**How will we validate our game for our customers?**

**Software Testing**

Lets first describe where we are, namely that our mobile application game is at the point that its’ first release is ready, and we need to validate this to get it released. We are using an Incremental approach to development using Agile methods. We are also using integration for some of our code, that we may have modified to work with our project. In short, we have a working version of our game, even it is a little on the simple side of what it could develop into. At this point we can add some more into how we developed our application, namely Test-Driven Development (TDD). TTD is an approach to program development in which you interleave testing and code development (Beck 2002; Jeffries and Melnik 2007). “Test-driven development was introduced as part of the XP Agile development method. However, it has gained mainstream acceptance and may be used in both agile and plan-based processes” (Sommerville, 2016). This means that we have developed tests for code while we were developing our code. This puts us well on the way to testing our game.

The tests that we developed while we were coding using TTD will provide a core set of tests to validate our application using unit testing. We will be using automated testing using these tests thru an automation framework such as JUNIT. Another way that we will validate our code is like pair coding. Instead the two of us sitting at the same computer while coding, we will each develop tests and code for a certain task, then the other person will inspect the code before the automated tests are run by JUNIT. This will give us the benefits of pair coding and inspection and let us separately work on different parts of the application. Using these methods will give us excellent code coverage as our application develops.

As we have a complete working version of our game we can finish off with release testing. Release testing should not be done by the development team. This could cause a problem for a very small company currently consisting of only 2 members. We can do our own in-house Alpha testing ourselves to see if it works at least. However, being recent graduates of the B.C.S. program from TRU we have contacts from TRU that can be our release testers. Let our friends do the release testing for us by letting them have pre-release copies of our game. This would be essentially our group of Beta Testers. We then get feedback from them anonymously thru submission thru the software or by setting up a web form or a forum where they can leave comments that we can review later.

Add some more references…

**References:**

Beck, K. (2002). *Test Driven Development: By Example.* Boston: Addison-Wesley.

Jeffries, R., and Melnik, G. (2007). *TDD: The Art of Fearless Programming*. IEEE Software 24: 24- 30. Doi:10.1109/MS.2007.75.

Sommerville, I. (2016). *Software engineering*. Harlow, England; New York: Addison-Wesley, 2016.

**Old Thoughts:**

With our software on the market our first line of testing is customer feedback and bug reports. Our software keeps track of the moves that have been made during a game in memory. Only when the customer decides to do a bug report does this data get written to a file and sent with the bug report. Locating and fixing the error that causes bug reports is the priority. Many users playing the will give testing of the whole application. It also tests the part of the code that is important to the game play experience. This will also give of feedback as to the playability of the game and sill give comments the will help with the improvement of the user interface. Another advantage of user feedback and bug reports is that they may test combinations of the code that we may not have thought of. User feedback will help to do the testing of the release application.

Examining the feedback and bug reports then gets to the need identify the cause of the individual reports. This will require us to identify which part of the code caused the report. This will involve running the complete code in a error testing environment were we can input the same moves the player did while watching the variables while it walks thru the code. Once we have identified where the error is and what input has caused it we can make the needed correction. It could be that in input was not checked as valid, or that there is an error in the logic of the algorithm.

The next thing to test would be any new functionality we are adding or parts we are improving. Any new or revised modules can be run thru test for that individual module.