**FORM A1**

**STUDENT PLAGIARISM DISCLAIMER FORM**

****

**PLAGIARISM DISCLAIMER**

STUDENT NAME: Raphael Salaja

STUDENT NUMBER: A00269349

PROGRAMME: Bachelor Science (Honours) Software Design with Virtual Reality and Gaming

YEAR: 3

MODULE: Software Engineering

LECTURER: Michael Russell

ASSIGNMENT TITLE: Assignment 1

DUE DATE: 26th November 2021

DATE SUBMITTED:

ADDITIONAL INFORMATION:

I understand that plagiarism is a serious academic offence, and that AIT deals with it according to the AIT Policy on Plagiarism.

I have read and understand the AIT Policy on Plagiarism and I agree to the requirements set out therein in relation to plagiarism and referencing. I confirm that I have referenced and acknowledged properly all sources used in preparation of this assignment. I understand that if I plagiarise, or if I assist others in doing so, that I will be subject to investigation as outlined in the AIT Policy on Plagiarism.

I understand and agree that plagiarism detection software may be used on my assignment. I declare that, except where appropriately referenced, this assignment is entirely my own work based on my personal study/or research. I further declare that I have not engaged the services of another to either assist in, or complete this assignment.

**Signed: Raphael Salaja**

**Dated:**

Contents

[1](#_Toc88561270)

[Introduction 3](#_Toc88561271)

[Testing Integration 3](#_Toc88561272)

[Customer Involvement 3](#_Toc88561273)

[Nature of The Processes Steps 4](#_Toc88561274)

[Ability to Handle Change 4](#_Toc88561275)

[Conclusion 5](#_Toc88561276)

[**DEVELOPMENT CYCLES COMPARISIONS SHEET** 6](#_Toc88561277)

# Introduction

When creating products in the software industry, teams employ various Software Development Life Cycles (SDLC) to produce high quality pieces of software. Each version of a SDLC has their own nuances and differences between them. It is up to the team to use and employ the most suitable SDLC for their respective product.

In this essay I will compare the characteristics of the following SDLCs; Waterfall Model, Iterative Model, Spiral Model, V-Model, Big Bang Model, RAD Model, and the Software Prototyping Model. The characteristics that I will discuss are the testing integration, customer involvement, nature of the processes steps and the ability to handle change.

# Testing Integration

Testing is a crucial part of the development of a product in the software industry. Testing will allow for errors to be found. If a process integrates testing into its lifecycle it will be able to produce a product of greater quality than those that do not.

In the Waterfall Model, testing is integrated after the implementation stage and before the deployment stage. In this stage all the units developed during the implementation phase are tested thoroughly before deployment. The V-Model varies slightly as it is an extension of the Waterfall Model and is based on the association of a testing phase for each corresponding development stage. This ensures a maximum amount of quality. Both models described previously, vary greatly from the Big Bang Model. Testing here is not implemented at all unless it is implemented under the user’s discretion. This model does not ensure that a product will come out with the greatest of quality.

Testing and debugging can be done easier if is done during smaller iterations. This is what the Iterative Model employs with testing being done after the design and development stage of each build. The Spiral Model is different as it is both iterative and systematic in nature. Here, testing is implemented during the third quadrant known as the construct or build stage.

With the Agile Model being Agile in nature, testing is implemented frequently. This minimizes the risk of any major failures in the future. In the RAD model the overall testing time is reduced as the prototypes are independently tested during every iteration. However, the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues. The difference in the Software Prototyping Model is that testing takes place during the development of the product but there is no dedicated testing phase or stage.

# Customer Involvement

The involvement of the customer is necessary to achieve not only a quality product, but a product that meets all the requirements of the customer at hand. Therefore, the customer’s input should be retrieved at as much as possible in a development lifecycle.

In the Agile Model, the customer has a great amount of customer. Customer interaction is vital in the Agile methodology. This varies greatly in the Waterfall Model, where the customer is only involved at the start of the process during the requirement and gathering stage. They are also involved near the end at the deployment and maintenance stages. The customer is not heavily in this process. In the V-Model however the customer is only involved during the first phase, the business and requirement analysis stage. Even though the customer is involved during this one stage, detailed and comprehensive communication is had with the customer to understand their expectations and fully.

