Unit 3

Fundamentals of Object-Oriented Design (Reflection)

This week's consideration focused on the concept of Object-Oriented Design (OOD). We considered terminology and concepts. Also, this unit saw participation in a design activity where I put to the test my understanding of designing an object-oriented model. For programming exercises, OOD was explored in the context of how Python uses OOD principles.

Collaborative Discussion Summary

Summarising my engagement in the discussion forum and research, I garnered a broader understanding of why information systems (IS) fail, the main cause being a failure to test for all scenarios followed by the very nature of complexity. *Why do I reach this conclusion?* First, based on the experiences and case studies provided by other students, mixed with real-world observations, it seems that the amount of money thrown into a project does not correlate with its reliability or resilience to failure. Second, failures tend to occur in situations that were not predicted or accounted for during design, revision, and isolated testing.

Nelson (2007) classifies IS failures into four categories: people, process, technology, and product which I agree with, though I would exclude “product” from that list because several processes come into play to *deliver* a finished product and organisations ought to improve on quality and risk management practices at this point.

Reflections on UML Modelling

This week, I developed a UML object model based on a fictitious supermarket. Unfortunately, the activity did not specify the complete list of requirements, leaving requirements open to interpretation in the modeller’s mind. Nevertheless, this was a great activity to engage in since

it forced me to consider how objects are influenced by the design of classes when modelling a system. I preferred to approach this activity by starting with a handful of English sentences to describe the system. From there, I modelled the objects and presented the initial diagram for other students to comment. I was mindful of the book The Elements of UML 2.0 Style and wholeheartedly agree with many recommendations made in the book. A few techniques heavily influence me to ensure my models are easy to read: reduce overlapping lines; use the same shape sizes; group elements-of-a-feather together; prefer straight lines over a mixture of styles.

The words "object**"** and "class" in object-oriented terminology are not synonymous. A class is akin to a "template", while an object represents a single instance of a given class. My experiences in workplace conversations are that "object" is used to refer to every business entity. I believe this is driven by the fact that "object" is merely a "thing" in English vocabulary. But in software engineering, it very much means something different. In contrast, **domain-driven design** terminology uses the words **entity** and **value** to describe objects with *identity* ("entity") or that contain pure data ("value") (Evans 2004).

References

Evans, E. (2004) *Domain-driven Design*. Westford, Massachusetts: Addison-Welsley.

Nelson, R.R. (2007) IT project management: Infamous failures, classic mistakes, and best practices. *MIS Quarterly executive* 6(2).