End-of-Module Reflective Assignment

e-Portfolio: https://rob-essex.github.io/portfolio/OOIS/

Reflection on Object-Oriented Programming based on lessons learned

Prior to this course, I was purely a self-taught programmer. My projects were instigated out of a business need, and thus, understanding the inner workings of object-oriented programming was not a priority. My knowledge, like the code I designed, was cobbled together out of necessity. Though I continued to learn over the years, I lacked the foundational knowledge that I have gleaned over the past few months.

During this course, I have learned to appreciate the details of "how," instead of focusing solely on "what" code does. I have begun to recognise and to write elegant code. Furthermore, I have learned that programming languages require just as much time and attention to learn as human languages. Computer programming will be a life-long endeavour to improve not only my capabilities, but the efficiency with which my code communicates with machines and humans, alike (Martin, 2010).

Reflection on work in the module

Assignments in this module helped me understand the evolution of object-oriented software development – and system development as a whole. Prior to this course, my software planning and design efforts were grossly insufficient. Learning about Unified Modelling Language (UML) and Entity Relationship Diagrams (ERDs) has enabled me to clearly design and communicate system specifications. The System Design and ERD in Unit 7 gave me the opportunity to apply these techniques to real-world business cases.

Similarly, the System Implementation assignment in Unit 11 gave me an opportunity to apply lessons learned from course seminars, Lecturecasts, and Codio exercises. This assignment has reinforced my belief that there will *always* be more to learn about optimising code. Despite the many days I spent working on this assignment, I would have liked to continue improving the code, documentation, and testing.

Reflection on the impact on my professional and personal development

I haven't learned this much about computer science since I was a child, deconstructing MS-DOS computer games to see if I could cheat my way through them. Whilst I never stopped learning over the decades, this course has filled many of the gaps in my self-education. Studying object-oriented information systems has reinvigorated my desire to learn; no longer will I resign myself to ignorance because a technology seems too complex. Learning the fundamentals of object-oriented programming in this module has put useful software development within my reach.

Professionally, my analysis of technical problems at work has become more methodical and deliberate. By properly planning the systems development lifecycle (SDLC), from requirements gathering to UML design to implementation and beyond, my ability to estimate the time and resources needed for a given solution has improved. Understanding how to design and plan object-oriented information systems has also enabled me to communicate, and thus, delegate assignments across a development team.

Alignment to Learning Outcome 1: Appraise and evaluate critically the concepts and principles of information systems

Units 1 and 2 provided a great introduction to information systems. The reading assignments provided useful overviews of information system concepts and principles, and the group discussion reinforced this material by looking at real-world examples of information systems gone awry. This introduction allowed me to assess information systems throughout the module, culminating in the System Implementation assignment during Unit 11 and the study of emerging trends in information systems during Unit 12.

Alignment to Learning Outcome 2: Design or modify and document an object-oriented information system using appropriate tools

The System Design assignment was my first experience translating end-to-end business requirements into a UML class diagram. Furthermore, it was my first time designing a whole system at once including objects, attributes, methods, and relationships.

Prior to this course, my system designs – on the few occasions I did any design, at all – were done on a whiteboard, notepad, or in PowerPoint. Learning about UML guided me to a slew of purpose-built tools for software development. I started with a tool called SmartDraw since I had previously used it for flow charts; however, further experimentation led me to Lucidchart. Lucidchart offered the right balance of simplicity whilst also allowing me to customise diagrams and develop a cleaner product.

Alignment to Learning Outcome 3: Develop an object-oriented information system design, implementing this knowledge in applicable programming languages, such as Python and SQL

Learning the “normal forms” of database design demonstrated how to optimise schemas in relational databases such as SQL (Idelson & Dany, 2014) . Creating an ERD in third normal form (3NF) during Unit 7 advanced my ability to plan and communicate data structure requirements.

The System Implementation assignment forced me to think through the practical application of object-oriented information system design. I had to determine the best way to translate a UML diagram into code, and in doing so, I realised that drawing an arrow from one class object to the next is much easier than scripting it into existence. Some cases, such as inheritance, are easy – there is only one way to define a subclass from a super-class. However, determining the best way to enforce multiplicity between class objects, or the most practical method to implement composition, comes with a lot more nuance. It also presents challenges to developing *exactly* what a design document states if there are ambiguities in the design.

Alignment to Learning Outcome 4: Develop, implement and evaluate critically information system solutions to facilitate business decisions

The System Implementation assignment forced me to make decisions about the practicality of my code. Since there was no information about who the users would be, nor how they would interact with the object model, I limited development as closely to the

class diagram as possible. I understood that humans may not interact with this code, at all – that it could be used as an application programming interface (API), and that functions like input normalisation and error handling may occur in a manner outside of my Python code (Shatnawi et al., 2018). This activity underscored the importance of integrating business process owners and stakeholders in the design, development, and testing phases.

References

Idelson, M. and Dany, C. (2014). *Freeing entities of attributes: Revisiting database normal forms from an organizational knowledge perspective.* [online] IEEE Xplore. doi:10.1109/ICMIT.2014.6942406. [Accessed 27 May 2022].

Martin, R.C. (2009). *Clean Code: A Handbook of Agile Software Craftmanship*. Upper Saddle River [Etc.] Prentice Hall.

Shatnawi, A., Shatnawi, H., Saied, M.A., Al Shara, Z., Sahraoui, H. and Seriai, A. (2018). *Identifying Software Components from Object-Oriented APIs Based on Dynamic Analysis*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/8973109 [Accessed 27 May 2022].