Software Testing Research Report

Testing software prior to release is essential to ensuring users avoid negative experiences as a result of errors or defects when using the software, and although it is difficult to verify the software can handle *all* possible eventualities, the employment of various testing models can enable developers to minimise and mitigate these faults before they impact user experiences, and give feedback to assist in making decisions around further development around quality.

The ISO/IEC/IEEE 29119 standard[1] proposes that testing is a process; it should be preplanned, but continue throughout and beyond the development of software. For this project, this model is adapted such to make it easier to follow for the smaller number of developers involved, and simpler given the relatively minimal complexity involved with the scale of the project when compared to that which the standard was designed.

A key principle to the testing model used is that it is *agile*, and so can be used in conjunction with the SCRUM-based development cycle. One recommendation for agile testing is that as much of it should be automated as possible[2] One particular difficulty with using the LibGDX framework and libraries is that it is not trivially compliant[3], [4] with JUnit5 [5], a white-box style unit testing framework for Java projects. It instead requires significant additional configuration and a *headless* implementation due to LibGDX's dependency on OpenGL for rendering components.

It can additionally be difficult to ensure code reliability when developing in an agile way, particularly with a limit placed on unit tests, as an increased number of smaller updates can lead to a higher number of opportunities for the overall software code to fail (such as through concurrently modifying a feature and its dependency). Regular regression testing must therefore be performed throughout the project to catch errors or bugs arising from these code updates.

Not all errors can be caught through regression testing, and therefore it is important to have additional points of software verification and testing throughout the project. By performing a *code review,* it is possible to catch significant issues in code prior to their implementation in the main program - done practically in this project by requiring a different person to approve the merging of branches than the developer committing the changes.

Bibliography

[1] ‘ISO/IEC/IEEE International Standard - Software and systems engineering –Software testing –Part 1:Concepts and definitions’, *ISO IEC IEEE 29119-12013E*, pp. 1–64, Sep. 2013 [Online]. Available: 10.1109/IEEESTD.2013.6588537.

[2] M. Costick, (Nov. 26, 2013), ‘Agile testing at the Home Office’, Government Digital Service Blog. [Online]. Available https://gds.blog.gov.uk/2013/11/26/agile-testing-at-the-homeoffice/ [Accessed: 02 January. 2020].

[3] thisisrahuld, ‘Unit Testing Libgdx applications’, *BadLogicGames Forum*. Jul. 27, 2015 [Online]. Available https://www.badlogicgames.com/forum/viewtopic.php?f=11&t=20103 [Accessed: 01 January. 2020].

[4] T. Pronold, *TomGrill/gdx-testing*. 2019 [Online]. Available https://github.com/TomGrill/gdx-testing [Accessed: 01 January. 2020].

[5] *JUnit 5*, JUnit. [Online]. Available: https://junit.org/junit5/ [Accessed: 02 January. 2020].