In the Big Bang Model, there is little to no customer involvement, due to it being sporadic and unpredictable in nature. This differs greatly from the Iterative Model whereby models are produced early which facilitates customer evaluation and feedback. The Software Prototyping Model builds on this as it facilitates the team to understand the customer requirements at an early stage of development. Due to the rapid creation of prototypes, the customer can evaluate and give input on each iteration of the product. Similarly, the RAD model focuses on the gathering of customer requirements through workshops or focus groups, testing of prototypes and many other ways, so that they can obtain as much information from the customer. The Spiral Model is like the V-Model in the way it interacts with the customer. Here the customer is involved in two quadrants, the identification, and the evaluation stage, where the customer explains their requirements and evaluates the work of the team respectfully.

# Nature of The Processes Steps

Depending on the task or project at hand, a certain procedure would be needed to fully achieve the required goal. Some may benefit more than others, and all have their advantages and disadvantages that separate them.

The first nature of process is the sequential pattern. This means that one step, stage or phase must be completed before you can proceed to the next stage of the process. This is a highly disciplined model. The advantages are that it is simple to follow, and it enhances managerial control. The disadvantages are that it can be very time consuming and if a change needs to be made later in the life cycle, then the product development can become difficult and expensive to bring. The processes that employ this style of pattern are the Waterfall and V-Model.

The next nature of process is the iterative pattern. This is where the process does not start with a full specification of requirements. Instead, development begins by specifying and implementing just a small part of the software, which is then to be reviewed to identify further requirements. The advantages of this are that potential defects are spotted and dealt with early, functional prototypes are developed early, progress can easily be measured, and it is easy to change project scope unlike the sequential model. The processes that make use of this style of pattern are the Iterative Model, Spiral Model, and Agile Model.

Another process is the prototyping pattern. This refers to the building of software application prototypes which displays the functionality of the product under development, but may not actually hold the exact logic of the original software. The processes that make use of this style of pattern are the Software Prototyping Model and the Agile Model.

RAD is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product. Whereas the Big Bang has no set process and is highly volatile.

Some models, make use of an amalgamation of different process. The Spiral Model combines the idea of both the iterative development with the systematic, sequential pattern. The Agile method is combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.

# Ability to Handle Change

During the development of a product, there is a myriad of possibilities and reasons for. This i a natural occurrence when developing a product. Some products may be more at risk to change than others. Therefore, it is vital for the team to use a development cycle that will be best suited for the product at hand.

The Waterfall Model and the V-Model should not be used where the requirements are at a moderate to a high risk of changing. This is due to them being very sequential and disciplined in nature. In the Iterative Model, the cost of change is lesser, but it is not very suitable for changing requirements due to how the steps are taken as I had discussed before. The Spiral Model is more applicable to change than the Iterative Model as it incorporates aspects of the Agile methodologies which favour change. The Agile method itself has a great ability to handle change due to the method favouring customer involvement throughout and the process itself is not strict in nature. The RAD model is like the Agile Model in this regard, as its priorities the rapid creation for prototypes which the customer can review and give feedback on. This makes it very good at handling change. The Software Prototyping is akin to the methods of the RAD as well, making the changing of requirements highly accommodatable

