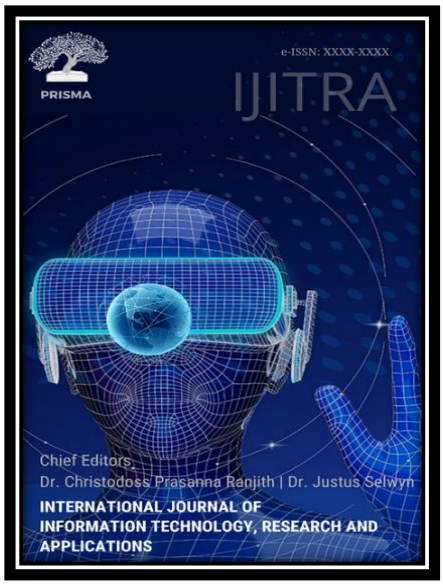


International Journal of Information Technology, Research and Applications (IJITRA)

Yehezkiel Yosua Parlindungan, Yani Widyani, M. Zuhri Catur Candra, (2023). Applying Scrum in A Game Development Life Cycle For Small Scale Game Project, 2(4), 24-37.

ISSN: 2583 5343

DOI: 10.59461/ijitra.v2i4.73

The online version of this article can be found at:
https://www.ijitra.com/index.php/ijitra/issue/archive

Published by:
PRISMA Publications

IJITRA is an Open Access publication. It may be read, copied, and distributed free of charge according to the conditions of the Creative Commons Attribution 4.0 International license.

International Journal of Information Technology, Research and Applications (IJITRA) is a journal that publishes articles which contribute new theoretical results in all the areas of Computer Science, Communication Network and Information Technology. Research paper and articles on Big Data, Machine Learning, IOT, Blockchain, Network Security, Optical Integrated Circuits, and Artificial Intelligence are in prime position.

PRISMA
PUBLICATIONS

https://www.prismapublications.com/

Applying Scrum in A Game Development Life Cycle For Small Scale Game Project

Yehezkiel Yosua Parlindungan, Yani Widyani, M. Zuhri Catur Candra

School of Electrical Engineering and Informatics, Bandung Institute of Technology,

West Java, Indonesia

Article Info

Article history:

Received October 23, 2023
Accepted November 25, 2023
Published December 10, 2023

Keywords:

Insurance Products
Buying Decision
Sales Services
Awareness
Preference

ABSTRACT

This research aims to develop a GDLC game development life cycle that focuses on the core stages of software development process. This GDLC proposal consists of 4 stages, namely Pre-Production, Production, Testing, and Release. This proposed GDLC is expected to provide adaptive but consistent guidance for game project design. These conditions are suitable for small-scale game development teams, with small project scales, and short work periods (under 1 year).

Pre-Production functions as the initial stage for creating the basis game design for the project linearly. Production, which is carried out using the Scrum approach, is the stage of video game development based on that previously made design. The Testing phase ensures the quality of the game through overall game testing. Lastly, the Release phase includes completing documentation for post-release planning.

This GDLC was designed by summarizing the core stages of various existing GDLCs and SDLCs related to multimedia software development. Scrum is implemented per stage, with iteration loops starting and finishing within that stage. However, in general, this GDLC is linear. This GDLC is validated with a case study of an actual video game development to ensire its efficiency in helping game developers.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Yehezkiel Yosua Parlindungan
School of Electrical Engineering and Informatics
Bandung Institute of Technology
West Java
Indonesia
Email: yehezkiel.parlindungan@gmail.com

1. INTRODUCTION

In the dynamic and rapidly evolving realm of video game development, the need for streamlined and efficient methodologies has become increasingly paramount. The success of a game hinges not only on its creative brilliance and engaging gameplay but also on the structured and systematic approach taken during its development lifecycle. This paper introduces the "Short Game Development Life Cycle" (Short GDLC), a novel methodology tailored to be adaptable for the challenges faced by contemporary game developers in creating high-quality games within tight schedules, while also giving the developers a consistent format.

Semi Agile GDLC is designed to encapsulate the essential phases of game development while adhering to Agile principles, promoting adaptability, collaboration, and iterative improvement. This framework recognizes the need to balance creative exploration with structured planning, ensuring that the development process is both efficient and effective.

In this paper, we will delve into the intricacies of the Short GDLC, breaking down its phases and elucidating how it aligns with industry best practices. From the conceptualization of game ideas in the Pre-Production phase to the iterative development cycles of the Production phase, and the comprehensive testing and thoughtful release strategies, the Short GDLC offers a comprehensive approach to game creation.

We will discuss how the Short GDLC addresses the unique challenges game developers face, including managing timelines, adapting to design iterations, and maintaining a commitment to producing exceptional quality games. By embracing this methodology, developers can navigate the complex landscape of game development more effectively and efficiently, ultimately delivering games that captivate players and resonate within the competitive gaming industry.

In the subsequent sections, we will delve into the specific phases of the Short GDLC, examining each step's methodologies, benefits, and potential for fostering innovation. Furthermore, we will explore how this approach complements existing game development paradigms and positions itself as a versatile framework for a wide range of game projects.

2. METHOD

To create this GDLC, we must first analize the essence of what is GDLC and what kind of stages it generally has, to create an initial untested model for the base of this research. Verious GDLCs (and some general Software Development Lifecycles) are gathered, analyzed, and compared to each other, to determine the core stages needed for a GDLC to exist. Then we determine which stages neede to be consistent and which ones needed to be adaptable to changes. Scrum is then implemented to those stages that needed to be adaptable.

The next step is to validate and fine tune the initial untested GDLC model. It is validated with a case study of actual game development using the said initial model. The differences between stages that are dictated by the initial model and the reality of the game development workflow are noted and used for fine tuning this GDLC. Changes according to the notes resulted in a final GDLC model that hopefully can accommodate developers to make a game.

