012. *Java.* 6th International ed. Boston, USA: Pearson.

Schmuller, J. & Green, T., 2004. *Sams teach yourself UML in 24 hours.* 3rd ed. Indianapolis, USA: Sams Publishing.

Schneider, D.I. & Recter, P., 2004. *An introduction to programming using Visual Basic 6.0.* Updated 4th ed. Upper Saddle River, USA: Pearson Prentice Hall.

Schneider, S., 1988. *That's all folks.* New York, USA: Henry Holt and Company.

Schwalbe, K., 2006. *Information Technology Project Management.* 4 ed. s.l.: Thompson Course Technology.

Schwalbe, K., 2007. *Information technology project management.* 5th ed. Boston, USA: Thomson Course Technology.

Schwalbe, K., 2011. *Managing information technology projects.* 6th International ed. Australia: Course Technology, Cengage Learning.

Senft, S. & Gallegos, F., 2009. *Information technology control and audit.* 3rd ed. Boca Raton, USA: CRC Press Taylor & Franics Group.

Seref, M.M.H., Ahuja, R.K. & Winston, W.L., 2007. *Developing spreadsheet-based decision support systems.* Belmont, USA: Dynamic Ideas.

Sestoft, P. & Hansen, H.I., 2004. *C# precisely.* Cambridge, England: The MIT Press.

Shelly, G.B., Cashman, T.J. & Kosteba, L.A., 2006. *Web design.* 2nd ed. Boston, USA: Thomson Course Technology.

Shelly, G.B., Cashman, T.J., Rosenblatt, H.J. & Arnold, A., 2008. *Systems analysis and design.* 7th ed. Boston, USA: Thomson Course Technology.

Shelly, G.B., Cashman, T.J., Woods, D.M. & Dorin, W.J., 2007. *HTML.* 4th Comprehensive ed. Boston, USA: Thomson Course Technology.

Shelly, G.B. & Hoisington, C., 2009. *Microsoft Visual Basic 2008 for windows and mobile applications.* Boston, USA: Course Technology, Cengage Learning.

Shneiderman, B., 2004. *esigning the User Interface: Strategies for Effective Human-Computer Interaction.* 4 ed. s.l.: Addison Wesley Longham.

Shneiderman, B. & Plaisant, C., 2010. *Designing the user interface.* 5th International ed. Upper Saddle River, USA: Addison-Wesley.

Shuen, A. & St. Laurent, S., 2008. *Web 2.0.* Sebastopol, USA: O'Reilly.

Silberschatz, A., Korth, H.F. & Sudarshan, S., 2006. *Database system concepts.* 5th ed. Boston, USA: McGrawHill Higher Education.

Silberschatz, Korth & Sudarshan, 2006. *Database System Concepts.* 5 ed. s.l.: McGraw-Hill Higher Education.

Skiena, S.S., 2008. *The algorithm design manual.* 2nd ed. London, England: Springer.

Snedaker, S. & Hoenig, N., 2005. *How to cheat at project management.* Rockland, USA: Syngress Publishing Incorpoarted.

Sommerville, I., 2011. *Software engineering.* 9th International ed. Boston, USA: Pearson.

Sprankle, M., 2006. *Problem solving and programming concepts.* 7th ed. Upper Saddle River, USA: Pearson Prentice Hall.

St Amant, K., Still, B. & Reed, S., 2007. *Handbook of research on open source software.* Hershey, USA: Information Science Reference.

Stair, R.M. & Reynolds, G.W., 2008. *Fundamentals of information systems.* 4th ed. Boston, USA: Thomson Course Technology.

Stroustrup, B., 2008. *The C++ programming language.* 3rd ed. Reading, USA: Addison-Wesley.

Sugumaran, V. & Neidig, J., 2008. *Intelligent information technologies and applications.* Hershey, USA: IGI Publishing.

Taha, H.A. & Stark, H., 2007. *Operations research.* 8th ed. Upper Saddle River, USA: Pearson Prentice Hall.

Tanenbaum, A.S. & Steen, M.v., 2002. *Distributed systems.* Upper Saddle River, USA: Prentice Hall.

Taylor, J., 2004. *Managing information technology projects.* New York, USA: AMACOM American Management Association.

Thankarathinam, T., 2007. *Professional ASP.NET 2.0 databases* . Indianapolis, USA: Wiley Publishing Incorporated.

Thatcher, J. et al., 2002. *Constructing accessible web sites.* Berkeley, USA: glasshaus.

Thomas, B.M., Thomas, T.M. & Sandler, C., 2004. *The Art of Software Testing.* 2nd ed. Hoboken, New Jersey: John Wiley & Sons.