# Conclusion

In this essay I have compared the characteristics of the Waterfall Model, Iterative Model, Spiral Model, V-Model, Big Bang Model, RAD Model, and the Software Prototyping Model. I have shown how the testing integration, customer involvement, nature of the processes steps and the ability to handle change of each model must be considered before the selection of a SLDC.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DEVELOPMENT CYCLES COMPARISIONS SHEET** | | | | | | | **TEAM MEMBERS** | **Daniel V.  Raphael S. Christopher P.** |
|  | **WATERFALL MODEL** | **V-MODEL** | **ITERATIVE MODEL** | **SPIRAL MODEL** | **BIG BANG** | **AGILE** | **RAD** | **SOFTWARE PROTOTYPE** |
| **Category** | Plan-Driven | Plan-Driven | Plan-Driven | Plan-Driven | No Specific Process | Agile | Plan-Driven | Plan-Driven |
| **Cost** | The process is cheap to manage. | The process is cheap to manage but may become expensive due to testing conditions | More resources may be required but it is less costly to change the scope/requirements. | The process is cheap to manage but may become expensive due to time and length of the project at hand. | Very few resources required | Low cost and inexpensive | Moderately expensive | Moderately expensive |
| **Requirements** | The requirements are very well documented, clear, and fixed. | The requirements are very well documented, clear, and fixed. | Requirements of the complete system are clearly defined and understood. Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time. | Customer is not sure of their requirements which is usually the case. Requirements are complex and need evaluation to get clarity. | Requirements are not clearly defined and are prone to change | Requirements are clear but can be changed if needed | Requirements are clear but can be changed if needed | Requirements are clear but can be changed if needed |
| **Ability To Handle Change** | Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model. | Not suitable for the projects where requirements are at a moderate to high risk of changing. Once an application is in the testing stage, it is difficult to go back and change a functionality. | Although cost of change is lesser, but it is not very suitable for changing requirements. | Changing requirements can be accommodated. | Highly malleable to change | The process has a great ability to change due to the iterative nature of it and the customer involvement | Changing requirements can be accommodated | Changing requirements can be accommodated |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **WATERFALL MODEL** | **V-MODEL** | **ITERATIVE MODEL** | **SPIRAL MODEL** | **BIG BANG** | **AGILE** | **RAD** | **SOFTWARE PROTOTYPE** |
| **Project Length** | The project is short in length | The project is short in length | End of project may not be known which is a risk. | Usually medium to long in length. | Very short in length | Due to its ability to be used on many projects, the length depends on the project at hand | Should be used where the requirements change during the project and working prototypes are to be presented to customer in small iterations of 2-3 months. | The length depends on the project at hand |
| **Testing Integration** | Testing is implemented/integrated after the implementation stage. Here all the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures. | The V-Model is an extension of the Waterfall Model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. | Testing and debugging during smaller iteration is easy. It is implemented after the design and development stage of each build. | During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available. | Testing may be implemented depending on the user of the project, but it is normally not implemented. | The product is tested very frequently, through the release iterations, minimizing the risk of any major failures in future. | The overall testing time is reduced in the RAD model as the prototypes are independently tested during every iteration. However, the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues. | Testing takes place during development but there is no dedicated testing phase or stage |
| **Risk Analysis** | This process does not factor into its cycle risk analysis, therefore there are high amounts of risk and uncertainty | This process does not factor into its cycle risk analysis, therefore there are high amounts of risk and uncertainty | Easier to manage risk. High risk part is done first. | Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management. | There is no risk analysis so there is a very high risk and uncertainty | There is low risk if planned correctly however there is risk of sustainability, maintainability, and extensibility. | There is low risk if planned correctly however there is risk of sustainability, maintainability, and extensibility. | Risk of insufficient requirement analysis owing to too much dependency on the prototype. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **WATERFALL MODEL** | **V-MODEL** | **ITERATIVE MODEL** | **SPIRAL MODEL** | **BIG BANG** | **AGILE** | **RAD** | **SOFTWARE PROTOTYPE** |
| **Project Complexity** | Simple and easy to understand and use | Simple and easy to understand and use. | Management complexity is more. | Complex in nature | Very simple to understand | The project is not complex but can get out of control is there is no management. An overall plan, an Agile leader and Agile PM practice is a must without which it will not work. | Requires highly skilled developers/designers. | Requires competent developers/designers. |
| **Customer Involvement** | The customer is only involved at the start of the process, Requirement Gathering Stage, and nearing the end of the process, Deployment and Maintenance stages. The customer is not heavily involved in this process | The customer is only involved during the first phase, Business Requirement Analysis Stage. This is where the product requirements are understood from the customer’s perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. | During the life cycle, software is produced early which facilitates customer evaluation and feedback. | The customer is involved in two stages. **Identification** Requirements are gathered from the customers and the objectives are identified. **Evaluation** Requirements are gathered from the customers and the objectives are identified. | There is little to no customer involvement | As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements | Rapid Application Development focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery. | Software prototyping is becoming very popular as a software development model, as it enables to understand customer requirements at an early stage of development. It helps get valuable feedback from the customer and helps software designers and developers understand about what exactly is expected from the product under development. |