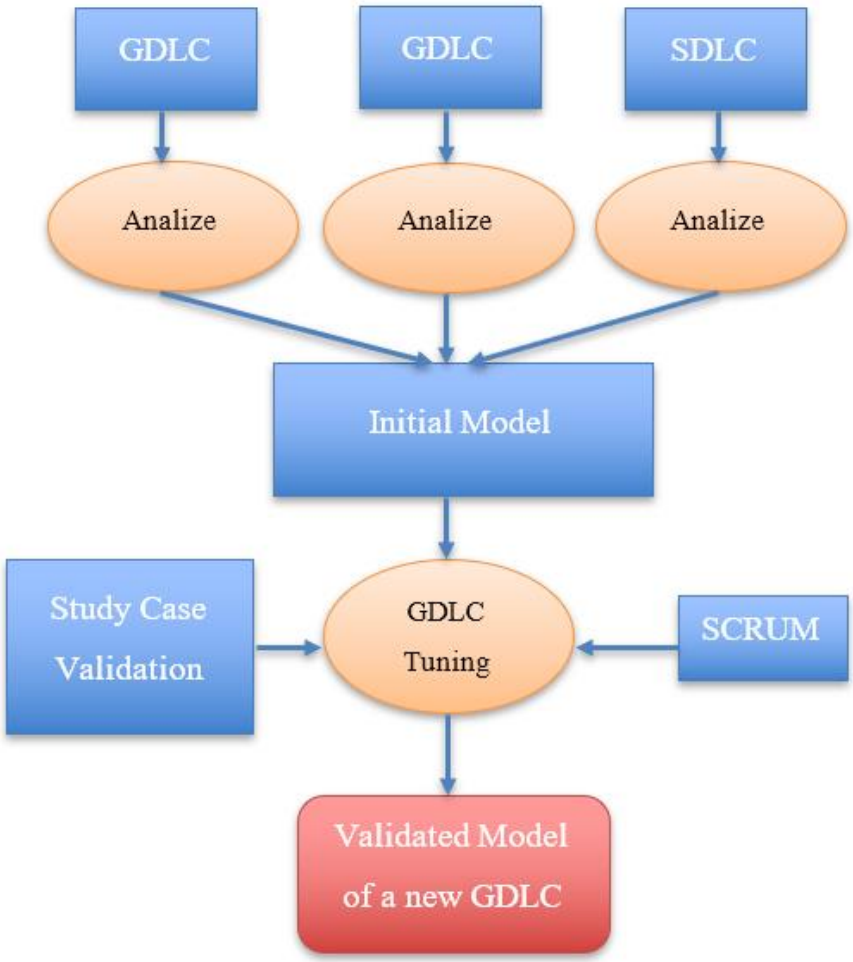


Figure 1. Overview of Prediction Modeling

2.1. Definition of Game Development Life Cycle

Video games have multidisciplinary characteristics. The process of making it include a team with different function from a different study, such as the study of story design, informatics, audio design, visual design, and other kinds of it [1]. In other words, the development process of a video game cannot be done with just one mindset.

The inspiration for constructing a GDLC comes from many things. One can get it from another existing GDLC, and another can get it from a methodology that is not set for game development. There is not a single one (software package management, that is the game engine and its libraries) that can fully track and provide the perfect development process of a game, thus it's hard to recommend only one of them [2].

There are 4 core roles or divisions in a game development team, which are production, design, art, and programming [3]. The whole team is led by the leader of the production division, which is the producer. Each division can be divided into other specific sub-divisions according to the development needs. For example, the designer division can be divided into level designer, story writer, mechanic designer, and other things. With each role or division having its own sets of sub-roles or sub-divisions, a team needs to have those 4 main divisions, or at least have team members that can play the roles above.

According to Heather Maxwell Chandler in his book, "The Game Production Toolbox", there is a need for milestones set for making a video game. These milestones can have a broad and labile definition. An "alpha" stage for one development team can be defined as a "beta" stage for the other team. But the purpose of these milestones is the same, that is to be the stopping point for decision making. This decision can be expanded or limited to the development of the game, erase, or add a feature, or change the work schedule of the team.

1. Prototype
   A product that only focuses on the core mechanics, in the simplest form. The prototype's purpose is to give the general direction of the development, and test which mechanics work or not in the case of gameplay.

2. Vertical Slice / Pre-Alpha
   Is the more complete version of a prototype. Vertical slice is a "finished" product assembled from the development, that is not fully tested yet but already has the visual and core aspects of the game. Placeholders that are used in the prototype have been replaced by proper assets.

3. Alpha
   Is the first stage of a complete product. The main mechanic of the game has been implemented, and the main features that are targeted for the final release version of the game have been reached. The existence of an alpha-stage product means the development process has reached its midpoint.

4. Beta
   Beta is a more stable version of alpha. In this milestone, the development goal shifted from "creating features and bug fixing" to "polishing an already existing feature".

5. MVP
   Or (Minimal Viable Product) is the technically finished version of a game product. This version can be shipped as a proper video game product on its own, but it can be improved more.

6. Release
   Is the complete version of MVP, and the game product is ready to be released to the market.

7. Post Release
   Is the milestone after the game has been released. It's optional, but these days many game developers tend to do it.

2.2. Analysis On Existing GDLCs

There are many various steps from various methodologies that are presented in a variety of GDLCs. Some GDLCs give solutions to specific problems and conditions. Some extend their' uses to all kinds of scenarios, with various steps and milestones. But at the heart of its core, all GDLC lies a concise yet

comprehensive framework, elegantly capturing the key stages that underpin successful game creation, and that focuses on the software engineering of a game.

2.2.1. Steps in Various GDLCs

Among those various GDLCs that have been mentioned before are Arnold Hendrick's GDLC, Doopler's Interactive, Ramadan & Widyani's GDLC, and Heater Chandler's GDLC. For a more valid reason, those four GDLCs will also be compared to two general SDLCs (Software Development Life Cycle) that are known to be popular for game development.

