CSE053 SE Project Plan

(Supplementary materials for the class project)

Software Engineering Laboratory
Dept. of Computer Engineering, Yeungnam University
ysseo@yu.ac.kr
Fall 2023

Contents

Abstract

This document is a tutorial on CSE053 SE class project. In this document, we will explain the project guideline such as schedule and evaluation criteria for your class project. Every student in CSE053 class is expected to read this tutorial carefully and perform their own project based on the guidelines provided in this document. If you have a specific question, and you want an immediate answer, email will always bring quicker result.

1. Introduction

From September, each student in CSE053 class should prepare to perform the class project. The major objective of the course project is for you to apply software engineering principles, methodologies and tools in the creation of a significant piece of software. This class project focuses on methodologies and tools for application development using object-oriented paradigm.

As notified in class hour, all course projects will be required to use the Waterfall model and the overall process of project execution is designed with considerations of the structure of the Waterfall model. The process of class project is depicted in Figure 1.

As shown in Figure 1, to complete the SE class project, you should perform total 5 phases (Requirement analysis, design, implementation, testing, and demonstration) and submit 4 results. Each phase of the class project will be explained in section 2. The system which is to be developed during the project is decided by you. After deciding the system to be developed, you will analyze, design and implement it. In section 3, we will explain the schedule of the class project in detail. Evaluation criteria for class project will be discussed in section 4 and some miscellaneous points on class project will be described in section 5. You can find the references in section 6.

2. Project phases

In this section, we describe each phase of the project process and its associated report.

2.1 Requirement analysis

The essential purpose of requirement analysis is to aim development toward the intended correct system. This is achieved by describing the system requirements well enough so that an agreement can be reached between the customer and the system developers on what the system should and should not do. Through requirement analysis phase, you should prepare the SRS (Software Requirement Specification) that includes the followings:

- Project name, description
- Requirement description
- Traceability
- Priority

The details above each item can be found on the class materials and you can refer to format of the *SRS* report.

