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

Assignment 2

Due Thursday, November 15th, 2007, at 11:50pm

Part 1. Effort Estimation (20 points)

This problem is the similar to Part 1 from Assignment 1. You can use the same categories again for this assignment if that seems appropriate, or add new ones.

1.1 A priori estimate. (4 points)
When you start working on this assignment, write down an estimate of how long it will take you to complete each part of this assignment. Break down this estimate according to both the parts of the assignment and the categories of tasks.

The table for breaking down the estimate might look like the following. You should change the column categories to match what you are using.

Task	Project Management	Planning	Programming	Research	Row Sum
2.1					
2.2					
...					
2.5					
Column Sum					

The sum of the rows must equal the sum of the columns, which is the same as the overall estimate that was generated. An Excel spreadsheet is available in Assignment2.zip to help with this (add or remove columns in the spreadsheet if necessary).

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 Assignment 1. 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 two hours. 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.

Place your log under CVS version control. (It should be in the same project as LunarLander in Part 2.) Whenever you update your log, you must check it in to CVS. You will be submitting the revision history along with the log in your written report. This revision history can be submitted as a screen shot of the history view in the CVS Perspective of Eclipse.

1.3 Summary and Error Calculation. (3 points)
For each of the entries in the time log, allocate the time spent to a cell in the table. Calculate the error for each cell, as in Assignment 1 part 1.4. Calculate summary statistics for the rows and columns. The supplied spreadsheet makes this fairly easy.

1.4 Discussion. (3 points)
Did you benefit from the experience you gained in doing an effort estimate for Assignment 2? Did you learn anything new this time? Did you do the overall estimate first and break it down for the table? Or did you generate estimates for the table cells and add them together for the total estimate? Does breaking down the estimate make the job easier or harder?

Deliverables

Many of the deliverables for Part 1 and Part 2 will go in a "Report," which can be named Report.doc or Report.pdf.

- A log of your time spent on the assignment, in your Report.
- CVS revision history of the log (screen print), in your Report. Make sure the screen print is readable and does not cut off important information.
- Your answers to the discussion part (1.4) in your Report.
- A spreadsheet called Part1.xls showing effort estimate, actual time spent, and the error. Include summary statistics for the row and column totals.

Part 2. UML Diagrams and New Version for Lunar Lander (80 points)

The goal in Part 2 is to create the UML Diagrams for the Lunar Lander system and implement a new feature in the game. You should use the refactored source code you implemented in Assignment 1.

2.1. Set up a Project in Eclipse under CVS
Set up an Eclipse project that will contain all code, documents, JUnit tests, spreadsheets, and diagrams for Assignment 2. Put this project under CVS revision control.

2.2. Create a UML Use Case Diagram (15 points)
In this step you will analyze the source code of Lunar Lander that you already know from the previous Assignment. You should identify the primary actors of the system and the Use Cases that represent the behavior of the system. Draw a Use Case diagram using ArgoUML that shows the relationship among Use Cases and Actors. You should include at least 2 actors and 4 Use Cases. Make sure that you have at least one "extend" or one "include" relationship between two Use Cases.

2.3. Document each Use Case in the extended format (20 points)
Describe the details of each Use Case identified in part 2.2 using the Use Case extended format provided here. A word template is available in Assignment2.zip to help with this.

USE CASE #	< the name is the goal as a short active verb phrase>
Goal in Context	< a longer statement of the goal in context if needed >
Preconditions	<what we expect is already the state of the world>
Success End Condition	<the state of the world upon successful completion>
Failed End Condition	<the state of the world if goal abandoned>

Primary, secondary Actors	<a role name or description for the primary actor> <other systems relied upon to accomplish use case>	
Trigger	<the action upon the system that starts the use case>	
DESCRIPTION	**Step**	**Action**
	1	<put here the steps of the scenario from trigger to goal delivery,and any cleanup after>
	2	<...>
	3	
EXTENSIONS	**Step**	**Branching Action**
	1a	<condition causing branching> : <action or name of sub-use case>
SUB-VARIATIONS		**Branching Action**
	1	<list of variation s>

You should fill out all the fields in the template. In case you do not have Extensions or Sub-variations, you can leave these sections blank.

2.4. Create a UML Class Diagram for the Lunar Lander System (15 points)
In this step you will design the Lunar Lander system by writing a UML Class Diagram. Your design should be based in the current version of the Lunar Lander code. You are required to use ArgoUML to draw your diagram and not a general-purpose drawing program such as MS Paint, Word or Visio. You should include attributes, types of attributes, and multiplicity per each class.

2.5. Implement in Java 1.5 the new version of Lunar Lander System (10 points)
The current version of the Lunar Lander game does not keep track of the score. You should add this new feature to the game. This feature will include the following: show the actual score in the screen (the initial score should be 0); if the space craft lands on the platform, the user will receive 100 points; if the user presses a key to continue the game, the score should not be reset to 0. The score will be reset only if the user closes the window. Modify you UML Class Diagram to support this new feature.

2.6. Write a JUnit Test Case for the new feature in Lunar Lander System (10 points)
The new feature includes increasing the score after the spacecraft lands. You should add a new method in the GUIEnvironment JUnit Test Case that executes the fragment of code that increases the score and asserts that the score was increased. To create these JUnit Test Cases you are allowed to modify the source code and apply any necessary refactoring.

2.7. Create a JUnit Test Suite and Execute it (5 points)
Create a Test Suite that contains the Test Cases you have so far for the Lunar Lander. Run your Test Suite and report the results. If you find errors, you should solve them.

2.8. CVS Log (5 points)
Take a screen print of the history view in the CVS Perspective in Eclipse, showing that your object's files have been checked in and checked out. The most important files for this purpose are the source code files for your new Lunar Lander system.

2.9. 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 Use Case Diagram into your Report. Include a description of each primary actor you identified in the system.
- 2.3: Include in your report all the Use Cases in extended format that you detailed.
- 2.4: Export or include a screen shot of your UML Class Diagram into your Report. Include a discussion of any issue you had when you created this diagram.
- 2.5: In the report, discuss your implementation and what you needed to change in you UML Class Diagram to support the new feature.
- 2.6: In the report, discuss the changes you did to the source code to write the test case requested.
- 2.7: 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.
- 2.5, 2.6, 2.7: After finishing from section 2.5 to 2.7, go to the Eclipse workspace and zip the project folder. Rename the zip LunarLanderAssignment2.zip.
- 2.8: In the Report, include a screen shot from the CVS history view. Make sure this is readable and that no part of the view is cut off.

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. LunarLanderAssignment2.zip, containing source code for your LunarLander system.