Informatics 111 / CSE 121: Software Tools and Methods
Fall Quarter 2007

Assignment 3

Due Wednesday, December 5th, 2007, at 11:50pm

Part 1. Effort Estimation (25 points)

This problem is similar to Part 1 from the previous two assignments.

1.1 A priori estimate. (3 points)
When you start working on this assignment, write down an estimate of how long it will take you to do each task in parts 1 and 2. You are not required to use the table from Assignment 2 to generate a more accurate estimate. However, you may use that technique if you feel it will provide a more accurate estimate.

1.2 Logging. (10 points)
As you are working on the assignment, record what you are doing and how long you spent, in the same manner as Assignments 1 and 2. For this question, quantity of data, i.e. number of entries, is important. As a rule of thumb, you should add a log entry every time you switch tasks or at least one entry per hour. If you do two or three things in half an hour, you must have a log entry for each of them. You do not need to include time for logging, but should include the time spent answering the other sections of this part.

Keep a history of your logs. Either

- Place your log under CVS and submit a screen shot of the "CVS Resource History" view in the CVS Perspective of Eclipse, jus as in Assignment 2; or
- Put a two digit numeric suffix at the end of your log file name (e.g. "Log-01.doc"), and every time you save the log, use Save As and increment the suffix. Submit a screen shot of Windows Explorer showing all the versions, including the Date Modified field.

For either screen print, arrange the windows so that what you include in your report is completely legible and large enough to read easily.

1.3 Re-estimation. (3 points)
Take the total *a priori* estimate from 1.1 and divide the number in half. When you have worked this many hours on the assignment, update your estimates. Your new estimates should take into the account the amount of time that tasks have taken so far. Also, make a note in your log indicating when you do this re-estimate.

1.4 Analysis and Error Calculation. (4 points)
Tally up the time spent in each of the tasks and on the entire assignment. Calculate the error for

each task with respect to the Initial Estimate and the Revised Estimate. You should use Excel formulas to perform math operations. Also, produce a bar graph of the time spent per each task. Here is what a table of summary statistics might look like.

Task	Actual Time	Percentage	Initial Estimate	Error	Revised Estimate	Error
1.1 – 1.3						
1.4						
1.5						
2.1						
2.2						
2.3						
2.4						
2.5						
2.6						
Total		100.0				

Table 1. Analysis and Error Calculation

As always, the error is the amount off (the difference between the actual time and the initial estimate), divided by the initial estimate, and expressed as a percentage.

1.5 Discussion. (5 points)
You have used three different estimation techniques; simple estimation, estimation by category, and re-estimation. Compare and contrast these three techniques. What are the strengths and weaknesses of each? Which method is most effective? Can the techniques be combined? What are the advantages and disadvantages of doing so?

Deliverables

Your deliverable for Part 1 is an Excel spreadsheet named Part1.xls, with five tabs (worksheets).
- An initial estimate of hours that each task in parts 1 and 2 will take. (Tab 1)
- A log of your time spent on the assignment. (Tab 2)
- Revised estimates, after half way through estimated time. Also, include how you calculated the time when you had to do the re-estimation. (Tab 3)
- A table similar to Table 1 with the information requested in 1.4. (Tab 4)
- A bar graph showing time spent per each task. (Tab 5)

The following go in your Report.doc or Report.pdf:
- Screen print showing history of log
- Discussion of estimation method and resulting error

Part 2. UML Diagrams and Design Patterns in Lunar Lander (75 points)

The goal in Part 2 is to create a UML Class Diagram using Packages, UML Sequence Diagrams, and a UML State Diagram for the Lunar Lander system. In addition, we will find and analyze the use of design patterns in the system. You should use the source code you implemented in Assignment 2.
You are **required to use ArgoUML** to create this and other UML Diagrams in this assignment. You **may not** use a general-purpose drawing program such as MS Paint, Word, Visio or any other tool.