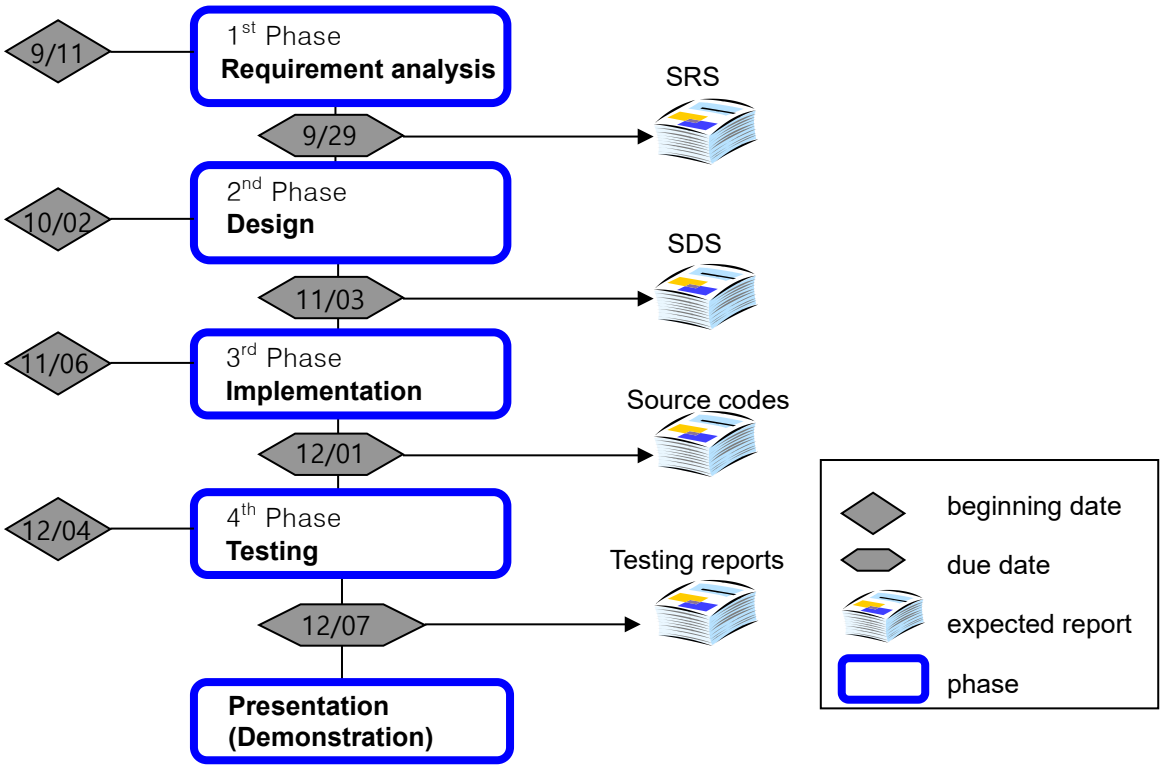

Figure 1. The process for executing a SE class project

2.2 Design

The goal of design phase is to implement the requirements specified from previous phases. The design phase consists of two activities. First is *system design* and the other is *object design*.

System design activity includes architectural design and appropriate allocation of requirement specified from previous phases based on the identified system architecture. The purpose of architectural design is to outline the design and deployment models by identifying followings:

- Node and their network configurations
- Subsystems and their interfaces
- Architecturally significant design classes (e.g. active classes)
- Generic design mechanisms that handle common requirements, such as the special requirements on persistency, distribution, performance and etc.

The details on architectural design are found in [7]. You can get more introductory contents on software architecture in [1, 4, 6, 8, 9].

When designing the architecture of your software system, if possible, not only the simple relationship among subsystems which you identified, such as inclusion and use dependency, but also the layered structure of the subsystems is strongly encouraged. You can freely decide the number of layers and their categories according to the characteristics of your software system, but it would be better to refer the layered structures presented for example in the class materials.

In the object design activity, you should develop the *design class*. A design class is a seamless abstraction of a class or similar construct in the system's implementation. This abstraction has the following characteristics:

- The language used to specify a design class is same as the programming language. Consequently, operations, parameters, attributes, types and relationships are specified using a chosen programming language syntax.
- The class model in design phase is concrete and detailed that a CASE tool (such as *StarUML*) can generate its full template code in the form of the chosen programming language.
- A design class can be active, implying that object of the class maintain their own thread of control and run concurrently with other active objects. Or design classes are normally not active, implying that their objects run in the address space and under the control of another active object.

After developing each design class, you should specify *some state machine diagram* for *important design class*. All of those *state machine diagrams* must be consistent with the class diagrams and the interaction diagrams within the design model.

In addition, you should build use case realization model using UML *sequence diagram* or *communication diagram*. A use case realization design is more detailed communication within the design model that describes how a specific use case is realized, and performed in terms of design classes and their interacting design objects. A use case realization design should have a textual flow-of-events description, class diagrams that depict its participating design classes, and interaction diagrams that depict the realization of a particular flow or scenario of the use case in terms of design classes and their objects.

Consequently, the SDS (Software Design specification) should include the followings:

- Introduction
- Class diagram
- Class diagram description
- Sequence diagram per each use case
- Sequence diagram description
- State machine diagram
- State machine diagram description
- Implementation requirements

2.3 Implementation

In this phase, you should implement your project using Java technologies. Your implementation does not need to be complete, but major scenarios in each requirement should be realized in your implementation model.

The implementation reports which you will submit consists of *installation and user manuals*. In this phase, you will submit all source documents and executables of your project software to TA. We do not impose any formation constraints on these manuals. However, *readability* and *understandability* are key evaluation criteria of your implementation reports (DO NOT PRINT OUT YOUR SOURCE CODE COMPLETELY).

Remember that the design model should be consistent with your implementation model. Otherwise, your implementation project cannot get good credits!

2.4 Testing

Software testing is a process of executing a program or application with the intent of finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application or product: Meets the business and technical requirements that guided its design and development. In this project, we will conduct the system testing by using SRS document. The testing report should include the followings:

- Input variables to test the function
- Test cases
- Test results

2.5 Demonstration

After completing implementation of each project, we will have a demonstration for your project. Prepare to demonstrate your system for other students for about 20 minutes. During that time, you should present the result of your project, and to let audiences to know that you have implemented a correct prototype system.