Thomsen, E. & Elliott, R., 2002. *OLAP solutions.* 2nd ed. New York, USA: Wiley Computer Publishing.

Townsend, J.J., Riz, D. & Schaffer, D., 2004. *Building portals, intranets, and corporate web sites using Microsoft servers.* Boston, USA: Boston, USA.

Treglia, D., 2002. *Game programming Gems 3.* Higham, USA: Charles River Media Incorporated.

Troelsen, A., 2007. *Pro C# 2008 and the .NET 3.5 platform.* 4th ed. Berkeley, USA: Apress.

Tucker, A.B., 2007. *Programming languages.* 2nd ed. Boston, USA: McGraw-Hill Higher Education.

Turban, E., Aronson, J.E., Liang, T.-P. & Sharda, R., 2007. *Decision support and business intelligence systems.* 8th International ed. Upper Saddle River, USA: Pearson Prentice Hall.

Turban, E. et al., 2008. *Electronic commerce 2008.* Upper Saddle River, USA: Pearson Prentice Hall.

Turner, J. & Stephens, M., 2002. *MySQL and JSP web applications.* Indianapolis, USA: Sams.

Ullman, J.d. & Widom, J., 2002. *A First Course In Database Systems.* 2nd ed. Upper Saddle River, USA: Prentice Hall.

Ullman, L. & Gulick, R., 2007. *PHP 5 advanced.* Berkeley, USA: Peachpit Press.

Ullman, L. & Gulick, R., 2008. *PHP 6 and MySQL 5 for dynamic web sites.* Berkeley, USA: Peachpit Press.

Vargas, A. & McPherson, J., 2004. *PowerPoint 2003: brief.* Upper Saddle River, USA: Pearson Prentice Hall.

Venugopal, S., 2007. *Data structures outside in.* Upper Saddle River, USA: Pearson Prentice Hall.

Visser, S. & Wong, B., 2004. *Sams teach yourself DB2 universal database in 21 days.* 2nd ed. Indianapolis, USA: Sams .

Wagter, R., van der Berg, M., Luijpers, J. & van Steenbergen, M., 2005. *Dynamic enterprise architecture.* Hoboken, USA: John Wiley & Sons.

Walters, B. & Tang, Z., 2006. *IT-Enabled Starategic Management – Increasing Returns for the Oragnization.* Hershey, USA: Idea Group Publishing.

Ware, C., 2004. *Information visualization.* 2nd ed. San Francisco, USA: Morgan Kaufmann Publishers.

Welling, L., Thomson, L. & AB, M., 2004. *MySQL tutorial.* Indianapolis, USA: MySQL Press.

Welling, L., Thomson, L. & Clapp, C., 2005. *PHP and MySQL web development.* 3rd ed. Indianapolis, USA: Sams Publishing.

White, R. & Downs, T.E., 2006. *How computers work.* 8th ed. Indianapolis, USA: QUE.

White, R. & Downs, T.E., 2009. *How computers work.* 9th ed. Indianapolis, USA: QUE.

Whitman, M.E. & Mattord, H.J., 2005. *Principles of information security.* 2nd ed. Boston, USA: Thomson Course Technology.

Whitman, M.E. & Mattord, H.J., 2009. *Principles of Information Security.* 3 ed. s.l.: Thomson, Course Technology.

Whitman, M.E., Mattord, H.J. & Helba, S., 2009. *Principles of information security.* 3rd ed. Boston, USA: Thomson Course Technology.

Whitten, J.L., Bentley, L.D., Randolph, G. & Dardan, S., 2007. *Systems Analysis & Design Methods.* 7th ed. Boston, USA: McGraw-Hill Irwin.

Wilkinson, J., 2009. *Foundations of IT Service Management Based on ITIL® V3.* London: Van Haren Publishing.

Williams, B.K. & Sawyer, S.C., 2007. *Using information technology.* 7th ed. Boston, USA: McGraw-Hill Irwin.

Williams, S., 2006. *Using Information Technology.* 7 ed. Irwin: McGraw-Hill.

Yee, R. & Moodle, M., 2008. *Pro web 2.0 mashups.* Berkeley, USA: Apress.

Zakas, N.C., McPeak, J. & Fawcett, J., 2007. *Professional Ajax.* 2nd ed. Indianapolis, USA: Wiley Publishing Incorporated.

Zak, D. & Cola, T., 2007. *Programming with Microsoft Visual Basic 2005.* 3rd ed. Boston, USA: Thomson Course Technology.

Zarnekow, R., Brenner, W., Pilgram, U. & Faessler, T., 2006. *Integrated information management.* Berlin, Germany: Springer.

# Index