|  |
| --- |
| **Group M302** |
| David Voqui |
| Flor Ardaneh |
| Jeremy Romana |
| Sricharan Bonda |
| Amnah Harahsheh |
| Nazli Khajehamirlou |
| Sara Sahebjam-Atabaki |
| Hari Hara Sudhan Venkateswaran |



WATERFALL Vs AGILE SOFTWARE DEVELOPMENT

Fall 2017 Project Report

ISE 167: System Simulation

Industrial and Systems Engineering

**Contents**

[Executive Summary 2](#_Toc500760381)

[Description 3](#_Toc500760382)

[Agile Process Flowchart 5](#_Toc500760383)

[Waterfall Process Flowchart 6](#_Toc500760384)

[Model Construction 7](#_Toc500760385)

[Input Data 8](#_Toc500760386)

[Decision Points 9](#_Toc500760387)

[Run-Time Parameter Selection 9](#_Toc500760388)

[Verification 11](#_Toc500760389)

[Model Output 12](#_Toc500760390)

[Scenarios 13](#_Toc500760391)

[Results & Analysis 15](#_Toc500760392)

[Base Scenario 15](#_Toc500760393)

[Waterfall method has a lower cycle time resulting in a higher throughput. 15](#_Toc500760394)

[Conclusion 16](#_Toc500760395)

[Appendices: 17](#_Toc500760396)

**Executive Summary**

KMM Software has contracted a team of industrial engineers to evaluate their software development process and approach. They are currently using a Waterfall software development approach, but are considering switching to an Agile development approach. Some of the key factors they would like to consider when deciding between the two options are benefits and costs. Our team of industrial engineers has provided a recommendation based on the simulation models and testing performed on each software development approach.

Our team has developed simulations for both Agile and Waterfall software development approaches and have recorded metrics and statistics for each method. Furthermore, our team has also tested some realistic hypothetical situations in the simulation that may occur when each method is implemented. With this, we aim to deliver a comprehensive recommendation that will be catered towards several scenarios in addition to the base.

The use of a simulation software here aids in providing a cost and time effective method to mimic the real-world conditions and thus providing a decisive advantage to the company in their business ventures. For each simulation model, we have followed the process maps, processing times, and scrap rates.

The performance metrics chosen to compare the two processes are throughput, cycle times and resource utilization. From the simulations models, we determine that the Waterfall software development approach yielded greater throughput along with shorter cycle times in comparison to the Agile approach. The ideal approach for each of the hypothetical scenarios however, vary depending on several factors. Since each scenario requires specific changes to the processes, the outcome and metrics differ for each approach.

The model was also tested for three scenarios where changes occurred in project scope, early release and loss of a team member in each process. Performance metrics were recorded for all these scenarios and compared for each software development method.

# **Description**

           To effectively evaluate the potential costs and benefits of switching to an Agile approach, a simulation will be built for both the Agile and Waterfalls methods of software development. In addition to building a simulation for each method, we will also implement certain hypothetical situations to see how the simulation responds to common realistic situations and issues. Upon analyzing the costs and benefits both simulations, our group will recommend whether to switch to an Agile method or continue developing software through the Waterfall method.

           The purpose for this project is creating a simulation model for both agile and waterfall methods for releasing the end product, in order to examine the method gives better results for the company after all. We will simulate and run the model, with several what if scenarios as well, which will be stated later in this report. And we will present the results to KMM to decide whether they will continue with waterfall method or switch to agile method. Following are the assumptions for developing the model:

• Simulations will follow process maps provided for both Agile and Waterfall methods for software development.

• Simulations will utilize provided processing times and scrap rates (decision points data) for both Agile and Waterfall software development methods.

• A triangular distribution will be utilized for processing times for Waterfall or the Agile project steps in which the minimum, maximum, and mode is provided.

• A uniform distribution will be utilized for processing times for Waterfall or the Agile project steps in which only two parameters are listed.

• A constant value will be utilized for processing times for Waterfall or the Agile project steps in which only one parameter provided.

• Both projects begin at the same time.

