**cs3307 – Object oriented analysis and design**

[**Dashboard Project**](http://www.cppforschool.com/project/banking-system-project.html)

September 2015

**Introduction and Project description**

This document describes the class project for this term. As you know from informal conversation, this project is for a real customer within the university. While the specifics are described in an accompanying document (see Appendix 1), the general idea for the project is the following.

There is personnel (faculty) data in CSV (comma separated values) files – as spreadsheets. There are four spreadsheets pertaining to: teaching, publications, presentations given, and grants obtained. The task in this project is to process the spreadsheets, given user-chosen parameters, and provide summaries of formatted information (e.g., publications list for a given faculty member during the specified timeframe; collaborators of a faculty member in research grants, etc.). Requirements, some mandatory and some “stretch”, ask for tabulated information as well as graphical layouts (e.g., histograms, pie charts, etc.). The system to be developed, in the C++ programming language, is a stand-alone system in that it does not need to interact with the customer’s other systems that capture faculty data. Also, the design and code is to be inspected and analysed quantitatively using object-oriented metrics.

Finally, the project milestones and events (see Appendix 2) dictate deliverables at specific times in the project and depict sessions with the customer. It is really **important** that this be examined along with this document in order to familiarise with such issues as:

* Submission of early questions
* Submission of demos
* Submission of the final system
* Customer interaction sessions (in class)
* Etc.

**All submissions from this project are to be made in the OWL system.**

**Learning Outcomes**

* Experience with the programming language C++.
* Experience with using libraries (e.g., Qt).
* Experience with specialised roles: manager, developer, builder, inspector, etc.
* Experience with working in a team.
* Experience in dealing with a real customer.
* Experience with collaboration infrastructure (e.g., Github).
* An understanding of the object model during programming, design and analysis phases.
* Experience with how to classify objects from given problem descriptions and how to make choices considering quality attributes.
* Experience with design patterns.
* Experience with OO metrics and the use of metric tools.
* Experience with design inspections.
* Experience in project documentation.

**Group Work, Responsibility, and Peer Review**

This project is group work. Individuals in the group collaboratively create different parts of the system into an integrated whole. All the members of a group are thus expected to contribute earnestly to the success of the group.

The mark given to the individuals in the group is the *group mark*. For any reason, if one or more members do not contribute as expected to the group’s goal, the rest of the group is expected to fill the shortfall; the end mark for individuals is still the group mark.

Besides, should anyone in the group request a peer review, it will trigger peer review if after assessment by the instructor he determines that it is warranted. The peer review will determine 50% of an individual’s project mark in the group concerned. Please note that despite the peer review, an individual in the group can get at most the *group mark*. Thus, it is really important that everybody contributes to the success of the group to raise the collective group mark.

**Discretionary Individual Performance Marks**

Based on observations and evidence, the instructor may give discretionary marks to an individual in a group for his or her noteworthy performance in the project.

**Communication with the Customer and the Instructing team**

Please kindly note the following communication protocols:

* Only a group’s designated liaison and assistant liaison persons are permitted to contact the customer for clarifications during the project’s lifecycle. These liaison persons’ contact details (e.g., email addresses) will be shared with the customer. The customer’s contact details are given with the project description.
* Any member of a group will be able to contact the designated TA for that group. It is important to channel the communications to the designated TA (and not other TAs) so as to build upon the group’s knowledgebase with the TA.
* Anyone can contact the instructor at anytime.

**Development infrastructure and Library**

The project requires the use of:

* Github, a repository hosting service with source code management and revision control that facilitates collaboration with 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, useful for GUI development.

**Group member roles and “champions” in the project**

The project involves many different types of tasks (e.g., coding, designing, planning, testing, inspecting, documenting, and others.) and thus there are different roles (corresponding to such tasks) group members need to assume at different types in the project. Note that in the project a group member can play more than one role, concurrently or otherwise.

Here, we list *example* roles and associated tasks for consideration in a group. The group can 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, measuring design and code quality, etc.)
* *Quality Assurance*: planning for quality control (e.g., developing test suite and plans)
* *Builder*: integrating software components as per build plans.
* *Designer*: designing (part of) the system.
* *Requirements engineer*: eliciting, analysing and prioritising requirements.
* *GUI designer/developer*: creating screen layouts, graphical user interface design, etc.
* *Design and code inspector*: design and code reviewer.
* *Project documenter*: documenting software artefacts (plans, designs, requirements, etc.)

