Module Reflection

Eportfolio Link

https://micjustus.github.io/essex-eport2/

Reflection on Learning and Teamwork

This module dealt with securing information systems, using secure programming concepts, applying security to testing, using linters to identify code issues, cryptography, securing APIs and the role of microservices architect versus monolithic architecture. The SSD module was the first module where students worked together to develop a team project. The project aimed to emulate working in a world scenario by analysing a project's requirements, drafting a system design document and finally, delivering on the requirements within the project's delivery schedule.

In the team project, I took up the role of project manager, which was new to me, given I have no matching work experience. However, I leveraged career observations to guide what was required from this role. Although I desired to work as an architect, it was better to forego this role and allow other team members to take up the mantle to gain exposure to what this role involves. As a project manager, I leveraged Google Calendar in Slack to ensure our team had regular meeting notifications. I also found valuable the need to be cognisant of each team member's availability, providing regular communication to ensure we were all aligned concerning requirements, deliverables, and timeline.

As we set about looking at the initial design document requirements, it reminded me of the development-to-Scrum mapping exercise undertaken in Unit 2 (Sharma & Bawa, 2020). This research ensured security requirements remained foremost in our thoughts as we developed the project design proposal. During the requirement analysis phase, collaboration was vital in discussing each member's input and reviewing the customer's design brief. I found it valuable that the team maintained an open attitude that permitted each member to contribute to the project. This attitude of

openness leads to greater engagement, as identified by Dutra et al. (2015), who note that high-performance teams leverage motivation, knowledge, attitude, and communication skills.

After agreeing to the requirements, the team started coding the implementation in Python. This exercise raised a few unexpected issues in Python, such as the inability to assign class variables inside a `@staticmethod` function. The Python anomaly caused me to reconsider how secure the Python programming language is if one cannot set class variables inside a “static” method. I found it more of an annoyance than a security feature. Unit 8, focused on leveraging cryptography, was helpful for the team as we considered which data is “sensitive”—passwords, for example—and must be stored securely. We investigated a few cryptographic libraries and settled on bcrypt as providing sufficiently powerful protection. Also considered was the information we displayed to a user, such as the need to mask their password characters.

The implementation project also required developing an API alongside the monolithic codebase. I took responsibility for the authentication and authorisation code; this was both painful and enjoyable in Python. It was painful because of the above “`@staticmethod`” anomaly, difficulties debugging in Codio and frustration with the constant mix-up between spaces and tabs. And yet enjoyable because authentication eventually worked as expected, with the authorisation functions performing admirably.

The team worked well together; however, we encountered a few obstacles related to the Codio environment that caused loss of code and a few associated with MySQL. None of these were significant issues since we discussed the problems and overcame them together. Because the project was monolithic, the decision was to divide work tasks according to *functionality* so that each member had sufficient work while minimising the impact on other members. We divided the workload according to (1) authentication, (2) entries/DB and (3) API. This approach worked well, despite initial concerns it did not reflect real-world allocations. It allowed each member to “specialise” in their area, which was tremendously beneficial as we knew who to communicate with when issues arose with (1), (2) or (3). During the initial phase of the team project, an idea emerged to rotate roles in the second part of the project.

However, we decided against this approach for the implementation, favouring instead (based on agreement) each member's "experience" gained in a role.

I found the idea of "ontologies" an intriguing aspect of this module since it closely ties with the concept of "architecture". While an ontology is a hierarchal specification of concepts related to entities and their associated properties, architecture concerns itself with "the structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time". (TOGAF, 2011). When challenged to develop an ontology, I consulted the module tutor for further guidance, leveraging their feedback and research by Kendall (2013). I also considered whether UML is suitable for visually representing ontologies and found that the OMG group has a UML profile (https://www.omg.org/odm/). Research by Gaševic et al. (2019) was also valuable in understanding ontology design in UML.

Finally, looking back over the content of this module, the reference to OWASP Top Ten and Mitre (2021) helped broaden awareness of software vulnerabilities. While the module touched on secure programming languages, my research into this topic convinced me that no programming language is 100% secure. For instance, while Python has fewer vulnerabilities today than Java (WhiteSource, 2021), its most common vulnerabilities are input validation, access control; permissions; privileges and cross-site scripting. Linter tools are interesting because they help improve the quality of Python code. However, I found many of their reportable errors related to indentation. References to the McCabe cyclomatic complexity algorithm reinforced the idea of writing small, beautiful code functions—the idea is to "fail fast" to reduce cyclomatic complexity.

Overall, secure software development is successful if software developers, managers, business analysts and customers ensure that security is a first-class citizen in their thinking because people are generally seen as the weakest link in a security chain (Hadlington, 2017). This module also highlighted many resources available that ensure one minimises code vulnerabilities by using the latest packages, checking memory bounds, or considering risks at each development stage.

References

Dutra, A.C., Prikladnicki, R. & França, C. (2015). What do we know about high performance teams in software engineering? Results from a systematic literature review. *2015 41st Euromicro Conference on Software Engineering and Advanced Applications:*183-190.

Gaševic, D., Djuric, D. & Devedžic, V. (2009). Model driven engineering and ontology development. Springer Science & Business Media. Available from http://www.odbms.org/wp-content/uploads/2014/02/Gasevic_chapter1.pdf [Accessed 31 Oct 2021]

Hadlington, L. (2017) Human factors in cybersecurity; examining the link between Internet addiction, impulsivity, attitudes towards cybersecurity, and risky cybersecurity behaviours. Heliyon, 3(7), p.e00346. DOI: https://doi.org/10.1016/j.heliyon.2017.e00346

Kendall, E. (2013) Ontology Engineering in UML: Best Practices & Lessons Learned of a Working Ontologists. Available at https://www.omg.org/news/meetings/tc/dc-13/special-events/semantic-pdfs/T1-1-Kendall.pdf [Accessed 20 Oct. 2021]

Mitre (2021). Common Weaknesses Enumeration. Available from https://cwe.mitre.org/data/ [Accessed 16 Aug. 2021]

Sharma, A. & Bawa, R.K. (2020) Identification and integration of security activities for secure agile development. International Journal of Information Technology: 1-14

TOGAF (2018). The TOGAF Standard, Version 9.2: Part 1: Introduction. Core Concepts. Available from https://pubs.opengroup.org/architecture/togaf9-doc/arch/index.html [Accessed 31 Oct 2021]

WhiteSource (2021). What are the most secure programming languages? Available from https://www.whitesourcesoftware.com/most-secure-programming-languages/ [Accessed 31 Oct 2021]