List of GDLCs and general SDLCs That Are Suited for Game Development

Arnold Hendrick	Heather Chandler	R&W	Doopler Int.	MDLC	Scrum
Prototype	Pre-Production	Initiation	Design	Concept	Scope
Pre-Production	Production	Pre-Production	Develop / Re-Develop	Design	Product Backlog
Production	Testing	Production	Evaluate	Material Collecting	Design
Beta	Post Production	Testing	Test	Assembly	Sprint Planning
Live		Beta	Release Review	Testing	Sprint Backlog
		Release	Release	Maintenance	Sprint Automation
					Sprint Execution
					Sprint review
					Sprint Retrospective
					Usable Software

Cell Color Explanation:	
Yellow	Initiation Phase
Blue	Work process
Green	Testing
Red	Final Phase

Working Process:	
Double border	A work loop (sprint)
Double border	inside of a loop
Bordered Cell	A work loop iteration

Figure 2. Tabel of Methodologies Comparison

Each step in each lifecycle is described and then categorized into 4 phases or stages, namely the "**initiation**" stage (yellow), "**work process**" (blue), "**testing**" (green), and the "**final**" stage (red). This is done to facilitate concluding the elaboration of various steps. Some steps that are iterative or agile are marked with a border on its cell. Some important points that can be drawn from the comparison above are:

1. Every lifecycle has its own version of "Pre-Production", with some splitting it into many steps, while others just combine it into one phase of initiation.
2. There's always a "Testing" phase before its release.
3. The general phases are in line with the typical software development life cycle.

Figure 3. Typical software development life cycle.

2.2.2. GDLC's Core Step

Each step in each lifecycle has been broken down and analyzed. The stages are now combined based on the similarity of their meaning and regrouped into one "general" set of steps, a core of steps that every GDLC is based on. Those steps are:

1. Pre-Production
   The initial stages of game-making. It consists of creating a basic concept and design, creating a work plan, and ending with a prototype.

2. Production
   The stage consisted of the actual development work. It is here where the concept and design are translated into assets and features. This phase ends with a complete but untested game.

3. Testing
   The testing, in this case, is to test the already-made game. It consists of playing the game from the beginning to the end, noting everything that needs to be improved, and fixing those issues.

4. Release
   It's the final phase that consists of document completion and the publication of the already-tested game.

2.2.3. Applying the Scrum Approach

Scrum is one of the most popular approaches in agile methodology. It consists of 3 internal roles that work in sprits to ensure the production is consistent but adaptive in many unforeseeable circumstances. The role consists of the product owner, the scrum master, and the team member who works for the development [4]. The rules are made into many points:

1. The team must have qualities of collective commitment and a self-organizing mindset.
2. The whole development is based on a **product backlog** that is made by the product owner.
3. The development is being done in sprints. **Sprint** is a set of iterative work in a set of time (usually one or two weeks).
4. Daily meetings are held each day in a sprint to discuss yesterday's actual progress, today's intended progress, and the potential obstacles that may come. That daily meeting is called **daily scrum**.
5. Scrum master ensures the flow of a sprint. He/she will hold meetings with the team at the beginning and the end of a sprint. The meeting at the beginning of a sprint is held to discuss which tasks are planned to be worked on in the current sprint. The first meeting at the end is called **sprint review** and is held to discuss which task is being done, changed, or dropped. The second one is called **sprint retrospective** and is held to review the team's performance in that sprint.

The rules of scrum above are compatible with the flow and rules of a game development team. Those rules can be combined along with the roles. From the analysis of it can be extracted a simple conclusion that:

1. The roles of scrum can be implemented in the role of a game development team.
   a. The producer (head of the production team) alongside external clients (if any) can be assigned the role of product owner. It is stated so because, in game development, the producer is the one who has the initial idea of the game and is the one who lists the tasks needed to be done for the game to be completed.
   b. For the role of scrum master, it must be someone who comes from the production division that someone can be the producer itself, but better if it's someone else. If it falls to someone other than the producer, then that someone can be marked as a **product manager**.
   c. The other divisions will be assigned to the role of scrum team members since they are the ones who will do the actual development work.

2. Since it was stated above that the producer is the one who takes the role of a product owner, then he or she will be the one who creates the **backlog** for the project. It can be done by:
   a. Creating an initial GDD (Game Design Document) for core concept and design.
   b. List the features needed for that concept.
   c. List the tasks needed to make those features in a backlog document.

3. The tasks can be done by the team members in sprints. The sprints are managed by someone in the production team (the product manager) as a sprint master, and they will follow the scrum rules that are listed above.

3. RESULTS AND DISCUSSION

By the analysis of the various theories above, the first version of a GDLC can be created. It consists of four stages that will be done in a semi-agile approach.

3.1. Initial Version of The Proposed GDLC

What it means as semi-agile is, that the whole four stages will be done linearly like a waterfall, but each stage has sub-stages that will be done in a scrum-like agile approach. The initial pre-validation version of the proposed GDLC consists of:

1. Pre-Production
   It's the first stage that focuses on creating a backlog and initial GDD. Initially, it has 2 proposed sub-stages, which are the **initiation** and **material collecting** sub-stages.
   a. In the initiation sub-stage, the production division and the design division will create the initial concept for the basis of the backlog and if possible, create a **prototype**.
   b. In the material collecting sub-stage, the team will create basic assets for the basis of the next production stage, creating a **vertical slice** build.

2. Production

The team then will keep expanding the features created in the previous stage, in a scrum-like agile manner, to make an **initial alpha** build. The team works on the tasks that are listed in the backlog. The flow of working a task is: create the feature, check, or test the quality of that feature (QA test), and if it's good then integrate it into the game. Those tasks and features are worked in sets of sprints, following the rules that are mentioned above.