It should be quite obvious that the “developer/tester” role is not listed above. There is a good reason for that; everybody will need to code and test! See later below on this.

Besides above listed roles, for project’s success, it is advantageous to identify “champions” (i.e., people with knowledge and experience about specific items in the project) in the group so that they can share this with other members in the group. Examples of such items include:

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

Yet, there are other organisational roles of importance, such as:

* Liaison person with the customer

Individuals are encouraged to make noteworthy contributions to the group’s success for possible consideration for discretionary marks.

**C++ programming requirement**

*Important*: Everybody in a group (without exception) is required to program in C++ and perform testing. This should be explicitly indicated in the project plans (see Agent-Task spreadsheet for system demos below).

**Project Milestones and Events**

Appendix 2 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 the time of this writing. 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 real customers are involved!

**Submission of Early Questions**:

* Each group to submit at least three requirements questions (to be asked later to the customer in an in-class session with the customer).
* Similarly, each group to submit at least two system questions.

**Please check the project schedule for dates of delivery – it is imminent!**

Delivery to be made on the OWL system.

**System demos at Stage 1 and Stage 2**

From the project schedule in Appendix 2 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 both the customer and the instructing team. Please see the “**Evaluation**” section for assessment of the demos.

There are two important purposes of these demos that should not be overlooked by the project groups:

(1) to assess progress of the emerging system towards completion, and

(2) to enable the customer to rank the systems submitted by different groups.

The top three ranked systems by the customer (separately for Demo 1 and Demo 2) will receive discretionary bonus marks!

Stage 1 requirements:

1. *Deadline*: The demo **must** be submitted by the deadline (see project schedule). Late submission will not be accepted.
2. *Executable*:
   1. Operational system with screen display of at least *one* of the four summary types: Presentations, Teaching, Funding, and Publications, as shown in the customer’s document “Project Initial Requirements”, Appendix B – Dashboard Screens. Plus/minus (+ -) buttons should expand/collapse sections. Date range filters operational.
   2. Operational system showing visualisation (graphs) of at least *one* of the four summary types. For details on the graphs, please see the customer’s document “Project Initial Requirements”, Appendix C -- Visualisations.
3. *System design as at Stage 1* (see Fowler):
   1. Use case diagram
   2. Class 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; etc.).

1. *Development plans*:
   1. Timeline showing:
      1. Achievements at various times to date (Stage 1). Example (with appropriate dates):
         * Requirements understood
         * Development tasks determined and allocated to group members
         * Development infrastructure set up
         * Design accomplished: v1, v2, etc.
         * Features implemented (coded and tested): feature1, feature2, etc.
         * Etc.
      2. Plan for achieving Stage 2 requirements, if not already.
         * Identify major tasks and plot them on the timeline.
      3. Abstract plan from Stage 2 to the final submission.
2. Agent-Task spreadsheet, showing:
   * 1. Who has done what to date (Stage 1)
        + Tasks as rows; Agent-names as columns
          - Task description should be self-explanatory (NOT cryptic): e.g., feature description.
     2. Planned task allocation 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.

Stage 2 requirements:

1. *Deadline*: The demo **must** be submitted by the deadline (see project schedule). Late submission will not be accepted.
2. *Executable*: Operational system showing *at least the minimum* requirements listed in Appendices B (Dashboard Screens) and C (Visualisations) of the customer’s document “Project Initial Requirements”.
3. *System design* ***as at Stage 2*** (see Fowler):
4. Use case diagram
5. Class diagram
6. Sequence diagram for a scenario.
7. 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.).

1. *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.

1. *Implementation* in C++
   1. 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)
   2. Code files.
2. *Development plans*:
3. Timeline showing:
   * 1. Achievements at various times to date (Stage 1 and Stage 2).
     2. Plan from Stage 2 to the final submission.
4. Agent-Task spreadsheet, showing:
5. Who has done what to date (Stage 1 and Stage 2)
6. Planned task allocation to completion.

