Contents

Comparing existing model and product lifecycle models. 1

Project lifecycle plan.. 2

1. Planning. 2

2.1 Analysis. 2

2.2 Design.. 2

2.3 Coding. 3

2.4 Testing. 4

2.5 Maintenance. 5

2.6 Deployment 5

Conclusion. 5

# **Comparing existing model and product lifecycle models**

Software development life cycle is a methodology that defines the processes which needs to be undertaken for developing a high – quality software that meets the requirements of everyone involved in the project (Misra & Singh 2015). There are many lifecycle models which can be used for effective development of software by following certain procedures and steps. Some of them are code-and-fix, prototyping, waterfall, incremental, agile, spiral, and so on. As every lifecycle model has its own pros and cons, depending on the requirement of the project, any model can be used (Shen 2020). For the game development project, there are many appropriate models like waterfall, agile and incremental which can be used.

Comparison of product lifecycle models

|  |  |  |
| --- | --- | --- |
| Model | Advantage | Disadvantage |
| Waterfall | - Good for small projects whose requirements are well understood.- No functioning software is produced until the end of lifecycle | - Not effective for big projects as requirements about the projects must be collected before starting the project. |
| Incremental | - More flexible and easier to test and debug.- Each increment provides a functioning product. | - Phases do not overlap each other so there is no going back once each phase is complete.  |
| Agile | - Requires less upfront planning and design.- Frequent delivery of working software.- Changes in the requirements can be well adjusted at any time during the development cycle.- Frequent interaction between the users, developers, and stakeholders. | It mostly depends on interaction with the customers and if the customers are unsure about the requirements then the project might head in completely the wrong direction. |

Table 1.SDLC comparison (Shen 2020)

We followed certain criteria to choose the best software development lifecycle model which is by understanding the project requirements, scope, quality, and resources available.

In the chopper game project, the requirements can be changed in any stage. While using waterfall and incremental models the requirements must be well understood before starting the project, so these models were not found that effective for our project. As we need to regularly interact with the users to collect the feedback and use it in the next version, agile model is most suitable as it produces a functioning software once each iteration is complete.

# **Project lifecycle plan**

## **1. Planning**

Before starting any project, proper planning is required for the successful execution of the project. The project’s tasks and resource requirements are identified, along with the strategy for producing them. A detailed work plan is prepared at first which is already discussed in the first project management plan report including the project milestone list, work breakdown structure, change management, communication management, risk management and so on. The project manager is responsible for all the arrangements with proper planning and utilization of available resources. The main aim of preparing this document is to make sure that there will not be any problems during the development lifecycle of the project. If any problems arise then it can be mitigated by following the procedures that were written in the planning document. Once the plan is ready, the appropriate system development life cycle is chosen, and the game development process begins with analysis, design, coding, testing, maintenance and finally deployment.

## **2.1 Analysis**

Analysis phase is the phase where the system development life cycle begins. In this phase, all the functional and non- functional requirements about the project are defined and documented. Since, the model is agile, there will be very less up-front planning and analysis will be done in each iteration. In the first iteration, all the group members and project manager will discuss the game, evaluate it, and identify the necessary changes that can be made to make the game more interesting. Once the requirements are defined and well understood, other phases are executed and once the testing of iteration 1 is done which is version 1.0 in case of our game, the feedback from the users are collected. All the positive and negative feedbacks by users are closely considered and the new requirements are identified from the feedback. For example: There will be project members and more than six users for the evaluation of the game and the feedback might be about the background colour, chopper size or gameplay and so on. Depending on the feedback, we as a group identify which requirement is highlighted by the majority of users and discuss whether it would be suitable and can be implemented on the game considering the resources we have. If yes, it is used in iteration 2 to develop the second version of the game otherwise kept in consideration for the future versions of the game. This process continues on until all the user requirements are met and the game is ready.