3. Testing

It's the stage of validating the whole game. Not to be confused with the QA test of each feature, this stage consists of playing the whole build of the game (playtest), listing everything that is wrong or needs improvement, listing a set of tasks to fix those issues in a testing backlog, and then working those tasks in an agile manner like the ones in the previous stage. This stage can be done repeatedly as much as the team needs it, but is separated into 2 kinds:

a. Alpha testing

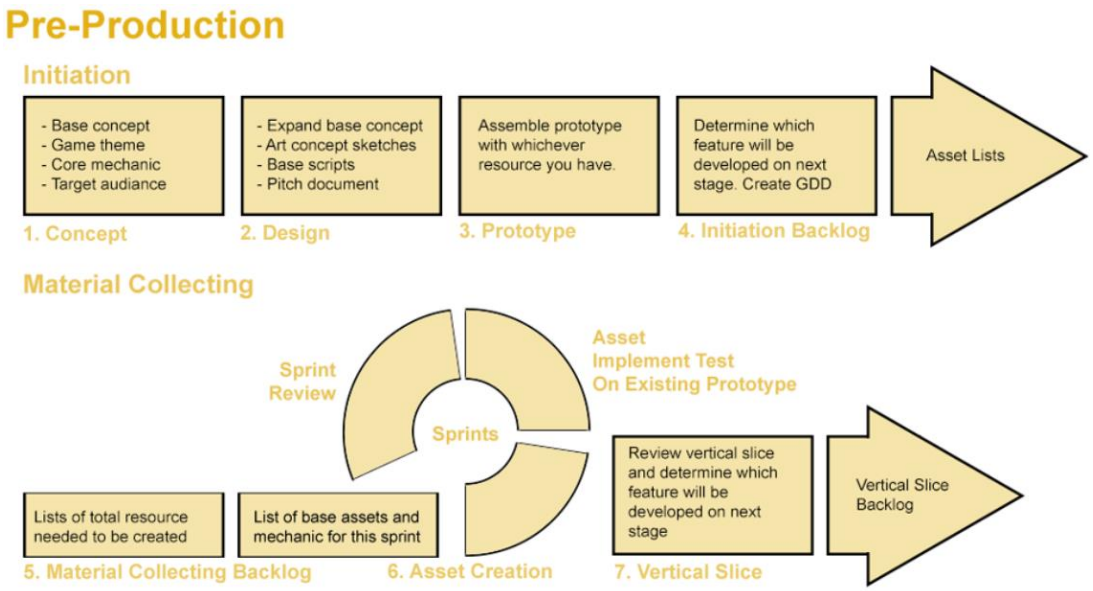
It's the kind of testing that focuses on fixing issues. Usually, the playtest is done internally by a QA division. But if the team has no QA division (because of the lack of workforce) then everyone in the team can test it, with a QA document made by the designer division. Each iteration of alpha testing will create a version of the **alpha** build, but the last iteration of it will create the **initial beta** build.

b. Beta testing

It's the kind of testing that focuses on improving the features that have already been made. The playtest can be done externally by potential players or internally by the team like alpha testing. Each iteration of beta testing will create a version of the beta build, but the last iteration of it will create an **MVP (minimal viable product)** build.

4. Release

It's the final stage that consists of marketing, hype building, and document finalization on each division. The design and production division must finish the GDD, the programmer must finish the TDD (Technical Design Document), and the artists must finish the ADD (Art Design Document) or Art Bible. If there is a need for improvement to the MVP build, it must be done in a quick and simple manner, to not break the already existing features. Only then the game can be released to the public as a **release** build.



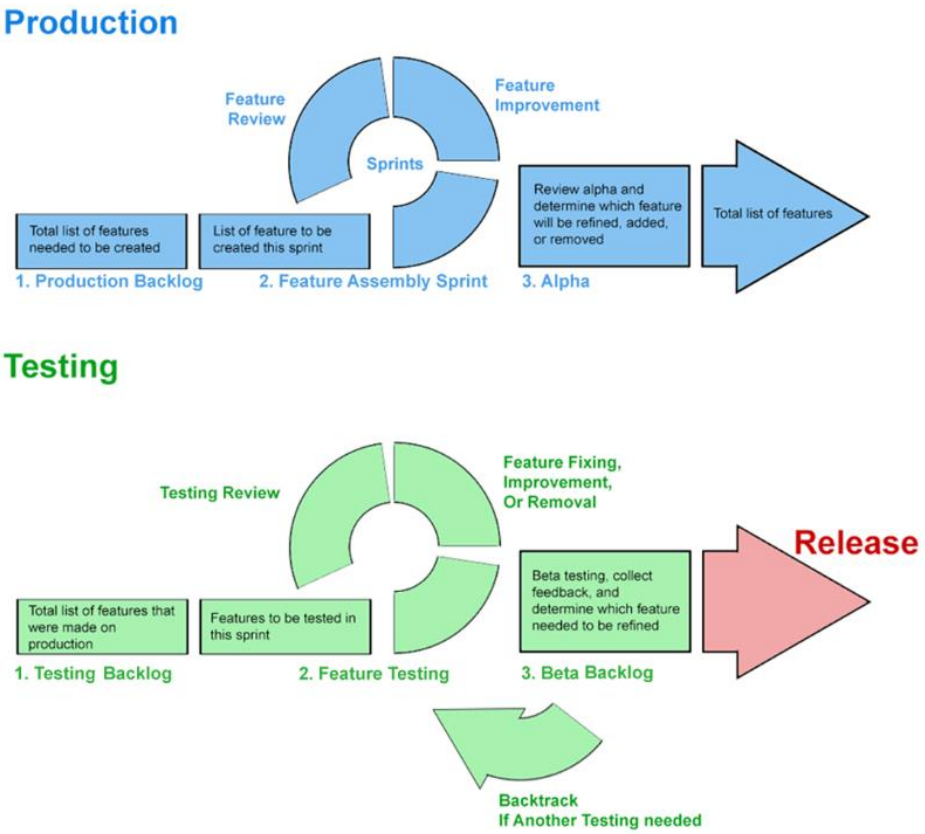


Figure 4. The Initial Pre-Validation Version of The Proposed GDLC.

3.2. Validation of The Proposed GDLC

The process of method tuning consists of comparing the core phases of GDLC with a study case of making a video game. The phases will be expanded and tuned based on what happened in the process of developing the game. The validation itself is based on Mary Shaw in her journal *"What Makes Good Research in Software Engineering"*, which states 3 points of software engineering, which are question, result, and validation [5]. She then stated that those 3 points can be described into many types. The research question, result, and validation for the making of this GDLC are listed in the table below.

