COMP4920 Senior Design Project II, Spring 2020
Advisor: Mehmet Ufuk Çağlayan

XYZAPP: XYZ Application
Design Specifications Document

Revision 2.0
09.03.2020

By:
Mehmet Uzunkavakaltındayataruyuroğlu 1, Student ID: 123456781
Mehmet Uzunkavakaltındayataruyuroğlu 2, Student ID: 123456782
Mehmet Uzunkavakaltındayataruyuroğlu 3, Student ID: 123456783
Mehmet Uzunkavakaltındayataruyuroğlu 4, Student ID: 123456784

Revision History

Revision	Date	Explanation
1.0	04.12.2019	Initial high level design
1.1	21.02.2020	- High level design in Section 2 has been refined - Detailed design of all classes are added to Section 3 Note that revisions in the document body better be marked clearly, for example by graying them
2.0	09.03.2020	- High level design in Section 2 has been refined - Detailed design of all classes are revised in Section 3 - Testing design in Section 4 has been reviewed and corrected for typographical errors. Note that revisions in the document body better be marked clearly, for example by graying them

Table of Contents

1. Introduction

Note that the text in different sections of this document contains hints for you to write a successful design specifications document and also some examples. This document is NOT a design specifications document TEMPLATE. Sometimes, it may be confusing to decide which text is a hint and which text is an example. Writing a successful design specifications document for a large and complex system is usually quite complicated and there is no single way of doing it.

The purpose of the project is to develop the XYZ Application in C++ and in LINUX and MySQL environment, to do mainly the following.

1. ……..
2. A short list of main functions of your system
3. ……..

The design is based on XYZAPP Requirements Specification Documant, Revision 3.5, in file Caglayan-XYZAPP-RSD-2019-12-20-Rev-3.5.doc [1]. The design process used produce the design conforms to organizational specifications given in [2]. The notation used in this document to describe the design of XYZ Web Application is mainly UML and conforms to organizational specifications given in [3]. The ISO and TSE standards in [4], [5] and [6] are extensively used during the design.

The software architecture and overall high-level structure of XYZ Application are given in Section 2 and design details of all application functions and the user interface in terms of methods of all classes are given in Section 3 of this document.

2. XYZAPP System Design

Note that if your project also includes the development of some hardware, then you will have at least two main subsystems. In this case, you need to give the design specifications of hardware and software subsytems in different sections of this document. For example, a statement such as:

XYZAPP system consists of two main subsystems, namely,

- XYZAPP Hardware Subsystem
- XYZAPP Software Subsystem

could be placed here. Also, provide a figure here showing the overall structure of hardware and software subsystems and their relation to each other, that is, interfaces among subsystems. This figure could be drawn as a set of UML Package and/or Component Diagrams.

3. XYZAPP Hardware Subsystem Design

Note that Section 3 will exist only if your project also includes the development of some hardware. Otherwise, you just give the design of Software Subsystem as it is the Software System itself.

3.1. XYZAPP Hardware Subsystem Architecture

- Overall hardware architecture, if any.
- Discussion and justification of design decisions, if any.

3.2. XYZAPP Hardware Subsystem Structure

- Overall high-level structure of the hardware subsystem in terms of UML Package/Component Diagrams
- Discussion and justification of design decisions, if any.
- Subsection numbering such as 3.2.1, 3.2.2. etc. may be necessary if this section is rather long, covering multiple pages of the document, for a complex hardware subsystem.

3.3. XYZAPP Hardware Subsystem Environment

- For example: You develop some hardware subsystem with USB interface, then your hardware subsystem will interface other hardware/software systems.
- Detailed description of hardware, system software and middleware, if any, for which the XYZ Hardware Subsystem has interfaces
- Complete specification of target specification language, year, version, etc to be used in implementation and testing. i.e. VHDL variations in spcific ASIC development
- Any software tools, especially if not specified in [2].
- Subsection numbering such as 3.3.1, 3.3.2. etc. may be necessary if environement specification is rather long, covering multiple pages of the document, for a complex hardware subsystem.

4. XYZAPP Software Subsystem Design

Since most of the 4920 projects are software only projects, yo will not have a Section 3 to specify hardware design, you just give the design of Software Subsystem as it is the Software System itself.

4.1. XYZAPP Software System Architecture

- Overall software architecture of the software system or software subsystem
- System/subsystem architecture specification is usually a short statement, identifying the architectural style of your system/subsystem.
- We do not have a UML diagram for architecture specification. İt is just some text, as a statement.
- Architectural style usually identifies how subsytems of a system are organized and are related to each other.
- You must be very carefull not to confuse the architecture and structure of a system/subsystem. In many technical documents, some people use the term "Achitecture of Bla Bla System" then give a diagram or figure actually showing the structure of that system. Don't do that, it shows your ignorance of the subject.
- If you can, write a few more sentences to explain why you have decided to use this architecture and give a short discussion. This is called the justification of a design decision and it must be done for all design decisions in this document.
- Example architectural styles of hardware and software system/subsystems are as follows.
  - Monolithic, that is no architecture at all. Example: one large single program with many statements
  - Layered Architecture: An architectural style commonly used in many systems. Examples are ISO OSI Reference Model and TCP/IP (or Internet) Reference Model. Many software systems will have layered architecture. A subsystem may have interfaces only to subsystems above and below it. In 4910-4920 projects, your system architecture will probably be layered. .
  - Hierarchical Architecture: An architectural style commonly used in many systems. An extension of layered architecture.
  - MVC: Model-View-Controller. Also, MVP, MVVP etc. These are also called architectural design patterns.
  - Multi-Process Architecture: Subsystems are concurrent processes in a shared memory environement.
  - Distributed (Cloud) Architecture: Subsystems are concurrent processes in a network environement, Subsystems and data are geographically distributed.
  - Client/Server Architecture: A special, less restricted form of Distributed Architecture.
- Further discussion of possible architecture specifications is out of scope of this document.
- In large and complex systems, there may be an overall system architecture but each subsystem itself may have a different architecture. A good example will be a large and complex system with client/server architecture. Client side may be an ordinary web browser with simple http/HTML functions, but could also be some specialized application able to execute active code. In this last case, the client will have some architecture. Server side may be a simple application with some SQL capabilities (almost no architecture) but could also be a large and complex system with distributed (cloud) architecture, for example a multi-process application with replicated data in a networked cluster environement.