For your system's effective demonstration, preparation of some presentation files made by some tools like MS Powerpoint, are strongly encouraged and please submit to instructor to get better score of your demonstration.

3. Project Schedule (tentative)

The overall project schedule is shown in Table 1.

Table 1. SE Project Schedule

Date	Contents
09/11	1st project (Requirement analysis phase) begins
09/29	SRS is finished
10/02	2nd project (Design phase) begins
11/03	SDS is finished
11/06	3rd project (Implementation phase) begins
12/01	Implementation report is finished
12/04	4th project (Testing phase) begins
12/07	Testing report is finished
12/04~	Project demonstration (presentation)

You must remember that the delay penalty will be applied rigorously. Thus, if your report is later than 7 days from the due, your score of the report will be zero.

If there are problems of your documents, you should submit reports of current and previous phase for checking traceability.

4. Evaluation of Project

This section addresses our evaluation policy. Each phase of the project has some deliverables which are assigned a certain weight and a specific due date. All due dates are identified as important dates as explained in section 3. *You must deliver on time*. Late assignments will be penalized.

We evaluate each report (requirement analysis, design, implementation, and testing) by the corresponding evaluation model. The final grade will be compiled from the course project and examination. The score of each report ranges from 0 to 100.

Some checklists of evaluation models are shown as follows:

- SRS
  - Does the report conform to the given form in the appendix?
  - Does the document include the table of contents and page numbers?
  - Does the introduction include objectives and goals of the target software system?
  - Is the report readable?
  - Is the report consistent?
  - Is the report unambiguous?
  - Is the report complete?
  - The diagrams used in the report conform UML specification?

- SDS
  - Is the use-case model structured well?
  - Is each of the analysis classes explained well?
  - Are the authors mentioned about well by section?
  - The diagrams used in the report conform UML specification?
  - Are the attributes in the design classes identified well?
  - Is the sequence diagram consistent with the class diagram?
  - The diagrams used in the report conform UML specification?

- Implementation report
  - Is the program easy to install?
  - Does user manuals provide sufficient install information?
  - Does implementation conform to design model?
  - Is your presentation performed effectively

5. Miscellaneous

In this section, we describe some miscellaneous considerations on your successful completion of the class project.

- The precise due date time is 23:59 of due date and NO DELAY IS ALLOWED! (Our penalty strategy for delay is very terrible).
- Good implementation with more functionalities could gain more optional credits.
- Readability, understandability and clarity of your report is very important.
- Refer the useful textbook[3] and try to confirm UML specification when drawing UML diagrams.
- We strongly recommend StarUML as project's standard UML modeling tools. You can use other UML modeling tools if you want.
- The final presentation is very important to show your whole artifacts during this semester.

6. References

[1] L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice (2nd Edition). Addison-Wesley, 2003.

[2] A. Cockburn. Writing Effective Use Cases. Addison Wesley, 2000.

[3] Alan Dennis, Barbara Haley Wixom and David Tegarden, Systems Analysis and Design with UML: An Object-Oriented Approach, 5th edition, Wiley, 2016.

[4] D. Garlan. Software architecture: A roadmap. In Proc. Of Int'l Conference on Software Engineering.

[5] http://www.omg.org/spec/UML/, Object Management Group. OMG unified modeling language specification version 2.5, 2015.

[6] C. Hofmeister, R. Nord, and D. Soni. Applied Software Architecture. Addison–Wesley, 1999.

[7] J. Jacobson, G. Booch, and J. Rumbaugh. The Unified Software Development Process. Addison-Wesley Longman Inc., 1999.

[8] N. Medvidovic and R. N. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages. IEEE Trans. Software Engineering, 26(1):70–93, Jan 2000.

[9] M. Shaw and D. Garlan. Software Architecture: Perspectives on an Emerging Discipline. Prentice–Hall Inc., 1996.