Research	***Type***	***Explanation***
Question	Method for means of development	- How to make a video game - Which stages should be expanded or simplified? - QA testing in the testing phase
Result	Descriptive Model	- Explanation for each phase or stage
Validation	Experience	- Implementation of the core stages of a study case - Comparison of the phases with the reality of the development process

Figure 5. Research question, result, and validation.

For the comparison that is mentioned above, the study case is to create a serious game titled "Mindverse". It's a mixed reality-based game for creating mind maps for the Hololens 2 platform. Players can make orb nodes/balls and text panels, interact with them physically, and attach them all to create a 3D-projected mind map structure. This game development was done in collaboration with the Metaverse Lab of STEI (School of Electrical Engineering and Informatics) Department, which is a part of ITB (Bandung Institute of Technology). It was developed by one person who took the role of both producer and designer, two programmers, and one artist. Everyone except for one programmer (for testing purposes) works remotely.

3.2.1. Project Description

The initial plan to create this "Mindverse" serious game is to **lay out tasks and work on them sequentially.** There are 15 tasks initially created to make this serious game, which consists of:

A. Pre-Production
   1. Requirement and Analysis
   2. Technical Preparation
   3. Design UI/UX

B. Production
   4. Game Feature: Create and move orb nodes.
   5. Game Feature: Save the game.
   6. Game Feature: Voice Command.
   7. Game Feature: Image projection.
   8. Game Feature: Save file sharing.
   9. Game Feature: Attach and detach node.
   10. Game Feature: Wiki Fetch.
   11. Game Feature: Attach and detach node to a spatial space.
   12. Game Feature: Expand and Collapse of a mind map structure.
   13. Game Feature: Menu navigation and redo/undo.

C. Testing (Task 14)

D. Release
   15. Documentation completion and user manuals.

The project was intended to be worked **from November 2022 to March 2023**. The Pre-Production and Release stage is planned to be done linearly, while Production and Testing will be done using Scrum approach.

3.2.2. Executing The Initial Model of GDLC

There is many unplanned rescheduling, redesigning, and common hindrances in the time of development. This is not to say that this research in making a GDLC failed. Rather, the difference that is of the development process will significantly tune the stages of this GDLC more to suit its users. Overall, most changes come from the Production and Testing phase, and the actual time length spans **from November 2022 to the start of July 2023**.

1. Pre-Production

There is little to say in this stage, as the work schedule and the reality of the development process run similarly, other than half of task 4 is included in this stage. No rescheduling or changes happened, and the 4 tasks were done sequentially as planned, with the producer creating a backlog for the next stage.

Pre-Production

		November			
	Week ->		1	2	3
Task 1	Requirement & Analysis	■	■		
Task 2	Technical Preparation		■		
Task 3	Design UI/UX		■		
Task 4	Create & Move Orb (with Gesture)			■	■

Figure 6. Pre-Production Stage of The Study Case.

2. Production

Most changes are happening inside of this stage. The changes happen mainly because of mechanical redesign and changes in priorities. The features themselves weren't made sequentially as planned, and there are a lot of backtracks that happened because one change on one feature affects the other. The actual workflow in these stages is listed in the table below.

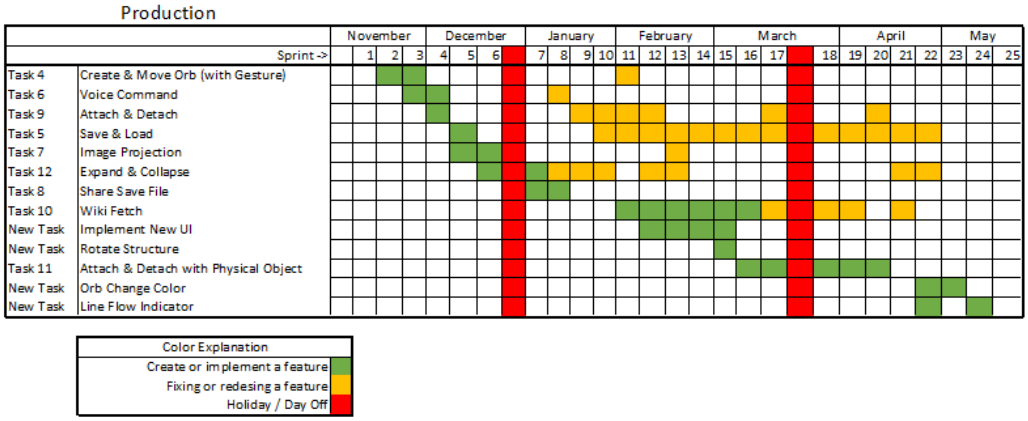


Figure 7. Production Stage of The Study Case.

As described above, many tasks are being rescheduled because of decision-making of priorities. For example, the feature "save and load game" was supposed to be done after "create and move orb", but instead rescheduled to 3 sprints after it. There are also many new features included in the development process that are not in the initial design. Lastly, there are some redesigns and bugs in the already existing features, which make the timeline postponed even more.

3. Testing

The testing stage was initially planned to have one iteration. But in the reality of the development process, it was done 3 times. Two of the first iteration was more focused on fixing known issues, while the last iteration focused on polishing the already existing features.

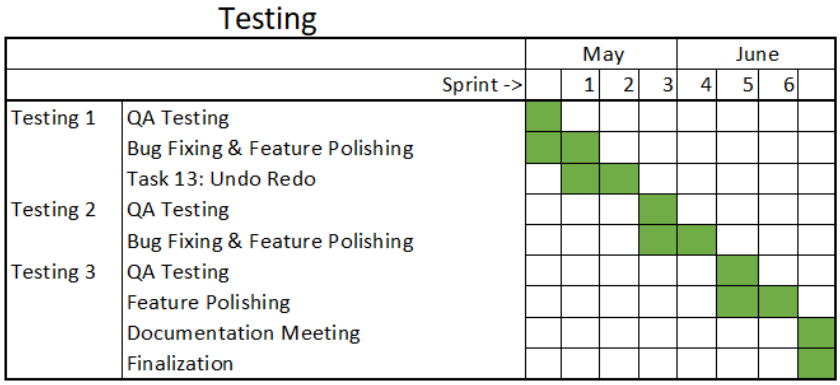


Figure 8.Testing Stage of The Study Case.

