**2805 ICT**System and Software Design  
**Milestone 3 | Reflective Challenge Report**

**Student Name:** Mitchell Roles **Student ID:** s5132278 **Student Email:** [Mitchell.roles@griffithuni.edu.au](mailto:Mitchell.roles@griffithuni.edu.au)

Contents

[Objectives 3](#_Toc52967764)

[Objective 1 3](#_Toc52967765)

[Objective 2 3](#_Toc52967766)

[Objective 3 3](#_Toc52967767)

[Objective 4 3](#_Toc52967768)

[Objective 5 3](#_Toc52967769)

[Objective 6 3](#_Toc52967770)

[Objective 7 3](#_Toc52967771)

[Objective 8 3](#_Toc52967772)

[Objective 9 3](#_Toc52967773)

[Objective 10 3](#_Toc52967774)

[Implementations 4](#_Toc52967775)

[Task 1 4](#_Toc52967776)

[Task 2 4](#_Toc52967777)

[Task 3 4](#_Toc52967778)

[Version Control Logs 5](#_Toc52967779)

# Objectives Implementation

## Objective 1

**Explain the design principles including of least privilege and fail-safe default, separation of concerns, information hiding, coupling and cohesion, and encapsulation**

### Of Least Privilege

The principle of least privilege is a principle which involves the basic concept of: “A subject should only be given the privileges needed for it to complete its task”. In the terms of this project, it refers to the fact that objects should only be allowed to access to methods that it requires to function.

Due to the way the system was created, this concept has been implemented to some extent, through the C++ standard of class creation and usage. However, the principle of least privilege is mostly implemented through the systems class structure. Within the system only class methods with a pointer/reference to the object it wishes to interact with can access the methods of that object.

### Information Hiding

Information hiding, or data encapsulation, refers to the act of limiting the visibility of a class’ inner workings to other objects within a system. In terms of this project it means, how is the access to a classes attributes and methods restricted?

Within the current system, data encapsulation was one of the topmost priorities, as it helps to remove any undefined behaviour within the system and makes the task of tracking changes to attributes considerably easier. Data encapsulation was implemented through the built-in functionality of C++, using the respective “private” and “public” access modifiers to flag which methods and attributes should remain hidden from the rest of the system.

### Coupling and Cohesion

Coupling and Cohesion are two concepts within software engineering which describe the modulization of a system. Coupling refers to how two objects interact with each other, whereas Cohesion what a specific object can do. A focus of all software development projects is on minimising coupling and maximising cohesion.

Within the scope of this project, it can be stated that it has moderate to low coupling and high cohesion. The amount of coupling can differ from class to class, but generally speaking each class only interacts with each other by passing data back and forth. There are some exceptions to this, where a object will receive a pointer to another object, but these are few and far between and are only needed when an object requires constant information. Cohesion within each class is generally high, as each class will only perform actions within regards to its scope. An example of this would be “TileMap”, as it is a class which only involves the game map and is not involved with the managing of other classes.

## 

## Objective 2

**Describe the design process for a software development project for each of the main software design methods**

### Waterfall

The waterfall software design method is a more sequential approach to the development of a system, involving a steady “flow” downwards towards completion. It is a plan driven approach to software development, and usually produces a large amount of documentation as a by-product of the completion of each phase. Within the modern context, it is largely seen as outdated, mainly because of its inflexibility in the face of changes to the design requirements mid development

For every project using the waterfall method there a 5 processes/phases: Requirements, design, implementation, verification, maintenance. Due to waterfall’s rigid nature, the first stage of Requirements is the most important, as it usually dictates the quality of the project and the processes of development. It involves the development team systematically documenting every aspect of the project to define its constraints and scope. The next stage, design, involves the templating of the projects final desired outcome and structure. The following phase, implementation, involves the creation of the systems that were outlined in the design phase and usually requires the project to be fully functional before the phase can be considered as completed. This phase also requires regular progress reports on any all and changes to the system. The second last process, verification, involves extensively testing the system to ensure that it meets the requirements and functionality as specified by the client. Finally, the maintenance phase involves the upkeep of the produced software, the project can be considered as complete at this phase as the only alteration that should be made are ones the fix any undefined behaviour not caught during testing.

### Agile

The Agile development method is a more software oriented approach to the development of a system, prioritising the production of working software over documentation. It is an interaction driven approach to software development and aims to involve the customer at every step in the project aiming to further meet their needs.

Agile does not have any ridged processes, as it was design to be adaptive to the needs to the needs of the current project. Instead, at its most basic, the focus is usually on short feedback sessions, the implementation of working software and only producing as much documentation as needed. The feedback sessions involve the members of the development team discussing what was implemented and what will be the focus for that day. After the feedback sessions the teams will then focus on the production of working modules of software that can be implemented into the project as a whole. Within Agile, documentation is used as a method to further the understanding of the requirements and aims to only produce as much as needed, instead of the exhaustive approach of waterfall.

### Spiral Development

The Spiral Development method is a form of middle ground between Agile and Waterfall, with a heavy focus on risk analysis. It acts similar to an iterative development model with a few adjustments the, main one being an increase of the documentation of risks.

Spiral Development boils down to four basic processes: Determining objectives, identifying and resolving risks, development and testing, and Planning the next iteration. These four phases a performed iteratively, meaning that they can and will occur multiple times with the completion of an aspect of the project. The first phase is used to determine the goals of the project and to outline the required to steps to complete the current system. The second phase involves the documenting any risks found within the project and documenting the step required to resolve, or minimise, them. The third step encapsulates the development of the system and the subsequent testing of said system, whilst documenting the results. The final phase is used to prepare for the next iteration of development, by creating plans and documentation that will allow for a smoot transition.

### With Respect to the Project

The method that best describes the development of the current project would most likely be the Agile development method. The usual focus with each step of the project was to create functioning subsystems within the game, whether that be the player or the map, instead of creating broad foundations. This approach had its successes and shortcomings. The primary success was with the ease of implementation of new features, as the modularity of the implemented systems extradited the process. The primary drawback, was when extra functionality was required of the already implemented features, sometimes requiring their complete retooling.

## Objective 3

**Create appropriate system models for the structure and behaviour of software products from their requirements specifications**

## Objective 4

**Use a design paradigm to design a simple software system, and explain how system design**

**principles have been applied in this design.**

## Objective 5

**Select an appropriate software architecture as the design basis for a given software requirements specification, justify the selection based on its advantages over alternative architectures.**

## Objective 6

**Create software programs that make use of appropriate design patterns.**

## Objective 7

**Create user interface software using either event driven or call back based designs**

## Objective 8

**Explain the importance of Model-View controller to interface programming.**

## Objective 9

**Discuss the properties of good software design including the nature and the role of associated documentation.**

## Objective 10

**Create appropriate design documentation for a variety of different designs.**

# Task Implementations

## Task 1

## Task 2

## Task 3

# Version Control Logs

git log --stat > log.txt