CS4125: Systems Analysis & Design

Team-Based Project

Autumn Semester 2023 – 2024

J.J. Collins

25th September 2022 (Week 3)- **Version 1.0**

1. Objectives

a) To apply an Object-Oriented Analysis and Design (OOAD) method using the Unified Modelling Language (UML) to the modelling of a software system.

b) To identify the strengths and liabilities of the OOAD approach using the UML through study, practice, and reflection.

c) Support quality attributes through architectural and design patterns.

d) Gain insights into team dynamics and project management from the experience.

2. Scenario

Your organisation - KISS Solutions Plc., have been commissioned by a client to develop/extend a system. A key business strategy in KISS Solutions Plc. is that the engineering effort should yield production assets in the form of components that can be leveraged across future software product lines. A list of example client business scenarios is as follows:

- Traffic/Airport/Other Simulation
- Document/Games Framework
- Car/DVD/Hotel Reservation/Rental.

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 a retail system with differing discount policies for various customer categories.

3. The Report

You are required to submit a design document, which contains in the following order:

I Front cover with business scenario title, student names and IDs, and module details.

II Table of contents.

1. A one-page (maximum) narrative description of your business scenario.

2. A two-page (maximum) discussion on the software lifecycle adopted with a paragraph on resolution of tension between management of complexity and agility.
3. Project plan. Should be iterative. Should identify roles and responsibilities of each team member. Should clearly outline deadlines and deliverables.
    - Add a table that details industry/Coop experience for each team member specifying domain(s), programming language(s), frameworks, etc.
    - Specify in a table the primary role(s) and contribution(s) of each team member.
4. Requirements:
    - Functional:
        - Use case diagram(s).
        - For TWO key use cases only, submit a use case description based on the template distributed in the lecture notes Week 2.
    - Two quality attributes identified, as add-ons in the use case description, one of which MUST be extensibility
        - Include a discussion of support for the quality attributes i.e. quality tactics.
    - Two GUI prototypes – no more than that!
5. A three-page (maximum) discussion of system architecture to include at least one high-level package diagram providing an architectural perspective.
    - You must include the Model View Controller (MVC) architectural patterns.
    - You should also discuss technology pipeline that will be used to host deployment i.e. web server, application server, Enterprise Information System (EIS) aka database, message bus, etc.
    - You may include other architectural styles and/or patterns.
6. Initial/Conceptual Design [aka analysis]: **SKETCHES** using an object-oriented methodology with the UML:
    - List of candidate objects
    - Sketch of the conceptual class diagram, with significant methods and attributes identified. The sketch should demonstrate inheritance, aggregation and composition, associations, dependencies, visibility, interfaces with pre and post conditions, etc.
    - Sequence or communication diagram.
    - One state chart for an object in either the sequence or communication diagram, with annotated transition strings.
    - Entity Relationship diagram with cardinality.
7. **Transparency and traceability:**

- A table listing the packages, # classes per package, and the total packages and classes.
- A table listing the classes in each package, their authors, and lines of code.
- A table listing total lines of code implemented by each team member.
- A table that shows who did what with respect to patterns, testing, CI/CD, etc.
- A diary in tabular format that briefly describes, for each Week from 4-12 inclusive, the contribution of each team member for that week, issues arising, and the plan for the following week. This diary will be reviewed in the lab, see Table 1 for suggested format.

Table 1: sample template for the diary

Week X	Contributions	Issues Arising	Plan for next week
Mem 1			
Mem 2			
Mem 3			
Mem 4			
Notes			

8. Code Snippets and brief descriptions of:
   - Aspects of Model-View-Controller (MVC) architectural pattern, and the any other architectural pattern(s) implemented.
     - You are not expected to implement MVC in full. Focus on the business tier of the MVC. If time permits and two or more team members have expertise in front end dev, then consider building out the views element of the MVC.
     - No need to implement GUI and/or DB, unless doing so for added value.
     - Consider simulating DB with file I/O and DTOs/repository support.
     - Separation of the business classes and UI is critical – hence MVC.
   - 4 design patterns implemented, including Observer, Decorator, Factory Method and State.
   - Key use cases.
     - Implement those classes necessary to support a subset of the use cases.
   - Automated testing using Junit/Nunit/other
   - Visualisations that show use of version control.
9. Description and code snippets to illustrate ADDED VALUE – opportunity for independent learning. Examples:

- UI with REST architectural pattern,
- Concurrency through threading,
- Security/OWASP,
- DB implementation using Object Relational Mapping (ORM),
- Software metrics,
- CI/CD
- Microservices architectural pattern

10. **RECOVERED architecture and design blueprints** based on the implementation but can also specify concepts that could be developed in future releases. Draw blueprints of:
    - ❑ Design-time package diagram.
    - ❑ Design time class diagram.
    - ❑ Design-time sequence or communication diagram.
11. One page (maximum) critique on the quality of the design and implementation with a paragraph on the utility of diagrams, the efficacy of the UML, etc. How one could evaluate the architecture, and how to adequately document the architecture, etc.
12. References.

