CS4125 SYSTEMS ANALYSIS AND DESIGN

TEAM-BASED PROJECT

Specification V2

Semester 2: 2016-2017

J.J. Collins

8th February 2017 (Week 3)

1. Objectives

1. To apply an Object-Oriented (OO) Analysis and Design (OOAD) method using the Unified Modelling Language (UML) to the development of an enterprise software system.
2. To apply architectural and design patterns where appropriate.
3. Gain insights into team dynamics and project management from the experience.

2. Scenario

You have been commissioned by a client to either develop a new system, or adapt an existing system to incorporate changes in the business model. A list of examples is as follows:

- Traffic flow simulation.
- Games hosting framework
- Warehouse distribution or retail.
- Accounts processing for a new mobile phone company.

The system should have sufficient "business rules" to make the application compute intense, as opposed to being one that primarily post queries and updates to a database from a GUI. One example of a set of business rules would be differing discount policies for various customer categories.

3. Specification

You are required to submit a specification and subsequent implementation, which contains:

1. Front cover with business scenario title, student names and IDs, and module details.
2. Table of contents.
3. A one-page (maximum) narrative description of your development project.
4. A one-page (maximum) discussion on the software lifecycle model adopted to manage the process of software development.

5. Lightweight project plan with allocation of roles clearly stated.
6. **Requirements**
    - Functional:
        - One or more use case diagram(s). Do not use decomposition!
        - For each use case, a semi-structured use case description. See the structure used at the end of chapter A2 in Bennett, McRobb, and Farmer, 4th Edition (2010).
        - For at least one use case, the use case description to be specified using the template distributed in lectures.
    - Non-functional requirements specified as add-ons in the use case description.
    - A short discussion on tactics that one might consider adopting to support specific quality attributes.
    - A brief selection of:
        - Screen shots of GUI prototypes - 2 should be sufficient, TOKEN ACKNOWLEDGEMENT of the role of prototyping in requirements elicitation.
7. A two-page (maximum) discussion of **system architecture** to include at least one package diagram, and detail the architectural decisions taken at this point in the project. Briefly discuss the UML workbench used for section (12) of this project.
8. Lightweight **analysis sketches**: use an object-oriented methodology with UML. Include the following descriptions / diagrams where appropriate, or show why not appropriate:
    - List of candidate objects derived using Data Driven Design (DDD).
    - Sketch of the analysis class diagram, with significant methods and attributes identified. The sketch should demonstrate inheritance, aggregation and composition, associations, dependencies, visibility, etc. There should be no duplication of attributes or methods in the diagram. The diagram should contain at least two class interfaces.
    - A sketch of a communication diagram or sequence diagram.
    - Entity Relationship diagram with cardinality - Not covered in CS4125!
    - **TO BE COMPLETED BEFORE STARTING SECTION 10.**
9. A listing in tabular format of classes in each package, their authors, and lines of code. Also include total lines of code developed (a) for the application as a whole, and (b) per team member.
10. **Code**:
    - Proof of concept prototype.
    - Implement those classes necessary to support a subset of the use cases.

- Implementation of Model View Controller architectural pattern.
- Application of one or more design patterns.
- Include coding fragments in this section of the report to illustrate implementation of MVC, design patterns, and other "interesting" aspects of the implementation. Also include screen shots of the GUI/UI.

11. **Code - Added Value**: optional extra. Additional marks are available for using:
    - The team should use a version control system such as Github, CVS etc.
    - Junit / Nunit for the implementation of automated test cases - 2 should be sufficient, TOKEN ACKNOWLEDGEMENT of the role of Test Driven Development (TDD) in Continuous Integration (CI).
    - Implementation support for either the database or concurrency using threads.
    - Include screen shots with brief description that visualise history of commits
    - Include coding fragments and/or screen shots with brief descriptions in this section of the report to illustrate implementation of Added Value such as TDD, concurrency, database, etc.
12. Recovered **architecture and design blueprints** based on the implementation but can also specify concepts that could be developed in future releases. Draw blueprints of:
    a. Architectural diagram
    b. Class diagram.
    c. One state chart for an object in either the sequence or communication diagram, with annotated transition strings.
14. Critique: compare and contrast the analysis sketches in (8) versus the blueprints created in (12).
15. References.

4. Notes and Guidelines

- Implementation usually follows on from design. Due to time constraints, we cannot adhere to this chronological ordering in this project because design will only be fully covered in lectures by the end of week 10. If we waited until Week 10 to finish the design phase of the project and then start coding, there would not be sufficient time for a meaningful implementation. Therefore, for this project only, implementation is guided by analysis artefacts - see section 8. We recover architecture and design post implementation.