• New projects begin every quarter

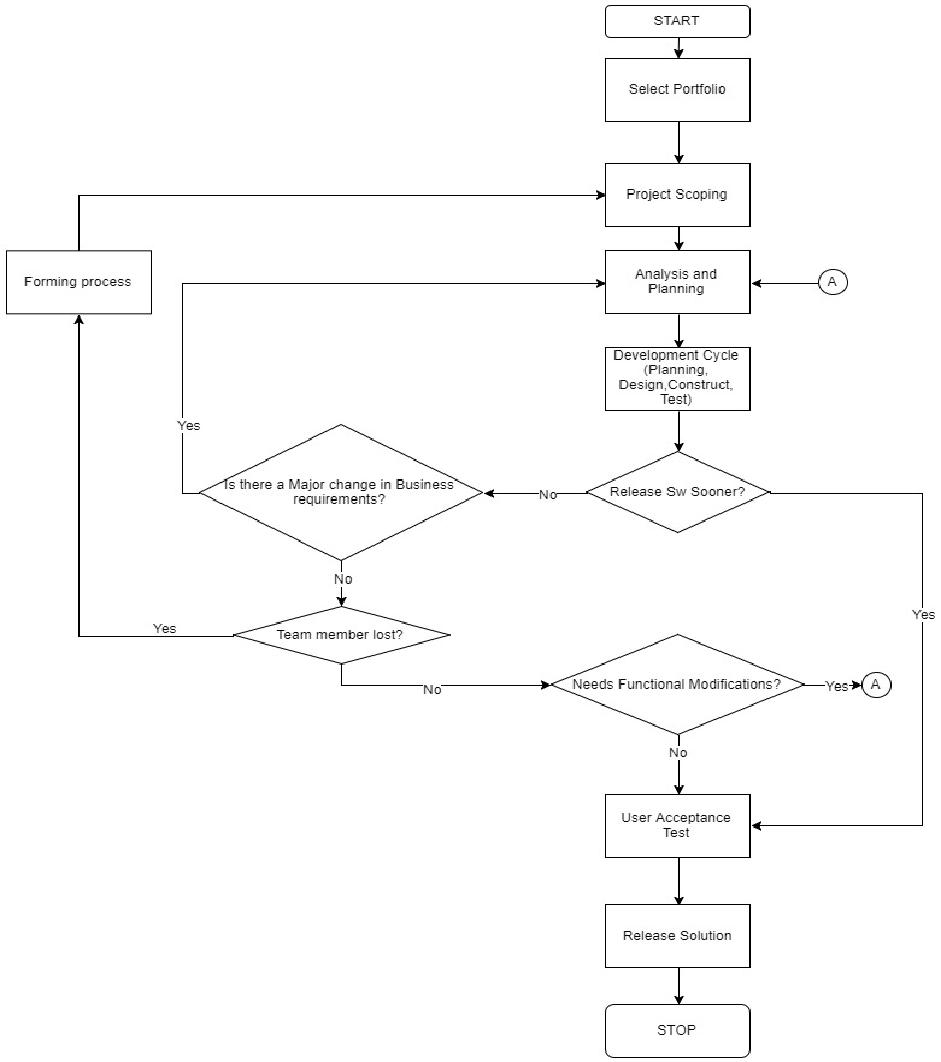
• The technology used in developing the software will stay the same as previous, for both Models.

• The software developing equipment, computers, areas of working etc... will stay the same as previous, for both models.

• The networking system, and the contracts with other companies will stay the same.