## **2.2 Design**

As the version increases, the level of design also increases, we need to look at all the necessary specifications, features and operations that are required to satisfy the functional and non-functional requirements of the game. As the version 0.1 is already provided to us with very basic features and game play, we need to further design the game in such a way to be able to deliver each requirement to the users. At first the technical details of the design are discussed with the group members such as risks, technologies to be used, capability of the team, project constraints, and time. As chopper game is web-based game, we need to focus on the display size of the game so that it fits in different platforms such as mobile, table, pc and so on. There are many chopper games available online and users might find our game similar to others and might not prefer to play so we need to make it different. For example, by changing the background, adding enemies instead of walls, sound effects after crash, level ups and many more. As the model is agile, changes in design can be done in iterations so that the feedback from the users can be collected and used in the next iteration. For example, if the majority of users did not like the background colour and chopper size in version 1.0 then it is analysed in the analysis phase of next iteration and the discussion is done within the group for potential design for the next version. The design is usually kept in the Design Specification Document (DSD). The design must be chosen in such a way that it runs on all the platforms and all updates of the game design must be updated on the document. A system security system must be built to make sure the game cannot be hacked or can there be any cheats for the game. A backup plan must be created to ensure the game plans and design are recovered in case there is any system affected by the hackers.

## **2.3 Coding**

After making a framework for how the project should run, dividing the activities in accordance with time and resources available, the construction of the game starts. The purpose of this phase is to prepare the application program code as per the specification required by the program. Programmers of the project team are allocated with specific time to conduct the portions of the game. Codes for introduction, control buttons and restarting if failed are included. If conditions are included so that if the player does not follow the rules, the game has to re-started. First stage of this gives out version 1.0 of the application. The coding is done in such a way that could adapt new changes easily as per the customer feedback. After the development of the code, the presentation of the application with clients is carried out through which new improvements are made in the coding which releases version 1.1 for the application. The game has 4 versions that is a result of continuous testing and amendments made. An overview document will show the detail of the steps of coding with feedback, improvement, and the new versions. Hence, development of code and amending them according to client needs is done during this process.



Figure 1. Coding process (Nystrom 2014)

(Source:<https://gameprogrammingpatterns.com/game-loop.html>)

## 2.4 Testing Phase

Software testing is a mechanism of evaluating if the functionalities of an application are working accordingly or it is defect-free. Before resuming with a test plan, it is crucial to have a comprehensive testing plan, which can act as a guiding document to ensure that all modules of the system have been tested. It is worth noting that testing is paramount, as software bugs can be expensive and dangerous. For the chopper game project under consideration, the application will be subjected to different types of testing, during, and after the development. Such software testing to be done will include unit, integration, acceptance, functionality, or end to end and finally, exploratory testing. According to Pearson (2015), unit testing is essential because it determines whether the functions or the procedures of software are free from any bugs. Performing unit testing in the chopper game project will be done every time the status of the code has changed in different software modules. The idea of introducing integration testing is to enable the testing team to have an opportunity of combining all the units within the program and testing them upon integration. With this, it will ensure that the integration software will be free from any bug. Studies reveal that performing integration can be beneficial since it ascertains how efficient the units of a software application can run together. In the acceptance testing, the rationale behind conducting this testing is to determine whether the system is ready to be released to the customers. Functional or end to end testing is essential in stimulating a whole level of user-level experience. Lastly, expiratory testing will be necessary for this project as it will engage different users to interact with the software.

## In ensuring that all the tests mentioned in this project have been performed accordingly, the use of an agile test plan will be crucial. Notably, an agile test plan is a crucial document which gives the quality assurance team, the ability to test the software. Ideally, an agile test plan comprises various parts, which include document scope, feature description, the functionalities to be tested, type or level, testing process, and risk and dependencies (Pavlov, 2019). The feature description is crucial in clarifying how the newly developed features work and how they can be tested based on the agile test plan. The testing process in the agile test plan, for the software project under consideration, must clearly state the kind of testing to be done. Other forms of testing in agile include functional testing, security testing, user interface testing, cross browsing testing, regression testing, among others. In understanding the importance of testing the software, it is good to note that it emphasizes quality at every step. The second importance is that it empowers the software developers to prove the quality of the software or features offered by that software. Therefore, it can be concluded that the testing phase in the software development lifecycle is essential in ensuring that there is a high-quality software product. Testing

