Portfolio I

Theodor Baur

02/03/2022

—

CM1301: Principles, Tools, and Techniques for Secure Software Engineering

Section 1: Reflection on Class Material

Across the course of this term we have been looking at methods to make code more reliable. About 2 weeks ago we had a series of lectures which gave me insight into different proccesses that should be undertaken to help improve reliability as well as a look into how to categorise and identify problems with code. The different focuses that can be used to improve reliability can include *Incremental Development, Rational Unified Process, Open Source and Agile Devlopment.* This section was important as it introduced me to the concept of *Incremental Development* where software is developed in smaller increments and then tested for reliability after each increment is completed, instead of the whole project. This is useful for identifying where errors are as they should be contained to each new increment.
My previous method for coding would be similar to the *Waterfall*, where I would develop my code first and test reliability at the end, often missing some errors.

I applied the idea of testing for developing incrementally when doing my Java exersize based on creating classes used to calculate the cost of carpet. Each time I would complete a method within one of my classes I would run some basic reliability tests and ensure that any errors were taken care of before developing the next increment, in this case each increment was a method. I notice several benefits of using this approach. As expected, it was far easier to identify where errors took place as, if one were to take place it would have had to occure due to the newest increment. It did seem take longer to write the code as I had to check its reliability more frequently, (after every method instead of after the whole task was completed) however this may be because I often do not fully check the reliability of my code. In addition, when using *Waterfall* development, each error takes far longer to solve as it is harder to identify what cause it.

Section 2: Independent Research

Open source software is a type of software that is distributed alongside its original source code, allowing the user to modify and distribute the software freely. Distributing software through open source is common practice, for example Linux distributions are some of the most widely used operating systems. They run opposed to traditional, closed source software where the a user only has access to a program's excecutable binaries. This can have a number of practical and buisiness benefits and drawbacks.

Some of the main practical benefits are that code is often more reliable, secure and quality *(Morgan and Finnegan, 2007).* The reason that open source software seems to be more reliable is because any broken code or bugs are likely to be found quicker due to the larger amount of developers scrutinising over the project. This is more applicable to larger projects with more contributors. Similarly, the security is likely to be better as any exploits are likely to be spotted early due to the exposure that open source projects receive. The technical quality of open source projects are said to be better as there are more testers who are often of high quality as they are developers themselves. It is important to know that these benefits may only apply in a limited manner towards smaller open source projects with little traffic.
In addition to technical benefits, open source projects can provide businesses with reduced costs in terms of virus protection and paid testers and developers. It can also establish de facto standards within an industry, for example a Linux kernel can be seen as a standard which may allow other developers to shift their focus onto other priorities.

However this is not to say that open source software does not have certain drawbacks. For example, on the technical side, different branches of software that have been developed from the same source can have compatibility issues with each other. On the business side there seems to be a certain risk to using open source software as there is little accountability for a piece of software that malfunctions due to there being no company to back it up, only a development team. Therefore it is clear that open source distribution has certain inherent benefits and drawbacks.

Appendix

- Morgan, L. and Finnegan, P. 2007, *Benefits and Drawbacks of Open Source Software: An Exploratory Study of Secondary Software Firms*, Availiable at: https://link.springer.com/content/pdf/10.1007%2F978-0-387-72486-7_33.pdf/ [Accessed: 2 March 2022].