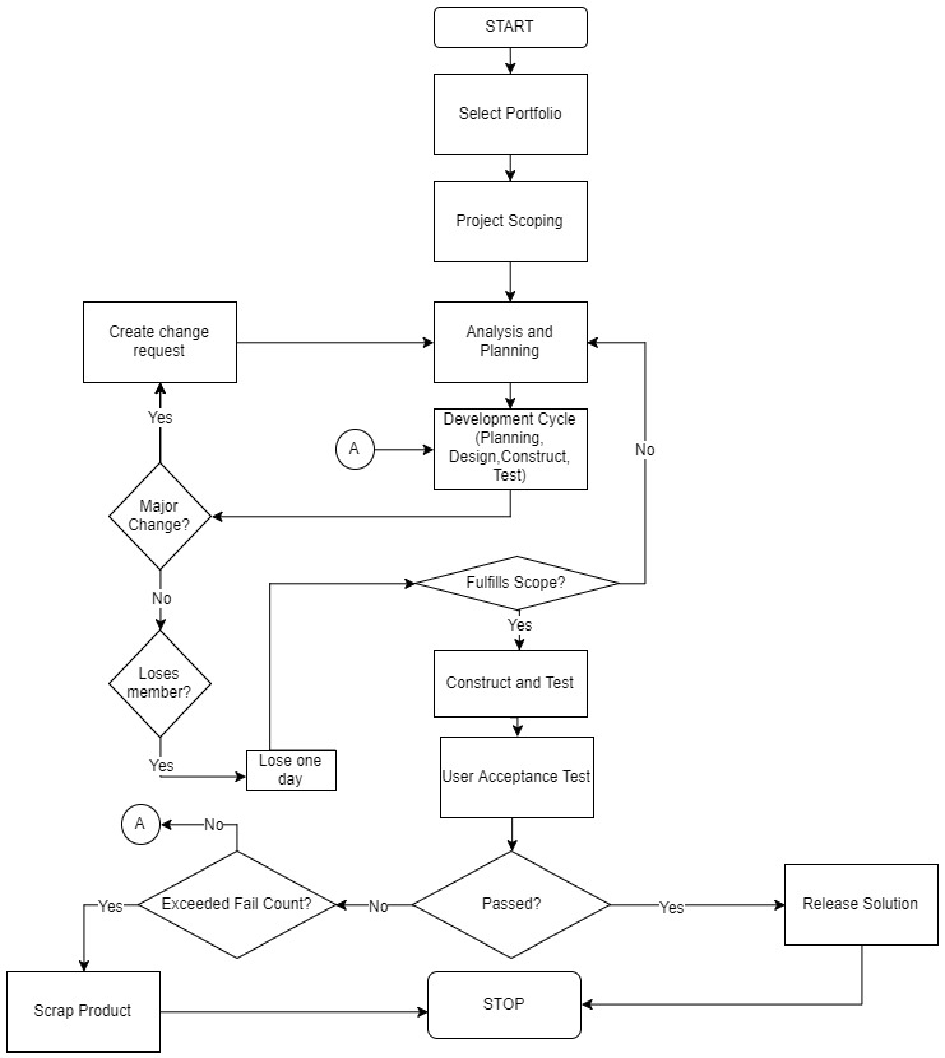
• The business strategy, advertising, and the market of consumers will stay the same.

**Agile Process Flowchart**



*Figure 1:Flowchart for Agile*

# **Waterfall Process Flowchart**



*Figure 2:Flowchart for Waterfall*

# **Model Construction**

Locations

* Portfolio selection
* Project scoping
* Analysis and planning
* Design, construct and test (Design and Construct independent for Waterfall)
* User acceptance test
* Release solution
* Queues in front of each location

Entities

In this model, projects are the entities, which moves through various processes.

Arrivals:

Projects arrive into each process method at a frequency of 91 days.

Resources:

Refer the input table no 1.

Processing:

Agile:

Entity (Project) moves through the locations and processing will take place at each location as per the various probability distributions given in the input. Based on the condition given, design construct and test location will have a condition check, entity which satisfies the above will move to user acceptance testing location, entity which does not satisfy will be routed back to analysis and planning location with a higher priority than the new project which is entering the analysis and planning, through this we are making sure the entity which entered the system first will be given more priority than the entity which arrives later.

Waterfall:

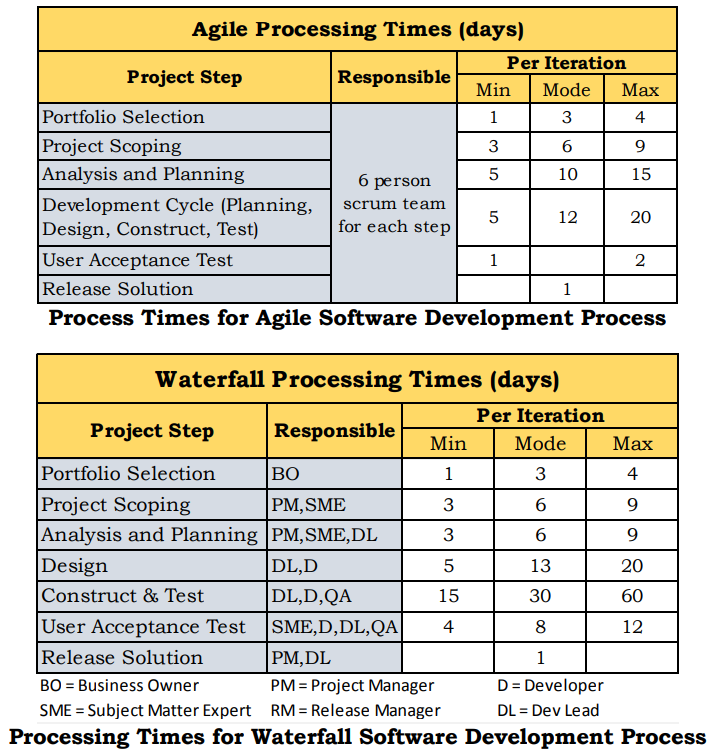
Entities moves through the locations as per the probability distributions, user conditions are checked at locations design, construct and test, and user acceptance test and routings/passes are made accordingly. Since all resources are not used in all the locations, to ensure that the oldest project is worked on first, priority values are provided to the resources based on the stations.

