Assessed Coursework 4:
Assessment of Coursework 3 Report
(Group Work)
February 11, 2011

Group Number Writing Report:
Group URL:
Names of Team Members:

Group Number being Assessed:
Group URL:
Names of Team Members:

Instructions: Each team is to assess the work and product delivered by another team in the form of a report. Each team writes their report by answering the list of questions provided. Consider how the other team's work compares to your own. The report is typed, includes the original questions, and is no longer than 12 pages. Both a PDF and a LaTeX source file for this document (the instructions) are provided on the module web page to help you get started.

The team delivering a late product will lose points according to how many days after the deadline the requirements are fulfilled. Those lost points will then be awarded to the recipient team. *Keep in mind that the team being assessed will receive a copy of the report you write.*

Assignment 4 Assessment Report Deadlines: Electronic and Printed

1. **Friday 4 March** by midnight: online submission of assessment report
2. **Monday 7 March** by 14:00: paper/printed submission of assessment report in the coursework submission drop-box or to the Departmental Student Secretary in room 206 of Faraday Building

1 Evaluation Report

The full evaluation is based on design, implementation, implementation documentation, testing, and group work.

1.1 Design

1. How many classes has the group identified? How does this number compare with the number of classes your group identified?
2. Are class names nouns? Do the chosen class names make sense? Are they intuitive?
3. Does each class have an author?
4. For each group member, how many class cards did they write in the design report? For each member of the group: list the classes they are responsible for designing. Use the `Author:` field of the class cards to identify this.
5. For each class, have the parent and child classes been identified?
6. Has at least one responsibility been identified for each class? If not, for how many classes have responsibilities been identified?
7. Has at least one collaboration been identified for each class? If not, for how many classes have collaborations been identified?
8. Have at least two complete protocols (return type, method name, and input parameters) been identified for each class? If not, for how many classes have protocols been identified?
9. Are method names verbs? Do the chosen method names make sense? Are they intuitive?
10. Have unit tests been identified and described for each class? If not, for how many classes have unit tests been provided?
11. Have at least three class hierarchies been identified and described? How do the hierarchies compare to those that your own team designed?
12. Do classes in the hierarchies appear to reflect the *is-kind-of* relationship? If not, which ones don't seem to?
13. Have at least two-three sub-systems been identified and described? How do the sub-

systems compare to those that your own team designed?

14. Do classes in the subsystems appear to reflect the *is-part-of* relationships? If not, which ones don't seem to?

15. Has a user data file format description been provided? How does it compare with that of your own group?

16. Is the software design complete? Does it make sense? Have decisions been justified? How does it compare with that of your own group?

17. On a scale of 1-7 (1 = very poor, 2 = poor, 3 = less than satisfactory, 4 = average, 5 = good, 6 = excellent, 7 = very excellent, outstanding) how would you rate how well the application was designed? And why?

1.2 Implementation

1. Has the group provided screen capture movies that demonstrate the features of their software?

2. Does the source code compile and provide a running application?

3. For each group member, how many classes did they implement? Provide a list of names and numbers. Use the `author` field of each classes to identify this.

4. For each member of the group, list the classes that they are the author of.

5. Do the class card authors match up with the corresponding Java class authors?

6. How closely does the source code conform to the coding conventions? [2]

7. For each class, list the methods that violate one of *Bob's Concise Coding Conventions (C^3)* [2] along with the rule that was broken and the author of the class. The most commonly neglected rule is #10-use symbolic constants rather than raw numbers when writing source code. For example:

   (1) Class Name, (2) Method Name, (3) Conventions Violated

   1. ClassName::MethodName() –1,5,10.

   2. etc.

8. Does the program support integer and floating point input data?

9. Does the program support a table view?

10. Does the digital calendar support three types of visualizations? Which types of visualizations?

11. How does a user create a visualization? How does the other group's visualization generation compare to your group's? Is it more intuitive?

12. Do the visualizations feature the following?

    - A title,
    - Axis labels,
    - Axis scales,
    - Min and max axis values,
    - A color legend,
    - The time and date on which visualization was created,
    - The author of the visualization
    - A 1-2 sentence descriptive caption of the visualization
    - Resizing of the visualization
    - Saving the visualization to a GIF or PNG file

13. Can the user edit or delete an existing visualization?

14. Does the application support two types of users? Which types of users?

15. Do the functions specified in the group report actually work as advertised?

16. Are there any important features missing with respect to the assignment 3 specification?

17. How well does the design match up with the implementation?

18. How does the software rate in terms of usability? Is it intuitive?

19. On a scale of 1-7 (1 = very poor, 2 = poor, 3 = less than satisfactory, 4 = average, 5 = good, 6 = excellent, 7 = very excellent, outstanding) how would you rate how well the application was written? And why?