4. Release

The final stage consists of updating the documentation. As mentioned, before there were many design changes in the process of development, so the documentation needs to be changed accordingly as well. After that, comes the final meeting to end the project and deliver the program to its owner. It was done in the first week of July 2023.

3.2.3. The Final Version of The Proposed GDLC

From the analysis of the core stages of a GDLC, and the validation of it by a case study, a new version of the GDLC is created. The final version of the proposed GDLC is intended to answer the research question: is there any GLDC that is suitable for a small video game project that is being developed by a small team (consisting of only 4 core divisions or roles: production, game design, programmer, and artists), abnormal way of working (like remote working for example), and short development timeframe (less than 1 year)? To answer that question, the GDLC needed to be adaptable, and only provide the core and necessary steps without complicating the process. The GDLC needs to be "straight to the point" in giving direction, but at the same time adaptable for changes that may come in the development process. The proposed GDLC is shown in the figure below.

Again, like the previous version, this GLDC works in **a semi-agile approach**. What that means (as stated previously) is that the **4 main stages are being done linearly like a waterfall, but each sub-stage (in the production and testing phase) is being done in a scrum-like agile manner**. These sub-stages are being done with sprints as the basis of the working process.

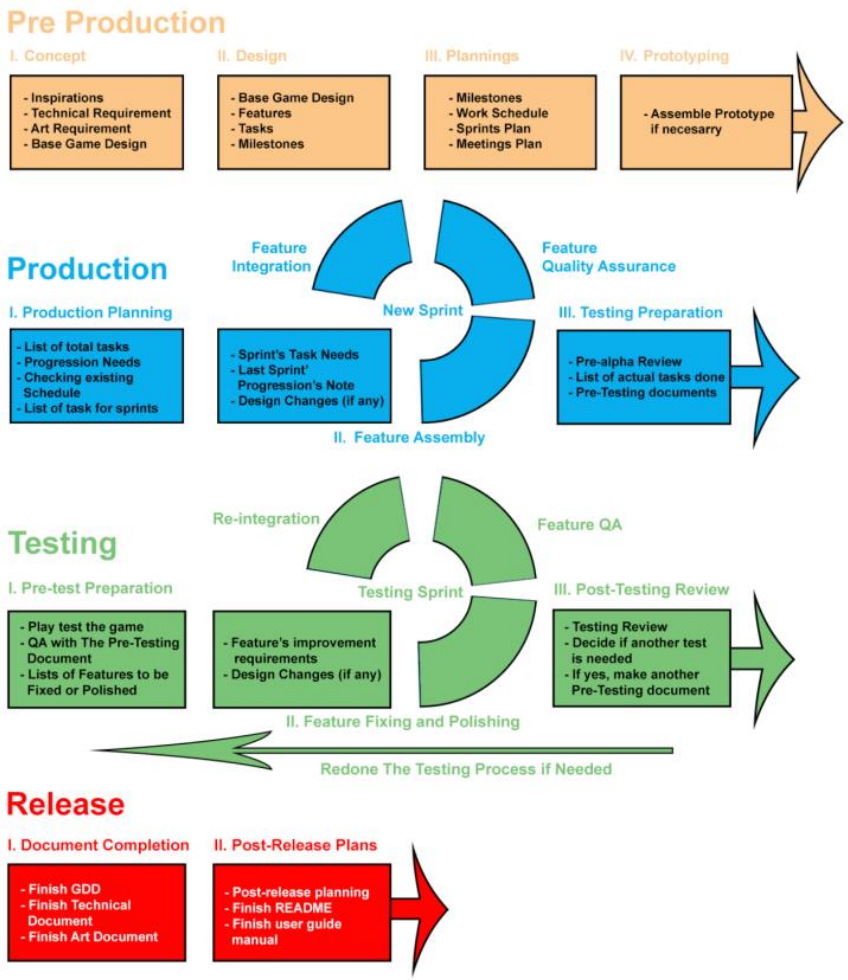


Figure 8. The Final Post-Validation Version of The Proposed GDLC Model.

3.2.3.1. Pre-Production

The Pre-Production phase is the first phase and is done linearly. It is the planning and brainstorming stage before you start building anything. It's where the production and design division designs what the game is going to be about, how it will look, and how it will work. It serves as the foundational stage wherein fundamental concepts are conceived, designs are formulated, comprehensive plans are crafted, and prototype models are constructed. At its core, the Pre-Production phase encompasses several sub-phases:

1. Concept

   This initial step involves the general concept and theme that defines the game. It is during this stage that the production and the designer division (mainly) identify the central premise, gameplay mechanics, and target audience, art requirement, culminating in a conceptual framework that defines the essence of the game. To identify all of that listed above, the team must take inspiration from other media. These **inspirations** may come from another video game or not (such as movies, books, and other kinds of media).

   After all of that has been identified, the art and programming division can determine the **technical requirements** and **art requirements** for this project. What software should be used to make the assets, what programming language and game engine are suited for the kind of this game's project, and which assets need to be made in-house or imported. Every technical need will be written in a TDD (Technical Design Document) draft, and art needs will be written in an ADD (Art Design

Document) draft. Then the game designer (or the division that is responsible for it) creates a GDD draft to document the concept or **base game design** that has been decided upon.

2. Design

Design, in this context, pertains to the visual and interactive blueprint of the game. Developers conceptualize the expansion of the core mechanics that are defined in the Concept sub-phase, characters, environments, user interfaces, and overall aesthetics. In short, the core concept is being expanded into many **features**, and the process for making those features is being expanded into many tasks. Those **tasks** are then collected into a document (usually called "backlog") and grouped into some **milestones** that will be worked on step by step.

