cs3307a – Object oriented analysis and design

Phase II Dashboard Project

Document 2/2 – Deliverables & Evaluation

September 2016

IMPORTANT!

- ***Submission*: All deliverables from this project are to be made into the OWL system (unless otherwise stated anytime during the term).**
- ***Deadlines*: Please see the spreadsheet "SCHEDULE OVERALL" for the deadlines and as specified on OWL.**

Submission of Early Questions:

- **DELIVERABLE 1**: Each group to submit at least:
  - three *requirements* questions (e.g., about: system functions, information display format, user interaction, etc.).
  - two *context* questions (e.g., about: type of hardware used by the users, operating system, databases, etc.).

  Please note that these questions "**do not**" require examination of the Phase I system code and documentation. Groups will have ample opportunity to ask more questions throughout the term when the project is underway.

  Each group is to use these questions to ask the customer *in an in-class session* on Monday 3rd October, 2016.

Understanding, Assessing, and Improving Phase 1 system

- **DELIVERABLE 2: Report: Assessment of the Phase 1 Project**

The provided system's code-base has been tested in a limited way. Thus, prior to enhancing it with new features, it is imperative that the group:
(i) understands and assesses:

- how the existing system works
- the system's design, code and documentation,

(ii) documents specific improvements it has decided to make to the Phase I system, and
(iii) makes these improvements.
See Phase II Goals Item (1), in the customer's presentation slides, in reference to the above tasks. (Special note: In the customer's slides it says: "(c) Assessment of the Documentation (ApplePicker)" – in fact, you need to assess the given documentation, not that of "ApplePicker")

As part of this Deliverable, the group must **create test-cases** in order to verify whether or not the current system meets its specifications – user interface aspects, backend aspects, other requirements, etc. In other words, for each requirement for the current system, create tests to check whether or not the system satisfies that requirement. If a valid test fails then this implies that there is bug in the system which should be fixed. This project uses the Qt framework. Qt has unit testing features which we encourage groups to explore.

Below, we give some starter Youtube and web resources; there is lots more on the web so please explore these for supporting your project.

Documentation for Qt test: http://doc.qt.io/qt-5/qttest-index.html

Learn what the current system does by reading code and UML/design specifications.
Run the software system and play around with it, etc.
Make sense of Qt and how it looks and feels when developing.
Understand the strengths and weaknesses of the software's current design.

Youtube tutorial on writing test cases

- How to write effective test cases quickly:
  https://www.youtube.com/watch?v=-VvQKEvsPpE

- How to write a Test Case:
  https://www.youtube.com/watch?v=BBmA5Qp6Ghk

How To Write Test Cases

http://www.guru99.com/test-case.html
http://www.softwaretestinghelp.com/how-to-write-effective-test-cases-test-cases-procedures-and-definitions/

Example test case format

Fields in a test case: **(adapted from:** http://www.guru99.com/test-case.html **)**

- **Test case id:**
- **Unit to test:** Which software component is to be verified?
- **Requirements:** Which requirements (give Ids) are the components being verified against.
- **Assumptions:**
- **Test data:** Variables and their values
- **Steps to be executed:** Here is a list of points that make up the scenario being test.
- **Expected result:**
- **Actual result:**
- **Pass/Fail:**
- **Comments:**

Your tasks to satisfy Delivervable 2 Report:

(a) Give an assessment of the Phase I system (see above), and make the decided upon improvements.
(b) Create test cases, test the current system, and log the test results.
(c) For failed test cases, fix the defects and show that the test cases have passed. So, show the failed code fragment, failed test result, modified code fragment, and passed test result.
(d) Draw a matrix showing requirements of the current system X test cases involved in its

verification.

System demos at Stage 1 and Stage 2

From the overall schedule, you will notice that there are two demos of the system to be delivered, at Stage 1 and Stage 2 of the project. They will be examined by the instructing team and/or the customer. Please see the "**Evaluation**" section for assessment of the demos.

DELIVERABLE 3: Stage 1 system:

i) *Deadline*: Stage 1 system **must** be submitted by the deadline (see project schedule). Late submission will not be accepted.

ii) *Executable*: Preliminary version (Demo 1) of a substantial part of the fully enhanced system. This includes the improvements made in DELIVERABLE 2.

iii) *Requirements Implemented*: Describe the *enhanced* requirements that have been implemented in the Stage 1 demo system. Please refer to Phase II Goals in the customer's presentation slides (Mandatory Requirements 2-9 and Stretch Requirements 1-5). Please indicate which implemented requirements are *mandatory*, and which are *stretch* (optional).