| **Clarity of Product Definition** | Product definition is stable. | Product definition is stable. | Definition is clear and to open to change | Definition is clear and to open to change | Very unclear and very prone to change | The product definition is clear but is liable to change due to new customer inputs or new findings from the team | The product definition is clear but is liable to change due to new customer inputs or new findings from the team | The product definition is clear but is liable to change due to new customer inputs or new findings from the team |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **WATERFALL MODEL** | **V-MODEL** | **ITERATIVE MODEL** | **SPIRAL MODEL** | **BIG BANG** | **AGILE** | **RAD** | **SOFTWARE PROTOTYPE** |
| **Attitude to Prototyping** | There is no prototyping In this process. No working software is produced until late during the life cycle. | There is no prototyping In this process. No working software is produced until late during the life cycle. | Some working functionality can be developed quickly and early in the life cycle. | Allows extensive use of prototypes. Users see the system early. | No prototyping takes place | Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation. However, there is no set prototyping stage | Demo working products are shown at the end of each prototype | Since a working model of the system is displayed, the users get a better understanding of the system being developed. |
| **Nature of The Processes Steps** | The process follows a sequential pattern, one step after the other. One stage must be completed before you can move to the next stage of the process | This is a highly disciplined model and Phases are completed one at a time. | An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model. | The Spiral Model combines the idea of iterative development with the systematic, controlled aspects of the Waterfall Model. It l is a combination of iterative development process model and sequential linear development model. It allows incremental releases of the product or incremental refinement through each iteration around the spiral split into 4 quadrants. | There is no process per say, the nature depends on the task at hand and the process structure is determined by the user or team. | A combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. | The RAD (Rapid Application Development) model is based on prototyping and iterative development with no specific planning involved. The process of writing the software itself involves the planning required for developing the product. | The Software Prototyping refers to building software application prototypes which displays the functionality of the product under development, but may not actually hold the exact logic of the original software. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **WATERFALL MODEL** | **V-MODEL** | **ITERATIVE MODEL** | **SPIRAL MODEL** | **BIG BANG** | **AGILE** | **RAD** | **SOFTWARE PROTOTYPE** |
| **Progress Analysis** | It is difficult to measure progress within stages. | It is difficult to measure progress within stages. | Progress can be easily read. | The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase. | Progress cannot be measured as there is no clear staging or phase to this process | Progress can be measured after each iteration | Progress can be measured after each iteration prototype | Progress can be measured after each iteration prototype |
| **Integration** | Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early. | Integration is prepared during the architectural design stage. This is where the tests for integration are document and designed. Then during the integration stage, the tests are performed to test the coexistence and communication of the internal modules within the system | Integration is easy to manage due to the incremental nature of the process | Integration is easy to manage due to the incremental nature of the process | Integration happens all at once with no clear planning or testing. It is prone to failure. | Integration will take place at the end of each iteration or in its own separate sprint or iteration | Integration from very beginning solves a lot of integration issues. | Integration will take place at the end of each iteration or in its own separate sprint or iteration |
| **Documentation** | Process and results are well documented | Process and results are well documented | Process and results are well documented. Issues, challenges, and risks identified from each increment can be utilized/applied to the next increment. | Process and results are well documented | No documentation | Process is heavily documented with backlogs and references that the suer can harken back on | Process is documented well | Process is documented well |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **WATERFALL MODEL** | **V-MODEL** | **ITERATIVE MODEL** | **SPIRAL MODEL** | **BIG BANG** | **AGILE** | **RAD** | **SOFTWARE PROTOTYPE** |
| **Project Size** | Works well for smaller projects where requirements are very well understood. | Works well for smaller projects where requirements are very well understood. | Better suited for large and mission-critical projects. | Better suited for medium to high-risk projects. | Recommended for products that are small in scale such as class projects | Cam be used for any project scope | RAD model can be applied successfully to the projects in which clear modularization is possible. If the project cannot be broken into modules, RAD may fail. | Favours projects larger in scope |
| **Ability To Manage** | Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process | Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process | More management attention is required. | Long-term project commitment because of potential changes to economic priorities as the requirements change with time. | Easy to manage, but can spiral out of control if the scope was larger than expected | The process is easy to manage | Management complexity is more. | Users may get confused in the prototypes and actual systems |