3. Planning

It's the sub-phase where the team designs the planning of the actual work. The team then decides on how to work on the already planned milestones and makes a work **schedule/timeline** document. This creates a **production backlog**. Of course, this document is only an estimation, and a changing schedule must be expected, but it's good to anticipate and plan ahead.

The team then needs to allocate human resources, as in assigning who will work on which task. The team also needs to create a plan for what kind of **sprint** in the "production" phase, and the **meetings** that come with it. What will be done at the beginning of a sprint, is there any huddle, will the huddle be done daily, how long does one sprint last (usually one or two weeks per sprint), QA format on each task done, and so on, and so on. These tasks can also be done at the beginning of each sprint, but again, it's good to plan ahead.

4. Prototyping

The last sub-phase is to make a prototype. Like what's been already explained before, a prototype is a product that only focuses on the core mechanics, in the simplest form. You don't need to have a detailed plan, working ethics, or even resources. The goal is "as long as the prototype is assembled and can be played" to the point of covering all the initial concepts, even if the product is a mess. Its sole existence is to give a simple but clear image of what will the actual product be like.

3.2.3.2. Production

The Production phase is the step where the actual work and assembly of the game take place. It's where the tasks, defined during the Pre-Production phase, are worked on in an agile manner. By adapting **scrum**, the tasks are being done in many sprints, each working according to the schedule made in the timeline document. The production phase embraces iterative development cycles that empower developers to adapt, collaborate, and efficiently bring the envisioned game to fruition.

1. Production Planning

Before doing the actual development, planning needs to be made to ensure a smooth workflow, because there might be a change that needs to be made after making the prototype. Lists of the total tasks, which is the **production backlog** (or simply just the backlog), along with their details such as who will do it and when it will be worked on may be reviewed again. Of course, the completion status of one or more tasks might be dependent on other tasks. Thus, the team must determine the progression needs, which consist of (but are not limited to) which task has a lower or higher priority, which tasks need to be done together or in parallel with each other, and which tasks need to be done after a certain task is done. For a clear decision to be made, the team needs to **check the existing schedule**, because there might be a change (and sometimes, a major one) in the timeline. This "progression needs" may be written in a new document, added to the backlog note, or added to the GDD. After the progression need is determined, the team can then **list the tasks to one or more sprints** once again and update the working schedule.

In short, this sub-phase is the review of planning made in the Pre-Production phase. The team may realize some development needs that are not being thought of before making the prototype. The view on which task should be prioritized may change depending on how the process of making the prototype took place. In some cases, there might be some tasks being removed completely from the document, as its feature may be deemed not fit for the game.

2. Sprints

During this sub-phase, the team will work in a relatively short and iterative development cycle, known as sprints, to work on a previously set task and implement the planned features. At the beginning of a sprint, the team needs to **check the current sprint's task(s) and its needs, the last**

sprint's progression notes, and design changes that are decided on the last sprint (if any). From these 3 points, the team can determine if there's a need for change in the working timeline, and if there's a need for adaptation in task work for this sprint.

If everything is already decided, then the team can proceed to do the task assigned to them individually, a process of **feature assembly**. If in a sprint the task is completed, then the features that are created from that task need to be **tested for its quality assurance**, an in-sprint QA test that focuses on testing one specific feature. If the feature meets the quality requirements, then it will be **integrated into the game**. If it's not, or there's a design change, then the workers (either programmers or artists) need to fix it and meet the QA requirements before integrating it. If the task itself is not done in the intended sprint or time, then that lateness needs to be noted so there's a change in the working timeline.

Everything that's been done, especially if it's something that has not been planned before, must be noted to determine how the next sprint will be worked on. The sprints will be done in a loop according to the sprint design that has been decided in the Pre-Production phase. But usually, a proper sprint has:

- A sprint planning on every start of a sprint, that discusses the preparation of that sprint.
- Daily huddle, a small meeting at the start of a working day, on a sprint, to discuss what will be done on that day.
- Daily progression report, to report what has been done on that day.
- A sprint retrospective at every end of a sprint, that discusses the actual work of that sprint, what's being done on schedule or not, and changes (if any).

In each change and progress, GDD, TDD, and ADD need to be updated according to the actual activity.

3. Post-Production Review

In this sub-phase, the development of the video game is technically done. The team then will do a **pre-alpha review**. It is done by reviewing **which tasks are completed or dropped**, which features have been changed from the initial design or being done exactly from it, and the overall design change of the video game. Although the team needs to be optimistic, it is wise to predict issues that may or may not appear in-game. All of that can be noted into one document, creating a **pre-testing document** out of it.

3.2.3.3. Testing

The Testing phase in this GDLC is to ensure the game works properly and is fun to play before it's released. It consists of giving the game a thorough check-up to find and fix any problems or issues with an agile methodology approach. There are 2 kinds of testing, alpha testing (which focuses on fixing and stabilizing issues), and beta testing (which focuses on polishing already-working features). Alpha testing(s) needed to be done first for a team to proceed with beta testing.

1. Pre-Testing

Since there's a technically complete build of a game after the production phase is finished, the first thing the testers (either the actual development team or a selected player candidates outside of it) need to do is to **play the whole game accordingly**. Play it as if the game has already been released. Then, note every issue, every bug, every glitch, and every feature that feels unpolished. From that list of issues, the team can create a list of "fixing tasks". Combine that with the pre-testing document (that was created in the previous phase), the team can determine the timeline to fix the issues and create a **QA document**. The document must include a list of features to be fixed or polished, which is the **testing backlog**.

2. Sprints of Feature Fixing and Polishing

It is the same sub-step as a sprint in the production phase. The difference is, that instead of creating a feature, the team fixes and improves features that are listed in the "fixing tasks" backlog. The features are fixed, checked' quality assurance, and reintegrated into the game, or dropped off completely from it.