At its minimum, a requirement should have the following:

- It's ID
- A description of what the requirement is.
- It's origin (where did this requirement come from (give specific reference or name of author, etc.)?)
- Supporting example – as needed: screenshot, diagram, etc. to explain this requirement.

As a starter, please see the following, but there is tons on the web on requirements: https://hubtechinsider.wordpress.com/2011/07/28/how-do-you-write-software-requirements-what-are-software-requirements-what-is-a-software-requirement/

iv) *Test cases*:

- Create test cases, test the Stage 1 system, and log the test results.
- If there are test cases that have failed but not resolved by the time of submission of Stage 1 then these should be identified explicitly.
- Draw a matrix showing requirements of the Stage 1 system X test cases involved in its verification.

v) *System design as at Stage 1* (see Fowler):

a) Use case diagram
b) Original Class diagram
c) Enhanced Class diagram

NOTE: Each diagram must be accompanied by an *explanation*:

- **For the Use case diagram**: If this is different for the enhanced system from that of the original system, then explain the diagram for the original system and also explain the changes needed for the enhanced system. Give rationale for your decisions.

- **For the Class diagram**: Similarly, explain the enhancements you needed to make to the original design. Include in your explanation: (1) *what* the diagram conveys and (2) the *rationale* for the design or enhancements (e.g., components that needed to be changed and why; the choices made for the classes; modularity; the correspondence between the use case and customer's requirements; etc.).

iv) *Development plans*:
You are free to use any appropriate project management tool to show development plans.
   a) Timeline showing:
      1) Achievements to date (Stage 1) with dates. Example achievements (with appropriate dates):
         - Phase I system documentation understood.
         - Phase I system's operational aspects understood.
         - Improvement to Phase I system identified and logged.
         - Improvements to Phase I system completed.
         - Test cases created for the Phase I system and system tested.
         - Demo 1 enhancement requirements determined.
         - Demo 1 enhancement requirements tasks allocated to specific group members.
         - Development infrastructure set up.
         - Design accomplished in iterations: v1, v2, etc.
         - Features implemented (coded and tested): feature1, feature2, etc.
         - Etc.
      2) Concrete Plan for achieving Stage 2 (Demo 2) requirements.
         - Identify major tasks and plot them on the timeline.
      3) Abstract Plan from Stage 2 to the Final submission.
   b) Agent-Task view (Excel spreadsheet), showing:
      1) Task allocation to date (Stage 1): Who has done what
         - Tasks as rows; Agent-names as columns
            - Task description should be self-explanatory (NOT cryptic): e.g., feature description.
      2) Planned task allocation from Stage 1 to Stage 2.

      NOTE: C++ programming requirement for all, described earlier, must be explicitly shown in the spreadsheet. This can spread over Stage 1 and Stage 2.

DELIVERABLE 4: Stage 2 system:

i) *Deadline*: Stage 2 system **must** be submitted by the deadline (see project schedule). Late submission will not be accepted.
ii) *Executable*: Advanced version (Demo 2) of the fully enhanced system.

iii) *Requirements Implemented*: A list of the requirements that have been implemented in the Stage 2 demo system. Please refer to Phase II Goals in the customer's presentation slides (Mandatory Requirements 2-9 and Stretch Requirements 1-5). Please indicate which implemented requirements are *mandatory*, and which are *stretch* (optional). (Format as in DELIVERABLE 3)

iv) *Test cases*:

- Create test cases, test the Stage 2 system, and log the test results.
- If there are test cases that have failed but not resolved by the time of submission of Stage 2 then these should be identified explicitly.
- Draw a matrix showing requirements of the Stage 2 system X test cases involved in its verification.

v) *System design as at Stage 2*: (see Fowler)

a) Use case diagram
b) Class diagram
c) Sequence diagram(s) for specific scenarios.
d) Package diagram

(See Fowler's book on UML for diagrams and their semantics.)