- This assignment **constitutes approximately 45%** of the total marks awarded for this module.
- You will work in a team of 4.
- Please email the lecturer with a request to be allocated to a team if you are not in one by Friday Week 4. The subject should be 'CS4125: team allocation request'.
- The award of an F or NG will automatically result in the award of an F or NG for the module.
- **Submission deadline is 17:00 Friday 21st April 2017 – Teaching Week 12.**
- NO SUBMISSIONS WILL BE ACCEPTED AFTER THIS DATE!
- Place the submission in the appropriate post office box under the stairs in the Computer Science building.
- You will be required to provide the lecturer with a walk through of your project submission during an interview in Teaching Week 13.
- The project will not be marked if a walkthrough is not provided.
- Submission should be stapled or glue-bound with a clear front cover.
- As you will be required to submit an electronic version on a usb key of your submission - report and code, at the beginning of the walkthrough.
- **All team members must contribute EQUALLY to the implementation.**
- **All team members must contribute EQUALLY to the writing of the report.**
- Programming language and development environment is at your discretion.
- Presentation is critical. The report should be concise, easy to read, with diagrams that conform to aesthetic guidelines, i.e. minimise line crossings.
- If the presented work is not to the standard expected at this level, the submitted work will be severely penalised. Please ensure that:
    - Layout conforms to specification.
    - Page numbers in table of contents and throughout report.
    - Spell and grammar checks done.
- Analysis **sketches** can be drawn on paper, captured using a mobile phone camera, and inserted in jpeg/gif/etc form into the report.
- You must use a diagramming workbench for the architecture and design **blueprints** recovered in item 12 of the specification - numerous lists of community edition tools are available on the net, select one and justify your choice in the report. Using Word or PowerPoint is definitely excluded.
- You may **REUSE** content from an existing project, but you must:

    - Demonstrate added value – for example, by extending the feature set
    - Demonstrate an in-depth understanding of structure and dynamics.
    - Adhere to the rules with respect to the prevention of plagiarism.
    - Clearly differentiate between what already exists and added value.
        - See "Cite it Right", available through the UL Library Catalogue.
- Quality not quantity. Depth not breadth. Do not attempt to develop a full enterprise system! Instead, focus on a more constrained and feasible objective. For example, an accounts subsystem that interfaces with other subsystems in the enterprise.
- Marks are awarded for having attempted to use the methodologies and tools, for developing a lightweight proof of concept prototype, and evaluating the utility of diagrams. Marks are not awarded for the complexity or expansiveness of your selected scenario.
- **Separation of the system model (business classes) and UI is critical where one elects to implement a GUI.**
- The lecturer reserves the right to make minor amendments to the assignment up to and including week 8. Amendments will be sympathetic to any work done to date.
- You should backup your work at all times. Accidental loss of work will not be accepted as an excuse.
- Plagiarism will automatically result in an F grade for the module and referral to the disciplinary team.

5. Note on Team Dynamics

- Problems with team dynamics should be reported to me in person A.S.A.P.
- If it becomes evident during the walkthrough that one of the team members did not contribute equally to the project, the following penalties will be imposed:
    - F for the non-contributing member
    - Other team members will be capped at a C3 for the other member because they did not bring the problem to the lecturer's attention.
- It is not your responsibility to deal with a team member who refuses to contribute fairly, that is the lecturer's job. However, it is your responsibility to report problems as soon as possible, which will be handled in a sensitive manner.

- Division of labour amongst team members is essential to a successful outcome, as is the necessity to commence immediately. Work clever.
- The primary mechanism to resolve problems with team dynamics will be to resort to individual submissions.

Collaboration between team members will be challenging and difficult. Consider using Zoho Projects / Dropbox / Google Docs / OneNote / etc. to document plans and progress - but keep it lightweight.

6. Labs

The purpose of labs is to provide guidance to teams. The lecturer/TA will effectively act as mentor.

The lab schedule for each team will be published Monday Week 4

Sample CS4125 Projects available on Sulis

7. Suggested Project Roles

Table 1

	Role	Description	Designated Team Member
1	Project Manager	Sets up group meetings, gets agreement on the project plan, and tracks progress.	
2	Documentation Manager	Responsible for sourcing relevant supporting documentation from each team member and composing it in the report.	
3	Business Analyst / Requirements Engineer	Responsible for section 6 - Requirements.	
4	Architect	Defines system architecture	
5	Systems Analysts	Creates conceptual class model	
6	Designer	Responsible for recovering design time blueprints from implementation.	
7	Technical Lead	Leads the implementation effort	
8	Programmers	Each team member to develop at least 1 package in the architecture	ALL
9	Tester	Coding of automated test cases	
10	Dev Ops	Must ensure that each team member is competent with development infrastructure, e.g. GitHub, Bamboo, etc.;	

8. Suggested High Level Project Plan

Table 2

Week	Workflow
3	Setup team roles, agree on scenario, learn to use Github, research on existing projects, etc.
4	Requirements
5	Analysis
6	High level architecture
7	Coding Iteration 1: basic infrastructure and 2 key use cases
8	Coding Iteration 2: 2 more use cases
9	Coding Iteration 3 Another use case and MVC (GUI)
10	Coding Iteration 4: Another use case and Added Value
11	Over run
12	Architecture and Design Recovery