3. Post-Testing Review

It is pretty much the same as a post-production review, only this time the team is reviewing a game that's already been more polished than before. The team needs to decide, to either call it done and release the game, or another testing iteration is needed. If there's a need for another testing,

then the team must decide the type of testing (alpha or beta), and who will be the ones that play the game (internally inside the team or external feedback). If not, then the team and the video game can move on to the final phase.

3.2.3.4. Release

This final phase consists of publishing the video game, finalizing the documentation, and post-release planning (if any, but these days it's likely). The team must finalize the document as best as possible, as these documents are the bridge that connects the minds of the players and the team.

1. Document completion

In this stage, ensure that all necessary documents are thoroughly completed and ready for reference. Every change, even the working schedule, needed to be documented according to the reality of the development. This documentation can be a precious reverence for another development or this game's post-release planning.

- Game Design Document (GDD):
  - Review and update the GDD to reflect any changes made during development.
  - Ensure that the GDD accurately describes the game's mechanics, features, and objectives.
- Technical Design Document (TDD):
  - Revise and finalize the TDD to document the technical aspects of the game's development.
  - Include information about architecture, programming techniques, and technology used.
- Art Design Document (ADD):
  - Complete the Art Document with detailed information about visual assets, art style, and guidelines.
  - Include asset lists, concept art, character designs, and any relevant artistic references.

2. Post-Release Plans

In this stage, focus on tasks related to launching the game and planning for the period after release:

- README.txt File:
  - Create a README file that briefly overviews the game, installation instructions, system requirements, and known issues.
- User Guide Manual (if needed):
  - Develop a user guide explaining gameplay mechanics, controls, and complex features.
  - Include troubleshooting tips and commonly asked questions.
  - But, these days, the functionality of this manual has already been covered by in-game tutorials.
- Post-Release Planning:
  - Identify strategies for post-release support, updates, and potential expansions. Redo the whole GDLC if needed.
  - Plan how to address user feedback, bug reports, and suggestions for improvements.
  - Establish a timeline for updates and prioritize features based on user demand and the completed documentation.

If everything above has been completed, the team can now release the game. Normally, a game developer team launches the game by publicizing it in an online store based on the chosen platform, followed by or after a game trailer to hype up the market. But some types of games have just been given as is to a contracted user, usually a serious game. But no matter the type of release, it is crucial to communicate with the users following the launch. Their feedback is the backbone of pre-release planning. Monitor the flow of hype to ensure the launch of the game's reputation is as successful as possible.

4. CONCLUSION

There are 4 phases of game development, according to this GDLC. Those phases are **Pre-Production, Production, Testing, and Release**. The Pre-production and Release phases are done linearly, while the Production and Testing phases are done with an agile approach. The agile approach itself was adapted or implemented with **Scrum**. Those 4 phases consist of tasks needed to be done by a team that has at least 3 divisions, which are the game design division, the programmer division, and the artists division.

Although arguably a team needs to look at the business and marketing side of a game publishing activity, it is these 4 phases and those 3 divisions that made the core of software engineering in video game development.

This GDLC in a broad sense implements a **semi agile approach** with Scrum as the agile part of it. As a whole, 4 stages are being done linearly like waterfall, but each stage has its own way of doing it. As described above, Production and Testing phases are done with an agile Scrum approach. The iteration loop only happens from the start to the finish of each stage. Those 2 stages also have their own backlog, and sprint rules based on the initiation meetings agreement. The Scrum aspect is implemented this way because only those 2 stages needed to be adaptable to changes, while the other 2 (pre-production and release stages) need to be consistent in its working process.

To further polish this GDLC, it is necessary to keep using this lifecycle in actual game development. The comparison of the actual process of creating a game and the steps provided in this GDLC will determine if the lifecycle can accurately help a team or if further methodology polishing is needed.

ACKNOWLEDGEMENTS

Only with blessing from THE LORD ALMIGHTY can the authors write this paper titled **"Applying Scrum in A Game Development Life Cycle for Small Scale Game Project".** The responding author would also like to thank the 2 other authors as guidance to conduct the research this paper is based on. The authors would also like to thank all those who help participate in the study case validation, namely proffessors who guide us and the work partners and friends from Redamantine Studios. It is a great pleasure to know this GDLC may be of help for the game development community.

REFERENCES

[1] V, K., Savita Chaudhary and Radhika D (2022) "Feature Extraction in Music information retrival using Machine Learning Algorithms", *International Journal of Data Informatics and Intelligent Computing*, 1(1), pp. 1–10. doi: 10.5281/zenodo.7093881.

[2] Hu, T. and Desai, J.P. (2004) Soft-Tissue Material Properties under Large Deformation: Strain Rate Effect. *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, San Francisco, 1-5 September 2004, 2758-2761.

[3] Ortega, R., Loria, A. and Kelly, R. (1995) A Semiglobally Stable Output Feedback PI2D Regulator for Robot Manipulators. *IEEE Transactions on Automatic Control*, 40, 1432-1436. http://dx.doi.org/10.1109/9.402235

[4] Wit, E. and McClure, J. (2004) Statistics for Microarrays: Design, Analysis, and Infer-ence. 5th Edition, John Wiley & Sons Ltd., Chichester.

[5] Prasad, A.S. (1982) Clinical and Biochemical Spectrum of Zinc Deficiency in Human Subjects. In: Prasad, A.S., Ed., *Clinical, Biochemical and Nutritional Aspects of Trace El-ements*, Alan R. Liss, Inc., New York, 5-15.

[6] Giambastiani, B.M.S. (2007) Evoluzione Idrologica ed Idrogeologica Della Pineta di san Vitale (Ravenna). Ph.D. Thesis, Bologna University, Bologna.

[7] Wu, J.K. (1994) Two Problems of Computer Mechanics Program System. *Proceedings of Finite Element Analysis and CAD*, Peking University Press, Beijing, 9-15.

[8] Honeycutt, L. (1998) Communication and Design Course. http://dcr.rpi.edu/commdesign/class1.html

[9] Wright and Wright, W. (1906) Flying-Machine. US Patent No. 821393.