# **Input Data**

           The input data provides the processing times and decision points for development processes. The following tables represent the input data of process times for agile and waterfall:

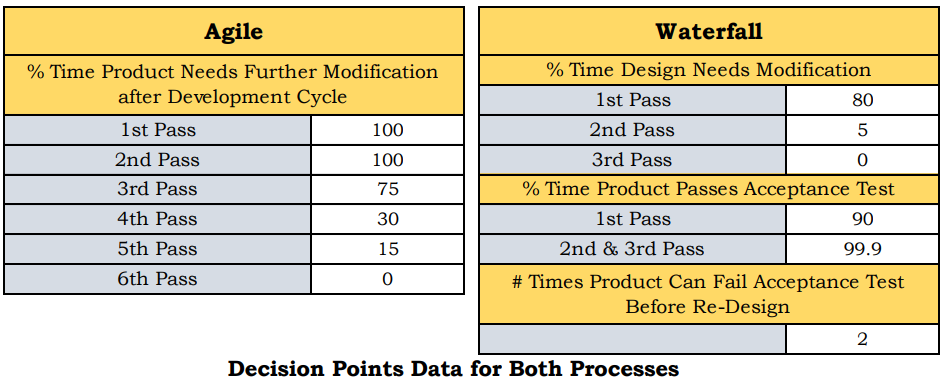
https://docs.google.com/drawings/d/sDZu2HWzgB2anadG-vazxhg/image?w=457&h=1&rev=1&ac=1

Figure 3



# **Decision Points**

Figure 4



           As seen in Figures 1 and 2, the process map for agile is relatively shorter. An agile attribute requires modification after development, while the entity (software in this case) follows six passes. For instance, the first and second pass have a probability of 100% needs modification, meaning 100% off times it will go back in the process. In the third pass, the probability becomes 75%. Therefore, we can claim that by the sixth pass, the entity has gone through enough modification, and the probability of it needing more modification will be equal to 0. After the software passes, it goes through two “customer acceptance” processes, and finally ready to release.

           The waterfall process map has three attributes, that the entity (software) goes through in in the related passes.  In the first one, 80% of the time the entity needs modification on the first pass, 5% on the second and finally, 0% on the third pass. After this stage, the customer acceptance test starts, which indicates that 90% of the time, the product passes the customer acceptance test in the first pass, and 99.9% on the second and third passes. Finally, each product can only fail the customer acceptance test twice, before it goes to be redesigned (The construct and test location.)

**Run-Time Parameter Selection**

To decide about the parameters of the model, we run the model with run time of 6months first, and compared the throughputs of warm ups of 1month, 2months, 3months and 6months. The results for 3 months were relatively close and that was our decision for the warm up time. We did the same process about the run time, warm up time for 3months and the number of replications 10, and we run the model for 1 year up to 20 years. The throughput results were relatively close around 15 years, and that was our decision about the run time. And finally, we tried different numbers of replications from 10 to 20, and decided on 15 replications based on the throughputs.

         As a summary of what was stated in the previous paragraph, here are our parameters for the model:

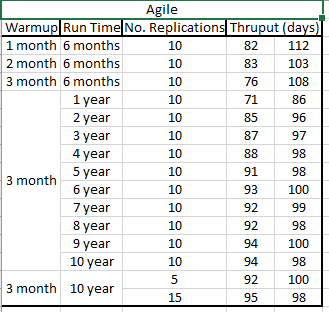
Warmup: 3 months

Runtime: 10 years

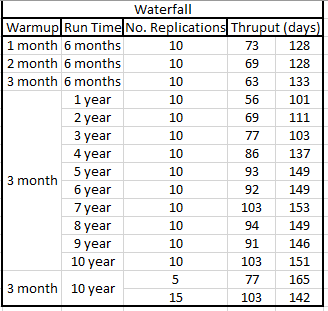
No. Replications:15

The following are the summary of tables of decisions for Parameters of the model:

*Table 1: Parameter decision for Agile*



*Table 2: Parameter Decision for Waterfall*



# **Verification**

           We need to ensure the conceptual model is reflected accurately in the computerized representation. Verification helps to validate the model construction. Is the model being implemented correctly in the simulation software? Are input parameters and logical structure of the model represented accurately?