4. Submission Guidelines.

- This assignment **constitutes approximately 50%** of the total marks awarded for this module.
- Grading will be individualised and not team-based.
- **An F grade for the project will automatically result in an F grade for the module. Likewise for an NG!**
- **Submission deadline is as follows:**
    - **Sections 1-6: 23:59 Monday 23/Oct/23 (Week 7).**
    - **Full Report and code: 23:59 Friday 01/Dec/23 (Week 12).**
- Submissions will NOT be accepted after the deadline.
- Submission will be through Brightspace and in electronic format.
- You are required to work in a team of 4.
    - You must seek approval via email for teams of 3 or 5.
- **DO NOT go back** to edit sections 1-6 after the Week 7 submission to correct mistakes that become obvious as the implementation progresses as you will not be awarded additional marks.
- You may be required to provide the teaching team with a project walkthrough during Weeks 12-15. Failing to comply with this requirement results in an F grade.

- **Programming language and development environment is at your discretion.**
- You may **REUSE** content from an existing project, but you must:
    - Demonstrate an in-depth understanding of structure and dynamics of the sourced material.
    - Adhere to the rules with respect to the prevention of plagiarism.
        - See "Cite it Right", available through the UL Library Catalogue
    - Demonstrate added value – for example, by extending the feature set
    - Clearly differentiate between sourced content and added value.
- Analysis diagrams 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 **UML workbench** for items (10) of the report - 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.
- Presentation is critical. Should be easy to read, diagrams should adhere to aesthetic guidelines, i.e. minimise line crossings. Please check that:
    - Layout conforms to specification.
    - Page numbers in table of contents and throughout report.
    - Spell and grammar checks done.
- Quality not quantity. Depth not breadth.
    - Purpose of project is Learn by Doing → demonstrate your knowledge of the concepts in lectures.
    - Do not attempt to develop a full enterprise system!
    - The implementation is a light weight proof of concept prototype.
- Supporting quality attributes such as extensibility is key for a successful submission.
- The lecturer reserves the right to make minor amendments up to and including week 8.
- 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 COMMITTEE.

Start Now & Work Clever

All team members must contribute EQUALLY to the implementation.

All team members must contribute EQUALLY to the writing of the report.

5. A Note on Team Dynamics

- If it becomes evident during a code 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 capped at a C3**
- Division of labour amongst team members is essential to a successful outcome → WORK CLEVER, and do not focus excessively on ensuring fidelity of problem with real world scenarios.
- Problems with team dynamics should be reported to me in person A.S.A.P.
- 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 then handled in a sensitive manner.
- The primary mechanism to resolve problems with team dynamics will be to resort to individual submissions

6. Suggested Project Roles

Table 2

	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 / Reqs. Engineer	Responsible for section 4 - 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, etc.	

7. Suggested High Level Project Plan

Table 3

Week	Workflow
3	Setup team roles, agree on scenario, learn to use Github, research on existing projects, etc.
4	Requirements
5	Architecture
6	Analysis
7	Coding Iteration 1: MVC & 1 use case, test cases
8	Coding Iteration 2: two more use cases and design patterns(s)
9	Coding Iteration 3 another use case and design pattern(s)
10	Coding Iteration 4: Added Value
11	Over Run
12	Architecture and Design Recovery

8. Labs

- Your team will meet with the Teaching Assistant (TA) and/or the lecturer every second week in the lab to review work to date and discuss plan for next two weeks.
    - Teams in the Fowler Group: Weeks 4, 6, 8, and 10.
    - Teams in the Parnas Group: Weeks 5, 7, 9, and 11.
- The Lab Schedule will be published Tuesday Week 3.
- YOU MUST ATTEND YOUR ASSIGNED LAB
- **If you are not in a team by CoB Friday Week 2, please email me with a request to be allocated to a team. Subject must be "CS4125 – Team".**

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

Sample projects available Mon Week 4.