Project Document for
Analysis of Software Artifacts

Somesh Jha
Computer Science Department
University of Wisconsin
Madison, WI 53703.
email: jha@cs.wisc.edu

Project is a large component of this class. This document gives the overview of the project and responsibilities of each student in the project.

1 Overview

The entire class will be divided into three teams. Each team will be assigned a project and a mentor. *Successful delivery of the project is very important*. A large portion of the project grade will be based on its successful completion.

1.1 Mentors

Each group will be assigned a mentor. Mentors are graduate students who have significant amount of programming experience. Mentors will have the following duties in the project:

- Mentors will attend project meetings and make sure that the group is correctly following the software process.
  Note: Each group is required to have an *hour long weekly* project meeting. Instructor should be informed about the group meeting times. The instructor will occasionally show up at these meetings and observe participation of various group members.
- Mentors will also review various documents and provide feedback.
- Mentors will monitor the progress of the project.
- Mentors will *NOT* actively participate in the actual work for the project. Mentors are there for guidance and feedback only.
- Mentors will also provide feedback about each group member to the instructor.

2 Background

You should immediately buy the following book (henceforth referred to as *the UML book*) and start reading it cover to cover. You should finish the book in the next 3 weeks.

- *The Unified Modeling Language User Guide (The Addison-Wesley Object Technology Series)*, G. Booch and Ivar Jacobson and James Rumbaugh, Addison-Wesley, October, 1998.

Please note the following important points about the project.

- All the development in the project has to be done in `Java`.
- Moreover, the development will be done using the tool `Together`. The tool can be downloaded from the following website:

  `www.togethersoft.com`.

  I will mail the license for `Together` to the class. You should immediately download the tool and start familiarizing yourself with it. I will also download `Together` and put it in a public director (thanks Jim Gast for the suggestion).
- Documents and code should be maintained using version control.
- Constant communication with the mentors is crucial.

3 The Software Engineering Process

You should adhere to the phases of the software engineering process described in the subsequent subsections. It is your responsibility to find out (from the UML book) what formalism should be used at each phase. Notice that you might have to revisit a phase later in the project. For example, system design might have to be revised during system testing, e.g., a serious flaw might be found during the testing process. It is crucial to keep the various document "in sync" with the system. There should be a document associated with each phase and every group member should participate at each phase. The precise logistics is left upto the group and its mentor.

Highly recommended: Each group should get one copy of the following book. Pfleeger's book has a good description of each of these phases.

- *Software Engineering, Theory and Practice (Second Edition)*, Shari Lawrence Pfleeger, Prentice-Hall, 2001.

3.1 Requirements elicitation, analysis and definition

Requirements capture what the customer wants the system to do. A *requirement* is a feature of the system or a description of something the system is capable of doing in order to fulfill the system's purpose. Typically, requirements are elicited through numerous interviews with the customer. After the elicitation step, the requirements are defined and then analyzed. In this class, the mentors will act as customers.

3.2 System design

This phase uses the requirements to construct a high level or architectural design. Since the system design is reviewed and approved by the customer, it should be at a high level, e.g., generally focusing on the overall architecture, functionality, and external interface.

3.3 Program design

This phase refines the system design and produces a more concrete program design, i.e., various objects, threads, protocols, and message sequence charts. This level of design should be detailed enough so that it can be shipped to a team of programmers to implement the system.

3.4 Program implementation

The program design is used as a basis for an actual implementation. It is very important to keep the implementation and all other design and requirements document "in sync".

3.5 Unit testing

In the testing phase a programmer makes sure that each program unit behaves correctly. For example, each object should behave according to its advertised API. Code walkthroughs and inspections are also an important part of unit testing.

3.6 Integration testing

Unit testing ascertains that each unit is working correctly. Integration testing verifies that the program units work correctly together and can be assembled into a working system.

3.7 System testing

The objective of system testing is very different from unit and integration testing. The objective of system testing is to ensure that the system does what the customer wants to do.

3.8 System delivery

Assuming that all the phases given above are successful, the system is delivered to the customer along with the appropriate documentation. Documentation is also reviewed by a team and "tested" for compatibility with the system.

3.9 Maintenance

This generally means setting up a system for reporting bugs and incorporating facilities for fixing the system. Guidelines for regression testing and releasing updates are also designed.

4 Grading

Grading for the project will be based on the following criteria:

- Successful delivery of the project.
- Feedback from the mentors.
- Inspection of various documents and code.
- Participation in the project.