To demonstrate the model accuracy, we need to analyze the model outputs. The following performance measures are used for this purpose:

• Utilization of resources

• Total time each model needs for developing

• Maximum and average queue Length for each station

• Number of resources

• Defect generation and resolution rate

• Average time to fix for failure in case it happens in each station (why is it important? Because fixing bugs earlier is cheaper)

           We determine the average number of passes that we will go through during a project’s lifecycle. Each pass in agile and waterfall is modelled as a probability value and the sum of all the probabilities are added together and considered as average cycle time. The average processing time for each station is the mean for each process based on its probability distribution. Multiplying the average cycle time to the average processing time for each station would give us the total time spend in each station. The sum of the time spent in all the stations is our theoretical calculation of the expected time a project takes to complete in Agile and Waterfall processes.

Table 3 represents the theoretical calculation:

Table 3: Verification for Agile Process

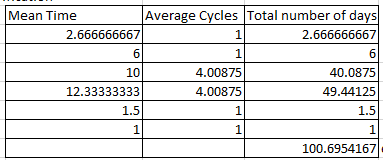
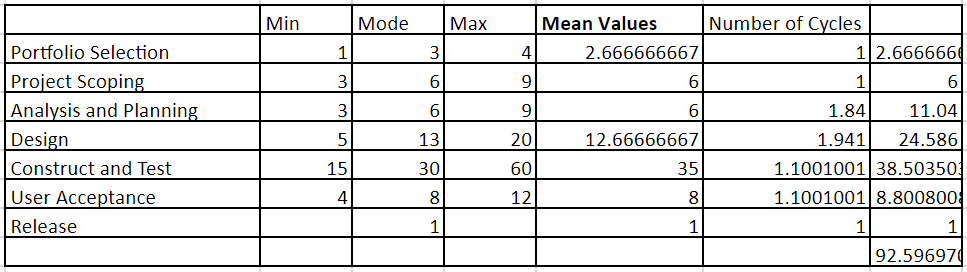


Table 4: Verification for Waterfall Process



# **Model Output**

Having selected our runtime parameters and constructing the model, the results are tabulated as shown below.

Throughput

Table 5: Throughput Comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **AGILE - Base Scenario** | | **WATERFALL - Base Scenario** | |
|  | 95% Low | 95% High | 95% Low | 95% High |
| Throughput | 35.56 | 37.11 | 39.68 | 40.19 |

Cycle Time

Table 6: Cycle time comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **AGILE - Base Scenario** | | **WATERFALL - Base Scenario** | |
|  | 95% Low | 95% High | 95% Low | 95% High |
| Cycle Time | 97.28 | 101.25 | 89.53 | 91.10 |

Resource Utilization

Table 7: Resource Utilization Comparison

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **AGILE - Base Scenario** | |  | **WATERFALL - Base Scenario** | |
|  | 95% Low | 95% High |  | 95% Low | 95% High |
| Agile Team | 99.60 | 100.07 | BO | 2.94 | 3.08 |
|  |  |  | PM | 36.76 | 56.34 |
|  |  |  | SME | 40.94 | 58.91 |
|  |  |  | DL | 90.51 | 94.57 |
|  |  |  | D | 77.45 | 81.13 |
|  |  |  | QA | 50.61 | 53.32 |

The model output for cycle time follows the theoretical calculation performed in table 1 and 2 thus validating the correctness of the model. From the verification table, Agile takes approximately 101 days to complete a project and therefore, dividing our runtime 3650/101 = 36.13 projects is the expected throughput. The model output follows the theoretical throughput for Agile. Following the same approach, Waterfall takes approximately 93 days which gives a throughput of 3650/93 = 39.2 projects. Based on the output from the model, we can safely say that our model follows the theoretical calculation. Refer appendix for ProModel results.

# **Scenarios**

Aside from the base scenario, further testing is required to check the effectiveness of each process. For this, three scenarios were designed to check the capability of each methodology under the respective real-world changes.