4.2. XYZAPP Software System Structure

- Overall high-level structure of the software system in terms of packages/components and classes (UML Package/Component and Class Diagrams. If UML packages are used, they must provide a high level overview of classes they contain)
- Discussion and justification of design decisions, if any.
- Subsection numbering such as 4.2.1, 2.2.2. etc. may be necessary if this section is rather long, covering multiple pages of the document, for a complex software system.

4.3. XYZAPP Software System Environment

Note that if your project also includes the development of some hardware, than you do not specify the design of your hardware here.

- Detailed description of hardware, system software and middleware, if any, on which the XYZ Application is designed to execute.
- Complete specification of target programming language, year, version, etc to be used in implementation and testing.
- Any software tools, especially if not specified in [2].
- Perhaps a UML Deployment Diagram could be placed here.
- Subsection numbering such as 4.3.1, 4.3.2. etc. may be necessary if environement specification is rather long, covering multiple pages of the document, for a complex software system.

5. XYZAPP Software System Detailed Design:

The detailed design section should be organized in terms of classes of packages/components for a complex software system, for example Section 5.1 will contain all classes/methods of package/component 1, that is subsystem 1, Section 5.2 all classes/methods of package/component 2, that is subsystem 2, etc.

5.1. XYZAPP Main Module/Class

- Main module/class is the main program or main process for developments in procedural languages such as Cobol, Fortran, Pascal, C, and is the main (control) class for developments in object oriented languages such as C++, Java, C#.
- Detailed design specifications of all methods and internal data in terms of UML Class and Activity Diagrams
- Discussion and justification of design decisions, if any.

5.2. XYZAPP Subsystem S1 Classes

5.2.1. Class S1-C1

- Class S1-C1 is a class with a number of methods, in case the design target is development in an object oriented language, such as C++, Java, C#.
- Detailed design specifications of all methods, that is, class internal data in terms of UML Class Diagram(s) and method logic as UML Activity Diagrams. If method logic is trivial, a single sentence may be sufficient.
- Detailed design specifications of each user interface, if any, including displays, reports generated, etc are here.
- Discussion and justification of design decisions, if any.
- Subsection numbering for each method, such as 3.2.1.i for Class S1-C1 Method i may be necessary if method specification is rather long, covering multiple pages of the document, for a complex class. Otherwise, a highlihted class name and method name as section name, as shown below, could be sufficient.
- If detailed design is for developments in procedural languages such as Cobol, Fortran, Pascal, C, etc., then instead of Class C1, a module specification could be provided. A module in this sense is a subprogram (procedure, subroutine, function etc. depending on the target procedural language).

Class S1-C1 Method M11

- Method M11 input/output parameters, parameter values and their meanings
- Method M11 internal data specification as UML Class Diagram(s)
- Method M11 internal logic/algorithm/operation specification as a UML Activity Diagram (if trivial, a single sentence)

Class S1-C1 Method M12

- As in Method M11 specification

……

Class S1-C1 Method M1n

- As in Method M11 specification

5.2.2. Class S1-C2

- As above in Section 3.2.1, then Class C3 will follow as Section 3.2.3, C4 will follow as Section 3.2.4, etc.

5.2.3. Class S1-C3, etc.

- …….. similar to former section.

5.3. XYZAPP Subsystem S2 Classes

5.3.1. Class S2-C1

- Class S2-C1 is a class with a number of methods, in case the design target is development in an object oriented language, such as C++, Java, C#.
- …….. similar to former section.

Class S2-C1 Method M11

- …….. similar to former sections

……

Class S1-C1 Method M1n

- …….. similar to former sections

5.3.2. Class S2-C2

- …….. similar to former sections

5.3.3. Class S2-C3, etc.

- …….. similar to former sections

5.4. XYZAPP Subsystem S3 Classes

- …….. similar to former sections

5.n. XYZAPP Subsystem Sk- Common Infrastructure Classes

- Note that you may have more than one subsystem as common ınfrastructure.
- Common infrastructures classes are to support the main functionality of the application, for example, a common *Log* or common *Sort-2D-Table* class used by all classes
- Section 3.i. is structured similar to Section 3.2., where each Section 3.i.j. gives the detailed design specifications of class j.

6. Testing Design

- In this course, we do not very much emphasize testing design, therefore we do not provide the details of test case design to test each, module, each method, each class, or provide the details of integration test design
- Some general test remarks could be made here to remind that testing design is here.

References

1. Reference to RSD for which this design is made
2. Reference to organizational design process procedure document(s), or a generic design process procedure document(s)
3. Reference to organizational design product specification document(s), or a generic design product specification document(s)
4. Other references to additional documents, like other internal organizational documents, software project management documents, software design tool documents, etc
5. References to additional bibliographic sources, like professional books, textbooks, handbooks, patents, standards, technical reports, journal/conference papers, etc.