PRogramming II AT1.1

Kyer Potts 3003389

Table of Contents

[Waterfall Model 1](#_Toc7422962)

[Waterfall Model Pros 1](#_Toc7422963)

[Waterfall Model Cons 1](#_Toc7422964)

[Agile Model 2](#_Toc7422965)

[Agile Model Pros 2](#_Toc7422966)

[Agile Model Cons 2](#_Toc7422967)

[Recommendation 3](#_Toc7422968)

[Waterfall Model 3](#_Toc7422969)

[References 4](#_Toc7422970)

## Waterfall Model

Waterfall methodology consists of sequential steps or phases of development that follow a linear progression. These steps vary depending on the requirements of the system, but will usually consist of logical sequences for planning, design, construction, testing and implementation, deployment and maintenance.

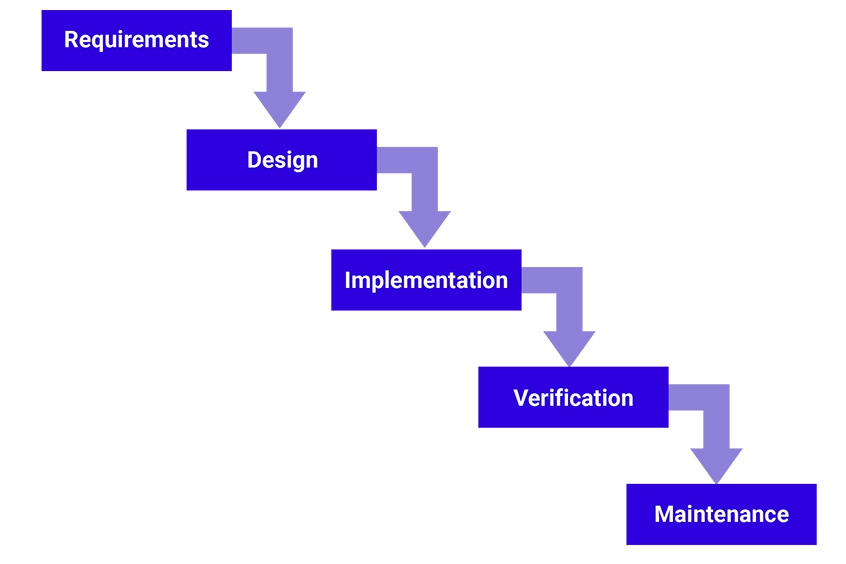
* The requirements stage sets up the framework and planning for the project and focuses on generating the specifications and setting the scope.
* The design stage covers any of the aspects of the technical design of the program, and results in the architectural framework that the code will follow.
* Once the planning and design stages have completed, the product will enter construction. During this stage the product will be coded and built according to the specifications and requirements of the previous stages
* When the build has completed, the project enters the testing and implementation stage. Any issues with the initial build will be discovered and reported to be resolved. Any bugs found are resolved and if necessary the testing phase will be cycled until as many of the bugs as possible are removed from the build.
* Once construction and testing has completed the product will be deployed to the customer or to market. Often the product will need to be maintained or supported by the developers to ensure functionality.

### Waterfall Model Pros

* Ease of use
* Clearly defined stages
* Clearly defined goals
* Organisationally simple
* Processes and results are well documented throughout stages
* Structure of the model ensures simple management of deliverables

### Waterfall Model Cons

* Testable or usable software is not available until late during the life cycle
* High risk and uncertainty factor due the rigid nature of the model
* Simplicity of the model does not mesh well with complex object orientated projects
* Structure of the model makes changing requirements difficult to implement
* Measuring progress of the product development is difficult
* Integration is done at the end of the project and doesn’t allow identification of technology or business bottleneck or other issues earlier within the development life cycle.



## Agile Model

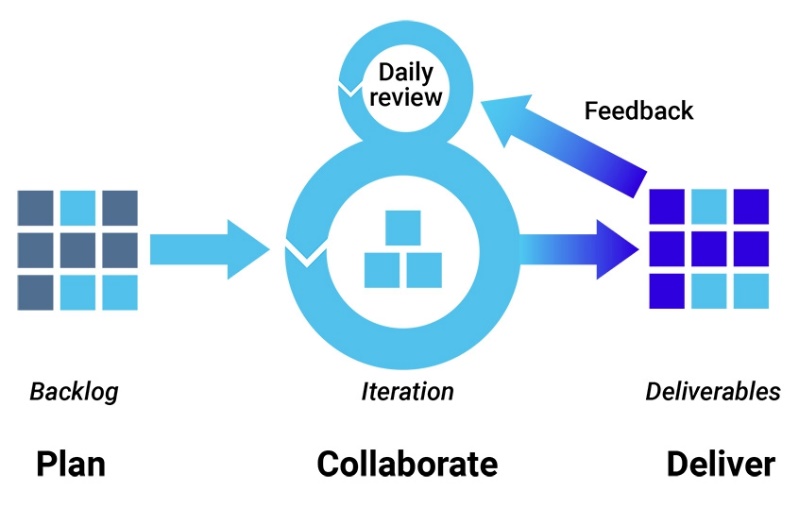
Agile methods segment the project into incremental builds which are provided in iterations. These iterations usually last several weeks, allowing deliverable versions of the software to be produced to the client for continual sign off until all software requirements are met. Daily reviews or “scrums” are held to keep staff on task during iteration cycles. These iterations are produced by cross functional teams working simultaneously on various areas of the life cycle of the project. This ensures that the requirements, design, development and testing and implementation stages are all worked on concurrently.

### Agile Model Pros

* Promotes teamwork and cross training
* Rapid development of functionality and features
* Low resource requirements
* Adaptable to changing or added requirements
* Delivers basic working products incrementally
* Documentation is easily employed
* Little to no planning required before development begins
* Allows the developers to be flexible with the project
* Minimises scope creep
* Development progress is easily measured

### Agile Model Cons

* Difficult to handle complex dependencies
* Risk of sustainability and maintainability of the project
* A high-level plan and appropriate team leader is required to implement agile methods
* Deadlines for deliverables require strict adherence
* Depends on customer involvement with the project. Deliverables require continuous sign off during the development of the project
* Individual responsibility of each team member is high as there is little documentation for them to follow.
* Transfer of technology to additional team members can be difficult as little documentation is produced
* Requires strict adherence to code practices to ensure that code is uniform and readable across all areas of the project



# Recommendation

### Waterfall Model

Considering the requirements and scope of the program, the waterfall model would be the most appropriate choice. It seems unlikely that the requirements will change or that scope creep would be detrimental to the project in this instance, and the waterfall model will allow partitioning of resources effectively to produce the product.

# References

Synopsis Editorial Team. (2017, March 28). *Software Integrity Blog*. Retrieved from Synopsys: https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/