WARWICK BUSINESS SCHOOL

# Masters Programmes

## Assignment Cover Sheet

**Submitted by:** *<2104720>*

**Date Sent:** 11/01/2022

**Module Title:** Programming Solutions for Enterprises

**Module Code:** IB96D0

**Date/Year of Module:** 2022

**Submission Deadline:** 11/01/2022

**Word Count:** 948

**Number of Pages:** 3

**Question:** *Reflective Report*

***"I declare that this work is entirely my own in accordance with the University's Regulation 11 and the WBS guidelines on plagiarism and collusion. All external references and sources are clearly acknowledged and identified within the contents.***

***No substantial part(s) of the work submitted here has also been submitted by me in other assessments for accredited courses of study, and I acknowledge that if this has been done it may result in me being reported for self-plagiarism and an appropriate reduction in marks may be made when marking this piece of work."***

# Reflective Report

This report aims to discuss the linkage between concepts taught in class and my personal experience of developing the room booking system, as well as explaining how these course objectives can be applied to my future career in the IT development industry.

Iterative development, one of the core values of agile development, states that the project scope becomes more clearly defined as the development progress increases (Schön et al., 2015). Initially, only three rough specifications, viewing the room details, booking the room and searching for available rooms based on certain criteria, are given in the guidelines, details of implementation and design of the system were not provided (Schön et al., 2015). This approach offered me higher flexibility, as I was able to redefine the scope at any stage throughout the development process (Winter et al., 2012). For instance, one would imagine that a real booking system requires a large database consisting of all available time slots over the next few months. Adhering to the principle of minimal design, all user booking data and available time slots are assumed to be limited to a particular day. This practice demonstrates the importance of responding to dynamic changes rather than following a fixed plan throughout the development process (Beck and Beedle, 2001).

Similar to other agile methodologies, iterative development not only allows incremental improvements at each development stage, but to also guarantee on-time delivery and maximize customer satisfaction (Schön et al., 2015). If I were to manage an IT project in the future, I would encourage project members to adjust the project scope continuously according to changing client requirements, and document them in the form of a storyboard (Winter et al., 2004).

Another key principle of agile development is “treating tests as a key resource”, utilizing techniques such as regression testing for test-driven development (Talby, Keren, Hazzan & Dubinsky 2006). Regression testing implies that all previous iterations should pass the test before starting a new iteration. When developing the booking system, I would make sure that all incorrect user input were being handled before moving on to the next functionality. For instance, exception handling is applied to warn users when they input wrong search criteria or wrong room number in order to prevent the program from crashing. For cases that users can input an incorrect input without showing any errors, while loops are effective in catching these mistakes, especially in situations where users are prompted a “yes” or “no” response. This practice makes sure users always provide an input that is recognizable by the program, so that they can navigate and book the rooms as normally expected.

Software testing is usually performed by DevOps engineers, who are responsible for designing and running tests on software components in short iterations. If I were to become one in the future, I would use regression testing to pinpoint errors resulted from software amendments and avoid bugs which are assumed to be fixed in previous maintenance stages (Siegel, 1996).

The last value of agile development worth mentioning is develop software with minimal functionality in hopes of getting feedback from clients in short, continuous intervals (B. Meyer, 2014). The initial requirement for the project is to develop a room-booking system just like the central University’s. The project could get complicated as it could be a full-fledged web application with frontend, back-end and database storage. However as suggested by “Manifesto for Agile Software Development”, the “simplicity” principle is

essential, in other words, limiting the work done to the smallest amount (K. Beck and Beedle, 2001). Therefore, in this project, I chose to deliver the solution in the simplest form, a command line program, requesting users to type in the search criteria, room number and other specifications through the prompt. A mini prototype was developed to satisfying all the compulsory requirements stated in the assignment guidelines.

To apply such methodology in software development, “limited negotiated functionality” is often emphasized to define the project scope with accordance to clients’ needs. Extra features will only add complexity to the project without adding any value to the client. Therefore, keeping the functionality simple is always the optimal way of achieving agility and optimal results with time efficiency.

The last course objectives applied in the project is related to modularity and code reuse. In this project, the room booking system is broken down into several functionalities, namely searching rooms based on user input, booking the room and checking whether a room is available, and recombined into an integrated software (Schilling, 2000). These functions are written on another module script and will be run in the main application script through passing arguments into them. Furthermore, a class “find available room” was written to factor the code, since similar functions can be applied to search for rooms based on equipment and available time slots. Code reuse is also applied in this case, since given the presence of these classes and functions in another module, duplicated code can be factored out, reducing the degree of complexity and increasing flexibility in designing the software (Baldwin and Clark 2000).

This demonstrates the importance of object-oriented programming (OOP), especially in terms of encapsulation and abstraction, as methods can be defined within classes which are hidden from implementation. As mentioned in previous studies, code reuse could enhance the software quality by minimizing the chance of generating errors and need for modifying the code (Mohagheghi and Conradi, 2008). Therefore, if I were given the chance to develop a large-scale software in the future, I would apply the principles of OOP to avoid repeating the same code multiple times for developing software that are less prone to errors, and at the same time, enhance the code base’s readability for future developers.

## References


Baldwin, C. and Clark, K., 2002. The Option Value of Modularity in Design: An Example fromDesign Rules, Volume 1: The Power of Modularity. *SSRN Electronic Journal*,.

Beck, K. and Beedle, 2001. *Manifesto for Agile Software Development*. [online] Agilemanifesto.org. Available at: <https://agilemanifesto.org/> [Accessed 30 December 2021].

Meyer, B., 2014. [online] Available at: <https://www.researchgate.net/publication/287382991_Agile_The_good_the_hype_and_the_ugly> [Accessed 30 December 2021].

Mohagheghi, P. and Conradi, R., 2008. An empirical investigation of software reuse benefits in a large telecom product. *ACM Transactions on Software Engineering and Methodology*, 17(3), pp.1-31.

Schilling, M., 2000. Toward a General Modular Systems Theory and Its Application to Interfirm Product Modularity. *Academy of Management Review*, 25(2), pp.312-334.

Schön, E., Escalona, M. and Thomaschewski, J., 2015. *Agile Values and Their Implementation in Practice*. [online] Ijimai.org. Available at: <https://www.ijimai.org/journal/bibcite/reference/2515> [Accessed 30 December 2021].

Siegel, S., 1996. *Object Oriented Software Testing: A Hierarchical Approach|Shel Siegel.*. [online] Hotelsoftheworld.info. Available at: <https://hotelsoftheworld.info/Object-Oriented-Software-Testing:-A-Hierarchical-Approach%7CShel-Siegel.cgi> [Accessed 30 December 2021].

Talby, D., Keren, A., Hazzan, O. and Dubinsky, Y., 2006. Agile software testing in a large-scale project. *IEEE Software*, 23(4), pp.30-37.

Winter, D., 2004. *Persona driven agile development: Build up a vision with personas, sketches and persona driven user stories*. [online] Academia.edu. Available at: <https://www.academia.edu/53074045/Persona_driven_agile_development_Build_up_a_vision_with_personas_sketches_and_persona_driven_user_stories> [Accessed 30 December 2021].