cs3307a – Object oriented analysis and design

Phase II Dashboard Project

Document 1/2 – Overall Description

September 2016

Introduction and Project description

This document describes a system *evolution* class project for this term. This means that the previous version of a system will be given as a starting point for upgrading it – it is not development from scratch. This project is for a *real* customer within the university – The Schulich School of Medicine. So, it is not a throw-away system once developed. The customer should be able to use the system in their day-to-day processes. Thus, they do not expect a “prototype” system. Also, as with all production-version systems, the delivered system must be accompanied by informative documentation so that the customer can maintain the system after delivery.

While the specifics of the system are described in an accompanying document (see Appendix 1 --- Customer's Specifications), here we describe the general idea of the project, for informational purposes.

There is personnel (faculty) data in CSV (comma separated values) files – as spreadsheets. There are several spreadsheets pertaining to: teaching, publications, presentations given, grants obtained, etc. The task in this project is to process the spreadsheets, given user-chosen parameters, and provide summaries of formatted information (e.g., a list of publications for a given faculty member during the specified timeframe; collaborators of a faculty member in research grants; aggregate time spent by a faculty member in various categories of activities for the given period; etc.). Requirements, some mandatory and some “stretch”, ask for tabulated information as well as graphical layouts (e.g., histograms, pie charts, etc.).

As mentioned earlier, this is an evolutionary project, not development from scratch. Thus, among the critical development tasks would be to understand the previous system (its code and design, its documentation, operational behaviour, etc.). Enhancements would need to be designed into the given system and code modified and enhanced to meet the given requirements. Likewise, testing components, subsystems and the overall system would need to make sure that what was previously working is still working after the system is modified – called “regression testing”. Also, the design and code is to be inspected and, schedule permitting, analysed quantitatively using object-oriented metrics.

The resultant system, in the C++ programming language, should be a stand-alone system in that it should not need to interact with any other external system for it to function.

Learning Outcomes

- Experience with the programming language C++.
- Experience with using libraries (e.g., Qt).
- Experience in evolving an existing, real system.
- Experience with collaboration infrastructure (e.g., Github).
- Experience with specialised development roles: manager, developer, builder, inspector, etc.
- Experience with group work.
- Experience in dealing with a real customer.
- An understanding of the object model as applicable to programming, design and analysis phases.
- Experience with identifying and classifying objects from given problem descriptions and making choices considering quality attributes.
- Experience with design pattern assessment, choice, and appropriate use in projects.
- Experience with OO metrics and the possible use of metric tools.
- Experience with design inspections.
- Experience in project documentation.
- Experience in project presentations and demos.

Group Work, Responsibility, and Peer Review

This project is group work. Individuals in a group are required to collaboratively create different parts of the system into an integrated whole.

For any reason whatsoever, if one or more members do not contribute as expected to the group's goal, the rest of the group is still required to satisfy the goals of the project.

Also, at stipulated times in the project, there is mandatory peer review (details to be provided separately). The peer review outcome will be used to moderate an individual's project mark in the group, with the maximum possible mark reflecting the overall project quality. A satisfactory peer review of an individual would imply no reduction in the individual's project mark. Thus, it is really important that everybody contributes to the success of the group to raise the collective group mark.

Communication with the Customer and the Instructing team

Please kindly note the following communication protocols:

- There will be scheduled customer sessions during which the project teams will be able to interact with the customer, for example to better understand the purpose of the system, how the customer will use it, data issues, etc.
- Outside the scheduled sessions, should a group need any information from the customer, then please contact your designated TA.

- Any member of a group can contact the designated TA for that group. It is important to channel the communications to the designated TA (and not to any other TA) so as to provide assistance efficiently and to build upon the group's knowledgebase with the TA. Exception to this protocol will be notified. Any member of a group can contact the instructor.

Group member roles and "champions" in the project

The project involves many different types of tasks (e.g., code understanding and coding, extending and changing existing design, planning, testing, inspecting, documenting, and others.) and thus there are different roles group members need to assume at different times, or concurrently, in the project.

Here, we list *example* roles and associated tasks for consideration in a group. The group can adapt these and create other appropriate roles not listed here.

- *Project manager*: project planning, task allocation, task tracking, progress monitoring, alerting, etc.
- *Quality Controller*: ensuring that requirements are met (e.g., testing someone else's code, inspecting design and code, etc.)
- *Quality Assurer*: planning for quality control (e.g., developing test suite and plans)
- *Developer*: code reader, programmer, tester, etc.
- *Builder*: integrating software components as per build plans.
- *Designer*: designing (part of) the system.
- *Requirements engineer*: eliciting, analysing and prioritising requirements.
- *GUI designer*: creating screen layouts, graphical user interface design, etc.
- *Project documenter*: documenting software artefacts (plans, designs, requirements, etc.)
- *Liaison agent*: communication with the TA.

Besides the roles listed above, it is advantageous to identify "champions" in the group (i.e., people with expertise on specific items in the project) so that they can share this with other members in the group. Examples include:

- C++, UML, etc.
- Design patterns
- OO metrics
- GUI
- Github, Qt, Maven, etc.
- Video and demo preparation
- Etc.

C++ programming requirement

Important: Everybody in the group (without exception) is required to program in C++ and perform testing as per the project plans. The plans should explicitly indicate agent names and features allocated to them for programming and testing among other project tasks.

Project Milestones and Events

The overall schedule shows the project activities and dates when key events are planned to occur. Though every effort will be made by the commissioning and instructing teams not to change major milestones (such as delivery dates of the demos), it is possible that the milestone dates can change for important reasons not known at this time. The impact on the project teams would be assessed in these circumstances and be kept to a minimum where possible. The project teams will be given as much advance notice of such changes as possible. Such changes can occur when in real projects!

Deliverables and Evaluation

Please refer to “Document 2”.

Development infrastructure and Library

Please note that Github and Qt are required.

- **Github**, a repository hosting service with source code management and revision control that facilitates collaboration within a project team.
  - Each group member needs to ensure that s/he has access to Github, but only one designated member needs to ensure that s/he hosts a private repository for the rest of the group.
- **The Qt library** for GUI development.
  - 2016: please use version 5.7 -- https://www.qt.io/download-open-source/

- **Additional information on Qt**
  https://en.wikipedia.org/wiki/Qt_(software)
  http://doc.qt.io/qt-5/qtexamplesandtutorials.html
  http://doc.qt.io/qt-5/gettingstarted.html
  http://www.qt.io/application-development/
  https://wiki.qt.io/Qt_for_Beginners

- **YouTube video: QT C++ GUI Tutorial 1- Installing Qt SDK**
  https://www.youtube.com/watch?v=KyuRksYpjRs

- **Maven**
  http://www.tutorialspoint.com/maven/
  http://tutorials.jenkov.com/maven/maven-tutorial.html
  http://www.mkyong.com/tutorials/maven-tutorials/

  Maven is a project management and comprehension tool. Maven provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.

END.