SE: 505 Software Project Lab II

Overview and Guideline

(version 1)

Prepared by

SE 505 Course Coordinators

Dr. Mohammed Shafiul Alam Khan

Amit Seal Ami

Md. Saeed Siddik

Date: February 26, 2018

Institute of Information Technology

University of Dhaka

Table of Contents

Overview 1
General Requirements 1
Activities 1
    Proposal Submission 1
    Midterm 1 2
    Midterm 2 2
    Final Defense 2
Other Roles 3
Stakeholders 3
Marks Distribution 3
Appendix One: SRS Components 4
    Inception 4
    Elicitation 4
    Scenario-Based Model 4
    Data Model 4
    Class-Based Model 4
    Flow-Oriented Model 4
    Behavioral Model 4

Overview

Software engineering ensures quality by following rigorous processes throughout the Software Development Life Cycle (SDLC). Requirement engineering, or the engineered process of collecting, analyzing, specifying and validating requirements is considered the most crucial of all these processes as it determines the functionalities of the software.

The goal of SE 505: Software Project Lab 2 (SPL2) is to ensure that students follow and apply the processes of requirement engineering while creating a software. Therefore, it is necessary that students choose and propose a project that will allow them to apply requirement engineering within the time frame of the project. Moreover, it is also encouraged and suggested that students choose projects with ample scope to gather requirements from stakeholders outside their comfort zone, i.e. from IIT, relatives, and/or friends.

General Requirements

All the projects will have to follow several general requirements to ensure that the goal of SE 505: SPL2. The general requirements are provided as follows:

- The project will require groups to discuss with several types of stakeholders to gather requirements and analyze, and/or to analyze existing products that provide facilities similar to the proposed proposal.
- The project must have implementation of the major functionalities identified in the requirement analyze phase.
- The project have to be maintained through publicly accessible distributed version control system.

Activities

Students will have to go through several activities within the time frame of this project.

Proposal Submission

Within the third week of the semester, students will form a group of two and will submit project proposal to the coordinators of SPL2. The proposal will cover how the project will incorporate requirement engineering and how the group plans to execute requirement engineering to prepare Software Requirement Specification Document. The contents will include the following deliverables:

a) Introduction

b) Scope of the project

c) Motivation

d) Work Plan

The proposal will be evaluated by the committee.

Midterm 1

Within the 7th week of the semester, the groups will report on requirements gathered and work progress. The deliverables are:

a) Publicly accessible URL of distributed version controlled repository,

b) Requirement collection notes (meeting minutes, survey questionnaire for example)

c) Analysis report

d) Specific requirements in SRS report

The evaluators will provide feedback on requirements gathered till this stage. If needed, students will rectify the requirements and present in Midterm 2. The committee members and managers will do the evaluation. SRS components are provided in Appendix 1.

Midterm 2

Within the 10th week of the semester, student groups will present the current work progress related to implementation. If necessary, groups may redefine the scopes of the implementations with the permission of supervisors and incorporate changes based on feedback midterm 1. It is mandatory to include a change log within the SRS document for any changes. The deliverables are:

a) SRS Report with change log (if applicable)

b) Work progress report

c) Updated repository details

The committee members and managers will be conduct the evaluation.

Final Defense

The final defense will take within the duration of semester exams, i.e. within the 14th week of the semester. The groups will present the implementation of the project. Furthermore, they will have to justify how the implementation aligns with the prepared SRS. The final deliverables will include:

a) Updated version controlled repository

b) User manual of the implementation, which may include installation details, setup, usage, description and features. Instructions necessary to build and run the project from source are mandatory.

The evaluation will be done by the committee members only and will focus on implementation, demonstration and code review.

Other Roles

Supervisor will supervise the overall activities of the student. The evaluation by supervisor will have to be done based on overall project completion and continuous effort.

Managers will evaluate the students based on continuous progress, specially focusing on the requirements gathering segment.

Stakeholders

Every project must have selected stakeholders, from whom students will collect their project requirements. Stakeholders are the persons, organizations or social groups who are directly involved to the project. Corresponding project supervisor and group members should not be considered as the stakeholders of that project.

Marks Distribution

There are six marking factors, which are:

a) Proposal — 5%

b) Midterm 1 — 15%

c) Midterm 2 — 15%

d) Supervisor — 30%

e) Managers – 15%

f) Final Defense — 20%

Plagiarism at any stage will result in gaining F grade, or severe grade deduction. Any student receiving re-submission at any stage will get at most 80% marks during the re-submission evaluation.

Appendix One: SRS Components

Inception

Identify the stakeholders and explore their different point of views about the software. Finally, use these different stakeholder requirements to key out the common, conflicting and final requirements of the software.

Elicitation

Translate the needs of customers into technical requirements for the software. Use Quality Function Deployment (QFD) technique to categorize the requirements into normal, expected and exciting requirements. Describe how the system functions and features will be used by different classes of end users by providing usage scenario (i. e., user story) of the software.

Scenario-Based Model

In this model, the system is to be described from the users' point of view. Develop and describe the use cases along with the use case diagrams. These use cases will describe the system's behavior and the ways users interact with it, under various conditions. Provide activity and swim-lane diagrams that will supplement the use cases by providing the graphical representations of every flow of interaction in clear and concise manner.

Data Model

[Conditional: software requirements include the need to create, extend, or interface with a database, or a complex data structure needs to be constructed and manipulated]

Define the data objects that are processed within the system. Construct E-R diagram and data schema for showing the relationships between the data objects and other pertinent information to the relationships.

Class-Based Model

[Conditional: Designed for Object-Oriented Programming (OOP) paradigm]

Class-based modeling represents the objects that the system will manipulate, the operations that will be applied to the objects, collaborations between the classes, etc. Thus, identify the analysis classes, specify their attributes and operations, and provide the Class Responsibility Collaboration (CRC) diagram.

Flow-Oriented Model

This model provides insight into system requirements and flow. For this, provide Data Flow Diagram (DFD) of the system. The DFD takes an input-process-output view of a system. It represents the way data objects flow into the software, are processed and the resultant data objects flow out of the software.

Behavioral Model

[Conditional: Designed for Object-Oriented Programming (OOP) paradigm]

The behavioral model indicates how software will respond to external events. Provide state diagrams to represent active states for each class based on the events (triggers). Provide sequence diagram to indicate how events cause transitions from object to object.