NOTE: Each diagram must be accompanied by an explanation: (1) *what* the diagram conveys and (2) the *rationale* for the design (e.g., the choices made for the classes; modularity; the correspondence between the use case and customer's requirements; how the functions interconnect to satisfy the scenario; criteria used to create packages; etc.).

vi) *Design patterns*:
Give graphical representation (together with its explanation) of any design patterns that have been used in the design of the system; if not, justify giving technical reasons why the design of your program could not be implemented with design patterns.

vii) *Implementation* in C++

a) Does the code satisfy the design? Explanation and justification of the implemented parts of the system with specific references to the code *and* the design of the system. (Maximum 1 page)
b) Code files.

vi) *Development plans*:

a) Timeline showing:
    1) Achievements at various times to date (Stage 1 and Stage 2).
    2) Concrete Plan from Stage 2 to the Final submission.
b) Agent-Task view, showing:
    1) Who has done what to date (Stage 1 and Stage 2)
    2) Planned task allocation to completion.

NOTE: C++ programming requirement for all, described earlier, must be explicitly shown in the view.

Inspections and Object-oriented (OO) Metrics [NOTE: Part of Stage 3]
Once the project has commenced, a further requirement will be given to the project groups to inspect the design and code of the system using an inspection instrument.

Lessons learnt and retrospective analysis [NOTE: Part of Stage 3]
For the final submission of the project, you are required to enumerate lessons learnt in, and retrospective analysis of, the project. To be addressed include:

1) *Describe your top three lessons learnt (**Technical issues only**)*: What went wrong (or what difficulties were faced) in understanding the old code and documentation, design of new features and blending these with the old features, coding and/or testing of the system, etc., that, with hind sight, you would do differently and in which way.
2) *Retrospective analysis (**Human issues only**)*: Analysis of your team organisation, management, and project performance. What would you have done differently?
3) *Value*: Please describe, in bullet points, the value of this project for your learning.

DELIVERABLE 5: Stage 3 requirements: **Final system delivery**
The specifications for the final system are:

- *Title page*: Course name and number; year and semester; project title; group name; member names; instructor name.
- *Table of contents*
- *Requirements Implemented*: A list of the requirements that have been implemented in Stage 3 (final) system. Please indicate which requirements are *mandatory* and which are *stretch*.
- *Test cases*:
  - Create test cases, test the Stage 3 system, and log the test results.
  - If there are test cases that have failed but not resolved by the time of submission of Stage 3 then these should be identified explicitly.
  - Draw a matrix showing requirements of the Stage 3 system X test cases involved in its verification.
- *System design* as at Final delivery
  - Use case diagram
  - Class diagram
  - Sequence diagram(s) for specific scenarios.
  - Package diagram

    NOTE: Each diagram must be accompanied by an explanation: (1) *what* the diagram conveys and (2) the *rationale* for the design (e.g., the choices made for the classes; modularity; the correspondence between the use case and customer's requirements; how the functions interconnect to satisfy the scenario; criteria used to create packages; etc.).
- *Design patterns*:

  Give graphical representation (together with its explanation) of any design patterns used in the implementation; if not, justify giving technical reasons why the design of your program could not be implemented with design patterns.

- *Code and design inspection* data: **specifications to follow**.
- *Implementation* in C++
  a) Does the code satisfy the design? Explanation and justification of the implemented parts of the system with specific references to the code *and* the design of the system. (Maximum 1 page)
  b) Code files.
- *Development plans* from project start to completion: Timeline and Agent-task view. For specifications, see Stage 2 requirements.
- *Lessons learnt* and retrospective analysis (see above).

Evaluation

The following constitute the various components for the evaluation of the project:

Deliverable	Description	Max % marks
1	Submission of Early Questions	0; -3 for not submitting
2	Understanding, Assessing, and Improving Phase 1 system	10
3	Stage 1 system	10
4	Stage 2 system	15
5	Final system	65
	Final system mark breakdown	
	Minimum reqts. implemented*	20
	Stretch reqts. implemented*	10
	Test cases for the entire system*	4
	System design documentation	10
	Use of appropriate Design patterns	7
	Code and design inspection	5
	Acceptance tests (conducted by staff)	0 for ok or minus 10
	Development plans	3
	C++ coding reqt. (individuals)	0 for ok or minus 5
	Overall project documentation and packaging, including user documentation on how to install and run the system, platform requirements, etc.*	4
	Lessons learnt and retrospective analysis	2

* Note: The system must be operational else these marks will be zero. We **strongly recommend** that the system be developed in "iterations" (and saved as operational versions along with the full set of documents reflecting each version saved) such that if the version to be submitted does not work for some reason then you can fall back to the previous version and submit that along with the corresponding set of documents.

END.