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

**Inspections and Object-oriented (OO) Metrics**

Once the project has commenced, a further requirement will be given to the project groups:

1. To inspect the design and code of the system using an inspection instrument
2. To measure the quality of the system using OO metrics and to interpret the measures.

Please note that the resultant data and information from these activities should be delivered in the **final submission of the project** (not during Stage 1 and Stage 2).

**Lessons learnt and retrospective analysis**

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. What went wrong in the design, coding and/or testing of the system that, with hind sight, you would do differently and in which way. Describe your top three lessons learnt.
2. Please comment upon whether or not documenting system design before/during system implementation has any merits over documenting after system implementation, justifying your answer.
3. Analysis of your team organisation and project performance.
4. How smoothly the interactions with the customer went.
5. Any improvement recommendations for a project for the next generation of students.
6. Value of this project for your learning.

**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.
* *Table of contents*
* *Minimum requirements*: Evidence shown that the minimum requirements listed in Appendices B (Dashboard Screens) and C (Visualisations) of the customer’s document “Project Initial Requirements” have been met satisfactorily.
  + - Includes test cases used
* *Optional*: Evidence shown that one or more “Stretch goals” listed in Appendix C (Visualisations) of the customer’s document “Project Initial Requirements” have been met satisfactorily.
  + - Includes test cases used
* *System design* as at Final delivery
* Use case diagram
* Class diagram
* Sequence diagram for a scenario.
* 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.
* *Metrics and analysis*: specifications to follow.
* *Development plans* from project start to completion: for specifications, see Stage 2 requirements.
* *Lessons learnt* and retrospective analysis

**Evaluation**

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

* *Early Questions*: Three Requirements questions submission on time then 0 else minus 3% – see project schedule.
* *Early Questions*: Two system questions submission on time then 0 else minus 3% – see project schedule – see project schedule.
* *Demo 1 submission* on time with Stage 1 requirements met then 0 else minus 5%. Discretionary bonus marks based on customer’s ranking.
* *Demo 2 submission* on time with Stage 2 requirements met then 0 else minus 5%. Discretionary bonus marks on customer’s ranking.
* *Customer ranking*. Discretionary bonus marks for the top three teams in priority order.
* *Discretionary Individual Performance* Marks
* *Final system submission* on time then 0, else minus 5% for one day late or minus 10 for two days late, not accepted thereafter.
* **Final system evaluation**:
  + **[40%]** Evidence shown that the *minimum requirements* listed in Appendices B (Dashboard Screens) and C (Visualisations) of the customer’s document “Project Initial Requirements” have been met satisfactorily.
    - Includes test cases used

NOTE: if the minimum requirements are not met then the rest of the marks categories indicated below will be moderated to 75% of its allotment (so, e.g., “System design” will achieve a maximum score of 7.5%, not 10%).

* + Evidence shown that one or more “*Stretch goals*” listed in Appendix C (Visualisations) of the customer’s document “Project Initial Requirements” have been met satisfactorily. Discretionary bonus marks.
    - Includes test cases used
  + **[10%]** *System design* as at Final delivery (see Stage 2 (iii) for details)
  + **[10% ]** Use of appropriate *Design patterns* (see Stage 2 (iv) for details)
  + **[10%]** Code and design *inspection* data
  + **[9%]** *Metrics* and analysis
  + **[10%]** *Acceptance test* results (staff-driven)
  + **[5%]** *Development plans* from project start to completion (see Stage 2 (vi) for details).
  + *C++ coding requirement* for each member of the group satisfied then 0, else minus 5%.
  + **[3%]** *Overall* project documentation and packaging
  + **[3%]** *Lessons learnt* and retrospective analysis

**Appendix 1** – see customer’s documents:

* in the zipped folder “V3\_Schulich\_Western\_CS\_collab”
* Presentation Slides: STAR 2015 Acuity STAR Western CS intro V2

**Appendix 2** – Milestones and events – see document v11\_2015\_PROJ & ASST SCHEDULE

**Appendix 3** – some useful links

**Qt library use**

<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>

There are also some usful videos on YouTube beginning with this one:

<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.