**A comparison of the testing done during software development under a waterfall model with that of an agile model**

Testing is a vital aspect of the System Development Life Cycle (SDLC). It involves a series of tests being carried out on the system to check its functionality. There are multiple different variants of the System Development Life Cycle with two common examples being the waterfall model and the agile model.

The waterfall model takes a linear approach to the development of the system, breaking the process down into specific stages. One of these stages would be for the testing of the system. This method is a traditional way of using the System Development Life Cycle.1 If this approach were to be taken, the tests would be defined very early in the system’s development. The tests would be agreed on by the developer(s) and the customer(s). The customer will need to ensure that the system is capable of doing the tasks they want to be done and the developer will need to ensure that these tasks are feasible. The tests would then be carried out after the system has been developed and the results would be documented so that the developer.2 This stage is also commonly known as the validation and verification stage as it involves testing the system’s functionality according to the initial specification in order. This then gives the customer and developer a good idea of the system’s validity under the given circumstances.3 After reviewing the results, it could then be agreed whether the system is complete or whether there is more work to be done. When the waterfall model is used, there will not often be much flexibility when it comes to changing the system after the testing has been carried out. The agile model would provide more flexibility in this area.

For more complex systems, the agile model offers a slightly different approach to the development of the system. The agile model involves several iterations of the process to ensure that the system is complete rather than only using one iteration like the waterfall model. However, the stages carried out in each iteration are usually the same as the stages used in the waterfall model.4 The agile model could be seen as a better approach than the waterfall model in larger systems especially because it requires the system to be tested regularly throughout the process.5 This means that the system can be developed quicker because if it does not meet the initial requirements at any time, the developers will know soon after writing that section of the program rather than having to wait until the end of the process, which could potentially lead to other errors in the system as a result of the first error which would mean that the process would take much longer.

The stage at which the testing is carried out is one of the main differences between these two models and would often play a big part in choosing which method would be best to use for a specific problem. The agile method would be best used for larger systems that can be broken down into smaller tasks. This way, each of the smaller tasks can be tested individually. However, if the system is fairly small anyway, the agile method might not need to be used. It would take longer to plan out the development if this model were to be used which would not be the best use of time on the development of smaller systems.

The agile method also provides a lot more flexibility. Again, this could be considered an advantage or a disadvantage depending on the size of the system. Smaller system will often need to carry out only a few set tasks. These tasks would usually be quite specific so there would not be much room for flexibility without changing the specification of the system. However, in a larger system with more requirements, flexibility could prove useful and could result in the system being developed quicker than it would if it had to meet lots of very specific requirements. On the other hand, this would also increase the importance of having a feasible specification. If this were not the case, many iterations would have to be carried out before the customer is happy. The emphasis here is on the developer having a full understanding of what the customer wants from the system.

To conclude, both of these models offer valid approaches to the System Development Life Cycle. The model that is used would depend on the specific requirements of the system that is being developed.

1 Lott, C.M.L. (1997) Breathing New Life into the Waterfall Model. IEEE Software [online]. 14 (5), pp. 103-105. [Accessed 25 November 2015].

2 Ullah, M.U., Ullah, F.U. and Shehzad, M.S.S. (2011) Enhancement of stakeholders participations in Water fall Process Model. International Journal of Computer Science and Information Security [online]. 9 (5), pp. 78-80. [Accessed 25 November 2015].

3 Bassil, Y.B. (2012) A Simulation Model for the Waterfall Software Development Life Cycle. International Journal of Engineering & Technology [online]. 2 (5), pp. 742-749. [Accessed 25 November 2015].

4 Trivedi, P.T. and Sharma, A.S. (2013) A Comparative Study between Iterative Waterfall and Incremental Software Development Life Cycle Model for Optimizing the Resources Using Computer Simulation. Information Management in the Knowledge Economy [online], pp. 188-194. [Accessed 25 November 2015].

5 Ji, F.J. and Sedano, T.S. (2011) Comparing Extreme Programming and Waterfall Project Results. Software Engineering Education and Training [online], pp. 482-486. [Accessed 25 November 2015].