2.1 Create a new baseline in CVS by using tags (3 points)

This assignment builds on your work from Assignment 2. Before modifying the source code, create a baseline called "Assignment2" by tagging the current version of the files in CVS.

2.2 Add Interfaces and Packages to your UML Class Diagram for Lunar Lander (20 points)

In this step you will add the interfaces and packages to your UML Class Diagram in Assignment 2. Represent the two interfaces and their associations with other classes in the Lunar Lander source code. Create 4 packages to organize the Lunar Lander source code. They could be interfaces, graphs, spacecraft, and process. If you have another package organization in mind, you can use it. You should assign each class and interface to one package.

Refactor the Lunar Lander source code as necessary to apply the package organization proposed in your Class diagram.

2.3 Create UML Sequence Diagrams for Lunar Lander (20 points)

In this step you will create 4 sequence diagrams for 4 of the use cases that you identified in Assignment 2. Make sure that you implement the main scenario and branches of each use case represented in the extended format. If you need to change the steps in the extended format to better represent the Sequence Diagram, you should make the necessary changes. Include in your report the extended format of your use cases and discuss what changes you made on them.

2.4 Create a UML State Diagram for Lunar Lander (10 points)

In this step you will create a State Diagram for the Space Craft class. You should include guards, events/actions, and activities when needed.

2.5 Find a Design Pattern in Lunar Lander (15 points)

The Lunar Lander source code implements the Observer Design Pattern using Java classes. Identify and describe the problem the pattern is solving, and explain how the pattern is solving the problem. Make sure that you will mention the classes that interact with this pattern and the role each involved class plays in the implementation of the Observer Pattern.

2.6 Execute the JUnit Test Suite (4 points)

Execute the JUnit Test Suite you created in Assignment 2 and report the result. You could have errors due to the package refactorings. If so, make the necessary changes to solve the errors.

2.7 Create a new baseline in CVS by using tags and CVS Log (3 points)
After finishing with all the changes to the Lunar Lander source code, create a baseline called "Assignment3" by tagging the current versions of the files in CVS.
Take a screen print of the CVS Perspective in Eclipse, showing that your object's files have been checked in and committed, and also showing the two versions you created in task 2.1 and in this task. The most important files for this purpose are the source code files for your Lunar Lander system.

2.8 Report and Documentation
Use the same Report file (Report.doc or Report.pdf) as in Part 1.
For Part 2, turn in the following:
• 2.1: No documentation required.
• 2.2: Export or include a screen shot of your UML Class Diagram into your Report. Include a description of each package you created and a list of classes you put in each package.
• 2.3: Export or include a screen shot of your four UML Sequence Diagrams into your Report. Include a discussion of any issue you had when you created these diagrams. In addition, include the extended format of each use case you are creating Sequence Diagram to. Discuss the changes you did, if any, to the extended format to better represent the Sequence Diagrams.
• 2.4: Export or include a screen shot of your UML State Diagram into your Report. Include a discussion of any issue you had when you created this diagram.
• 2.5: Include your discussion about the implementation of the Observer Design Pattern in the Lunar Lander source code including the details requested in the task.
• 2.6: In the Report, discuss the results of running the Test Suite. Include a discussion of any test case that had errors and explain how you solved the problem with the test case.
• 2.2 and 2.6: After finishing with tasks 2.2 and 2.6, go to the Eclipse workspace and zip the project folder. Rename the zip LunarLanderAssignment3.zip.
• 2.7: In the Report, include a screen shot from the CVS Perspective. Make sure this is readable and includes the information of the two versions you created (in 2.1 and in this task).

Handing In Your Assignment
Your assignment must be submitted electronically to checkmate.ics.uci.edu. Submit three files.

1. Report.doc or Report.pdf, containing deliverables from Part 1 and Part 2.
2. Part1.xls, containing information from Part 1.
3. LunarLanderAssignment3.zip, containing source code and test cases for your LunarLander system.