

CSSE2003

Software Engineering Studio

Team Assignment 2

Semester 2, 2009

School of Information Technology and Electrical Engineering
The University of Queensland

The purpose of assignments is to help students master the material in CSSE2003 (learning by doing). Assignments serve the additional purpose of demonstrating students' mastery of the material, thereby establishing a basis for evaluation.

- Goals:**
1. Give students experience investigating a large body of Java code written by other people.
 2. Provide an opportunity to reverse-engineer an existing Java program and generate UML design documentation.
 3. Require students to use the Ant build tool to manage the source code.
 4. Provide a significant task for cooperative learning.
 5. Encourage students to reflect on their learning activities and relate software design theory and practice.

Deadline: 5 pm, Friday 25 September 2009 (week 9)

What you have to do

This assignment will be continued in Assignment 3. Assignment 2 represents the first part of an intensive investigation of the source code of your selected Java program. It offers an opportunity to receive feedback on your approach including your plan for additional investigation as part of Assignment 3.

There are four deliverables for this assignment:

- 1. Source code description.** The team should first identify any existing design and code documentation for their Java program. Using any existing documentation as a base, the team should construct a design document for the program. The document should incorporate an overview of the program's structure including a complete list of classes (with a short description) and explain any structuring into packages. If the program relies on external libraries or other systems, these should be documented. Note: for long lists of classes and/or libraries, use an appendix.

Since some programs are much larger than others, teams may choose to focus their detailed description on a coherent subset of the overall source code. A rationale for the selection of the subset should be provided. As a very rough guide to the size of a subset for Assignment 2, we anticipate teams will focus on about 12 to 16 classes or about 2,000 to 3,000 lines of code (not counting comments).

UML class diagrams should be constructed to capture the selected classes and the relationships between them. Not all attributes and methods need to be included for each class – inclusion should be on the basis of assisting understanding. UML sequence diagrams are not required in this assignment, but they may be used where they assist the documentation goal.

Where teams find instances of known software design patterns, the patterns should be documented by showing how the program classes map to the generic pattern participants. The role of the pattern within the program should also be explained.

Where teams find instances of poor or questionable design, the weaknesses should be identified and possible improvements proposed. These instances may also be discussed in the plan for future research (see deliverable 4).

As a guiding principle, construct a design document that helps a programmer who is unfamiliar with the program understand its internal structure and behaviour.

2. **Ant build evidence.** The assignment is to include evidence of the team's use of Ant to build the program from source. As a minimum, your script should produce a single executable jar file. The evidence should include both the script itself and output from its execution on non-trivial targets. Extra documentation explaining the build script is also expected.
3. **Team learning journal.** The team is to keep a journal of their activities, experiences and ideas over the course of this assignment (the journal will be continued in Assignment 3). Learning requires integrating theoretical knowledge from lectures and reading with practical skills. The team learning journal provides an opportunity to "keep track" of your learning and experiences. It can be used to track the progress of the team but it should be much more than just a factual record of "who did what". The team should engage in regular sessions that require all team members to reflect on their activities, knowledge, and feelings. The team can use these reflective sessions to guide subsequent actions. A size limit of 5,000 words applies to this component of the assignment (note: this is an upper limit, not a requirement; it represents 1,000 words per week).

Use the following questions as a guide for what to include in your team journal.

- a. How, in your opinion, does the theory of software design help with the practical task of understanding the internal workings of a large software system?
 - b. What strategies are you using to understand the source code of your team's Java program?
 - c. What resources have you found useful? (How and why).
 - d. What mistakes has the team made and how will you avoid making the same mistakes in the future?
 - e. What do you now know or understand that you didn't know or understand at the beginning of the semester?
 - f. For your team, to what extent have the goals of this assignment been achieved?
 - g. What do you think you need to know to make further progress in understanding software design?
4. **Plan for future research.** The team should prepare a plan for the next stage of investigation of their program. While this plan will not be binding, it represents an opportunity to influence the direction of your Assignment 3.
 - a. Within the limited time available for Assignment 2, it is unlikely that all the desired design documentation can be generated. The plan should propose how the design documentation generated in this assignment will be extended in Assignment 3. The plan should nominate a set of classes and/or packages that will be investigated in detail.
 - b. The plan should detail some proposed extensions, improvements and/or refactorings to the Java program that the team feels can be implemented in Assignment 3. These should be described as precisely as possible without getting into implementation details (i.e., what is to be extended, improved or refactored and why, but not how this is to be achieved).
 - c. The plan should contain a test plan for the parts of the program to be affected by the proposed extensions, improvements or refactorings. The test plan should describe the class(es) to be tested, the proposed test process including any required test scaffolding, and the general types of tests that the team proposes to execute. You should explain which parts of your test plan will use JUnit.

Submission

The assignment is to be submitted via the web assignment submission page located at:

<http://submit.itee.uq.edu.au/>

Only one member of each assignment team should submit. Multiple submissions are accepted but only the final one is kept. The submission can be either:

1. a single file containing a composite document incorporating all parts of the assignment. The format of the document should be PDF, single spaced and should use standard 11 point fonts with at least 2 cm margins on all sides and it must be possible to copy text from your PDF document, **or**
2. an archive file (zip or gzip) containing a collection of HTML web pages representing an online document. Your web pages should not contain any unnecessarily fancy features such as applets. The root web page must be named "index.html". Note: since your web pages are to be copied, make sure that all your internal links are relative, not absolute.

Your submission should clearly identify the team and its members (including the tutorial class day and time). In addition, a paper copy of the CSSE2003 Assignment 2 coversheet must be submitted in the assignment box for CSSE2003 on level 1 of G.P. South by the deadline. The coversheet should allocate the percentage contribution for each team member (such that the sum of the percentages equals 100 and satisfies the limits specified in the course profile). All team members should sign the cover sheet to indicate their agreement that the percentages represent an equitable division consistent with their individual contributions. Any controversy over the allocation of contribution percentages should be referred to the team's tutor and then, as a last resort, to the course co-ordinator.

To submit a late assignment, see the course co-ordinator. If the team's performance has been adversely affected by exceptional circumstances, the team may apply to the course co-ordinator for special consideration. Suitable documentary evidence (such as a doctor's certificate) should be supplied where appropriate.

Timetable:

A suggested timetable is:

- Week 5-6: If not already done, install source code for selected program in group account under version control system (Subversion). Develop Ant build scripts as required. Commence team learning journal. Start investigating selected portions of source code and developing design documentation.
- Week 6-7: Continue investigating selected portions of source code and developing design documentation.
- Week 7-8: Prepare draft assignment report including the plan for future investigation.
- Week 9: Complete and submit assignment.

Evaluation:

This assignment is worth 15% of the final grade - the assessment criteria is provided on the assignment cover sheet.

Background Reading:

1. Andy Hunt and Dave Thomas. Software Archaeology, *IEEE Software*, 19(2):20-22, March/April 2002. (Available on-line via the UQ library subscription to the IEEE Xplore database).