## **2.5 Maintenance**

Every software requires regular maintenance to know that the software is performing well and meets the requirements that were established in the analysis phase. The main aim of this phase is to fix the issues and bugs in the game that is being developed. The potential users and project members are responsible for finding the issues and mitigate them in this phase. This phase is not about updating the game and adding the new features as per the new requirements. It is only about fixing the current issues/bugs in the game. For example, the chopper in the game is supposed to be inside the boundary that was declared while developing the game. If the chopper collides with the boundary then it should crash. The issue might be that the chopper flies beyond the boundary and the score keeps updating and it does not crash. In this phase this issue is targeted and the project team works on it to fix it first by changing the code and moving on to the development of the next version. So, the main advantage of having this phase is that we can fix this issue without going back to the analysis, design, and coding phase. Simply, the project members can update the code and fix the bug/issue.

## **2.6 Deployment**

In this phase of software development lifecycle, the final version of the game is released. Once all the functional and non-functional requirements set by the users are met and there are not any bugs to be fixed in a game, the game is ready to be played. This phase is only performed once in the life cycle at the end.

# **Conclusion**

In conclusion, the quality of a game depends on the process that was followed for effective planning and choosing the best development cycle as per the requirements. An effective collaboration within the team is most important along with the proper utilization of available resources. If all the things are organized and closely monitored, then the project follows the right path and the goal to make a user friendly game can be achieved with minimum risk.

References

Shen, H 2020, ‘Week 3 Software Lifecycle Planning’, lecture slides, ITEC312 Information technology project part A, Australian Catholic University, viewed 11 May 2020,

https://leo.acu.edu.au/pluginfile.php/3937361/mod\_resource/content/3/L03%20Lifecycle%20Planning.pdf

Misra, S.C. & Singh, V., 2015. ‘Conceptualizing open agile software development life cycle (OASDLC) model’, *International Journal of Quality & Reliability Management*, 32(3), pp.214–235. Viewed 12 May 2020,

https://acu-edu-primo.hosted.exlibrisgroup.com/permalink/f/sb9f4f/TN\_emerald\_s10.1108/IJQRM-08-2013-0127

Nystrom, R. (2014). *Game Loop-Sequencing Patterns: Game Programming Patterns*, image, Gameprogrammingpatterns.com, viewed 14 May 2020,https://gameprogrammingpatterns.com/game-loop.html.

 Pavlov, M. (2019). What is Agile Test Plan. Retrieved from https://www.spiria.com/en/blog/quality-assurance/what-is-an-agile-test-plan/

Pearson, L. (2015). The Four Levels of Software Testing. Retrieved from https://www.seguetech.com/the-four-levels-of-software-testing/

Appendix

Demonstration of both collaborative work and individual contribution

Team Members

1. Aarti Acharya

2. Roshan Khadka

3. Rosy Shrestha

4. Sandip Pala

Most of the group discussion was done in Facebook messenger where all the parts for the Software development life cycle was discussed. We created an online word document so that everyone can work on it together. We had frequent group meetings to track our progress and discuss different issues faced while writing the system development life cycle plan. The whole project was divided into 4 parts and each member had to finish it. All the discussion about how to write the phases was done so that the parts done by individual member would connect with one another and make sense.

Contribution by each member

|  |  |  |
| --- | --- | --- |
| Name | Section | Percentage |
| Aarti Acharya | a. Planningb. Codingc. Deployment | 25% |
| Roshan Khadka | a. SDLC comparisonb. Analysisc. Editing document | 25% |
| Rosy Shrestha | a. Designb. Conclusion | 25% |
| Sandip Pala | a. Testingb. Maintenancec. Editing document | 25% |

Screenshot: Group meeting

 In this meeting we discussed the different life cycle models that would be best fit for our game development project and choose the best one.

Outcome of this meeting:

1. Agile model to be used for game development.

2. Phases to be included: Planning, Analysis, Design, Coding, Testing, Maintenance, and Deployment.

3. Section that each member would be contributing which is listed in table above.

Group Discussion:

 Peer Evaluation

This section includes the evaluation done by each member for the other three members.