1.3 Testing

1. Have unit tests been defined for all of the classes? If not, for how many classes have unit tests been defined?

2. Try running the unit test for each class. Do they actually run and work? List each class that does not have unit tests defined for it or is not working properly. Note: It's fine if unit tests were written that test more than one class at a time. They just have to be documented (noted in the doxygen comments).

3. Are the visualizations actually correct? Is there a correct correspondence between the data and the visualization attributes?

4. Does the program visualize the data provided on the module web page correctly?

5. What happens when you try out one of your own CSV files exported from a spreadsheet? Does the application handle it properly?

6. When does the application crash? Which features cause it to crash?

7. Can the application handle random input?

8. What happens when the input file contains a very large value, e.g., 10,000, 100,000, 1,000,000?

9. What happens when the input file contains a negative number?

10. What happens when the input file contains a decimal number, e.g., 1.10?

11. What happens when the CSV file is very large? What happens if a row in the CSV file contains 100 entries? What happens if a column in the CSV file contains 1000 data samples? Does it crash if the input data file is too large?

12. How does the application handle empty cells, rows, and columns in the input CSV file?

13. How does the program respond when the user or input data file is missing?

14. How does the application handle corrupted user-data or input CSV files? For example, what if you replace the normal user-data file with a JPEG file? What happens if you edit the user-data file with a text editor and give it an unexpected format?

15. Do you think the group did a good job of testing their application overall? How does it compare with your group's testing?

16. On a scale of 1-7 (1 = very poor, 2 = poor, 3 = less than satisfactory, 4 = average, 5 = good, 6 = excellent, 7 = very excellent, outstanding) how would you rate how well the application was tested? And why?

1.4 Implementation Documentation

1. How complete is the doxygen output? Are both the short and detailed class descriptions there as described in *Bob's Concise Introduction to Doxygen* [1]?

2. Are both the class hierarchy and collaboration diagrams there?

3. Are all methods described?

4. Are all input parameters to methods documented?

5. How about return parameters?

6. List all of the methods that have undocumented input and return parameters and the corresponding class to which they belong.

7. Has the group output their source code as part of their doxygen output?

8. How does the other group's doxygen documentation compare to yours? Is it better? Is it more (or less) complete?

9. On a scale of 1-7 (1 = very poor, 2 = poor, 3 = less than satisfactory, 4 = average, 5 = good, 6 = excellent, 7 = very excellent, outstanding) how would you rate the quality of the doxygen output? And why?

1.5 Group Work

1. Have all the required files and documents been provided, e.g., design document, group report, minutes, source code, class files, doxygen web pages, and demo movies?
2. Have all the required files and documents been provided on time? If not, how late was the group in delivering the product?
3. Have all of the files been provided in the correct format, e.g., .PDF, .mpeg, etc., as described in Assignment 3?
4. Do all of the folders and file names conform to the naming conventions described in Assignment 3? List the files (and folders) that do not adhere to the guidelines (if any).
5. Did each group manager contribute to the group report? Does the group report answer the questions from the lecture on group work and team roles?
6. Did each group member contribute to the project? List the group members that contributed and those that did not. Evidence of group member contributions can be found in four ways: (1) by looking at the author of class cards in the Design Report (2) by looking at the author of Java source code (.java) files, (3) by reading the minutes of meeting protocol at who attended the meetings and (4) by reading the Group Report.
7. Did the group report any problems? Did they also report how they resolved those problems?
8. How well do you think the group worked together? Did the active members of the group make up for the inactive members? How does their group compare to yours?
9. Do you think the group met often enough? Did they meet more (or less) often than your group?
10. On a scale of 1-7 (1 = very poor, 2 = poor, 3 = less than satisfactory, 4 = average, 5 = good, 6 = excellent, 7 = very excellent, outstanding) how would you rate how well the group worked together? And why? In your answer, take into account how well the group coped with non-contributing members.

1.6 Overall Judgement

1. On a scale of 1-7 (1 = very poor, 2 = poor, 3 = less than satisfactory, 4 = average, 5 = good, 6 = excellent, 7 = very excellent, outstanding) how would you rate the quality of the work handed over to you? Why? How does it compare to your group's product?
2. Overall, how confident is your team in its ability to take the system you've been given and extend it with the functionality in the next assignment? Why?

References

[1] R. S. Laramee. Bob's Concise Introduction to Doxygen. Technical report, The Visual and Interactive Computing Group, Computer Science Department, Swansea University, Wales, UK, 2007. (available online).

[2] R.S. Laramee. Bob's Concise Coding Conventions (C^3). *Advances in Computer Science and Engineering (ACSE)*, 4(1):23–26, 2010. (available online).