**Scenario 1**

           In this experiment, the effect of change in business requirements in the middle of project was studied. In the Agile model, the work during development cycle was stopped to review the change in requirements and converting those into user stories. This change led to increase in Agile development cycle, which consists of design, construct, and test. On the other hand, there was a 2-day increase in the design phase of Waterfall. Also, in the case of Waterfall, the change created the need to go back to the beginning of project in project selection.

Table 8: Comparison of Output for Scenario 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **AGILE - Scenario 1** | |  | **WATERFALL - Scenario 1** | |
|  | 95% Low | 95% High |  | 95% Low | 95% High |
| Cycle Time | 96.37 | 100.20 | Cycle Time | 89.16 | 99.62 |
| Throughput | 35.91 | 37.56 | Throughput | 24.50 | 38.30 |
| Agile Team | 99.75 | 100.00 | BO | 2.96 | 3.09 |
|  |  |  | PM | 33.83 | 57.55 |
|  |  |  | SME | 38.28 | 60.24 |
|  |  |  | DL | 89.68 | 94.62 |
|  |  |  | D | 77.29 | 86.17 |
|  |  |  | QA | 51.61 | 68.40 |

           Comparing this scenario to the base scenario, Agile had to have a total of 7 cycles instead of 6 in the development process whereas the project in Waterfall did not go through an extra cycle and instead it was sent back to project selection which means the project goes through one extra step during cycles and not an extra cycle. As a result, the throughput remained the same for Agile, but it increased for Waterfall compared to baseline.

**Scenario 2**

In the second experiment, we tested the effects of a decision to release the software earlier than originally planned. We evaluated the changes in throughput rate for both Agile and Waterfall. In the Agile model, the project is sent to user testing after reaching the halfway point in the project. This was determined by looking at the average completion time of agile in the baseline scenario. In the Waterfall model, the project is sent to user testing when a stable product cut-off point is reached. Similar to Agile, the cut-off point was determined by looking at the average completion time of the waterfall project in the baseline scenario. To account for such changes, the simulation was changed by updating the code to route the project to user testing when the halfway point in the project is reached.

Table 9: Comparison for Scenario 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **AGILE - Scenario 2** | |  | **WATERFALL - Scenario 2** | |
|  | 95% Low | 95% High |  | 95% Low | 95% High |
| Cycle Time | 90.15 | 90.23 | Cycle Time | 89.35 | 90.56 |
| Throughput | 41 | 41 | Throughput | 40.67 | 41.06 |
| Agile Resource | 63.92 | 64.41 | BO | 2.93 | 3.07 |
|  |  |  | PM | 20.02 | 20.9 |
|  |  |  | SME | 27.21 | 28.02 |
|  |  |  | DL | 70.25 | 73.08 |
|  |  |  | D | 56.95 | 60.01 |
|  |  |  | QA | 30.83 | 34.76 |
|  |  |  |  |  |  |

The given parameters shorten the project duration substantially. This is because many cycles are skipped in the waterfall model when the cut-off point is determined. Similar to the baseline model, the Agile model still goes through all the cycles in this experiment. Once the halfway point is reached, no further modifications are made to the software, regardless of what development cycle the project is currently in.

**Scenario 3**

In the third experiment, we tested the effects of change in the team composition for each process. The real-world implication for Agile is generally higher than for Waterfall. Agile teams are usually skilled workers who work on the project as a whole for every process. Hence, the loss of a team member means that the replacement should be brought up to speed about the project. For Waterfall method, since all resources don’t work on all the processes, the knowledge loss is lower and hence a replacement can be brought up to speed much faster.

Using the above effects, the simulation is designed to have a higher impact on Agile. The simulation is designed so that the Agile team waits 7 days and then goes back to project scoping thereby increasing the number of days in the process. The waterfall team however is modelled to lose just one day around the halfway point and then continues forward with the project.

Table 10: Comparison for Scenario 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **AGILE - Scenario 3** | |  | **WATERFALL - Scenario 3** | |
|  | 95% Low | 95% High |  | 95% Low | 95% High |
| Cycle Time | 112.40 | 114.89 |  | 89.69 | 97.13 |
| Throughput | 31.06 | 31.88 |  | 24.37 | 39.50 |
| Agile Resource | 99.99 | 100.00 | BO | 2.94 | 3.10 |
|  |  |  | PM | 35.04 | 60.34 |
|  |  |  | SME | 38.91 | 62.61 |
|  |  |  | DL | 90.40 | 95.11 |
|  |  |  | D | 77.64 | 86.49 |
|  |  |  | QA | 50.61 | 68.41 |

The effect of the above design is shown in the results table above. The cycle time increases significantly for the Agile team while the Waterfall team’s cycle time stays near the base scenario cycle time. The change in cycle time also inversely affects the throughput as shown by the significant decrease in the output from Agile process.

