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.