Complete results in Excel sheet submitted online.

# **Results & Analysis**

# **Base Scenario**

# Waterfall method has a lower cycle time resulting in a higher throughput.

Agile method has a higher cycle time resulting in lower thoughtput and the addition of a higher queue time.

**Scenario 1** – Major change in Business requirement

Both methods go through an additional cycle resulting in lower cycle time. However, Waterfall method’s cycle time is lower than Agile despite having to go back to project scoping. This is due in part to the number of cycles that the project goes through in Agile. On an average, project on agile goes through 4 cycles while in Waterfall, the average number of cycles is 2.

**Scenario 2** – Releasing the software faster

The Agile method goes through lower number of cycles in this scenario allowing for a faster cycle time. In Waterfall, since construct and test is the stable cut off point, the reduction in cycle time is not significant. Agile’s cyclical approach to product development helps deliver the product faster in this scenario.

**Scenario 3** – Change in team composition

Agile is significantly slowed down due to the additional wait time for forming process and the need to go back to project scoping, thereby losing progress in project development. Agile’s need for a tightknit group significantly affects the loss of a team member.

Waterfall is not significantly affected by the loss of a team member and so its cycle time is only slightly longer than in the base scenario.

# **Conclusion**

Based on the simulation model’s results, we recommend KMM software to continue to utilize Waterfall software development approach due to having a better cycle time and the method’s flexibility when it comes to loss of member in the team. The reduced number of passes also ensure that a major change in the business requirements from the customer would not adversely affect the Waterfall method’s cycle time or throughput.

# **Appendices:**

Table 1: Pass Probability calculations for Agile

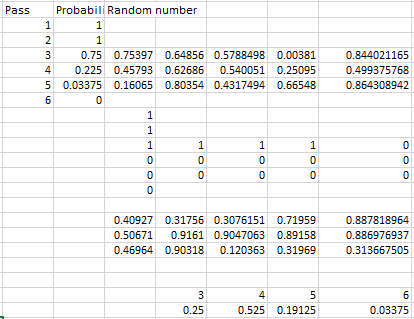


Table 2: Agile spreadsheet

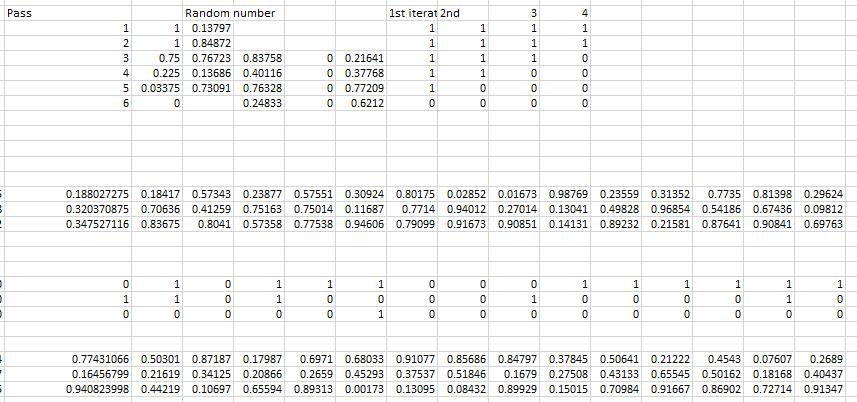


Table 3: Pass probability calculations for waterfall

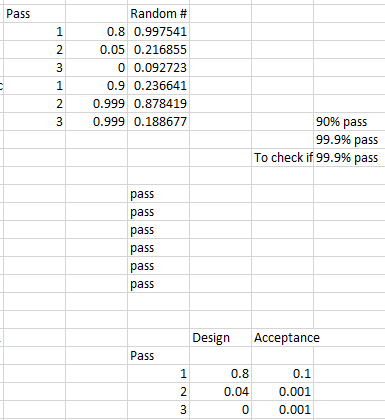


Table 4: Waterfall spreadsheet

