**Chapter 3**

**Project Development Methodology**

**3.1 Software Development Life cycle**

Software development life cycle is the most important element in software development. It depicts the necessary phases in software development.

Software Development Life Cycle (SDLC) is a process of designing, building and maintaining software systems. Typically, it includes various phases from preliminary development analysis to post-development software testing and evaluation. It also consists of the models and methodologies that development teams use to develop the software systems, which the methodologies form the framework for planning and controlling the entire development process.

A software application or an information system is designed to perform a particular set of tasks. Often, this set of tasks that the system will perform provides well-defined results, which involve complex computation and processing. It is therefore a harsh and tedious job to govern the entire development process to ensure that the end-product comprises a high degree of integrity and robustness, as well as user acceptance. Thus, a systematic development process which is able to emphasize on the understanding of the scope and complexity of the total development process is essential to achieve the said characteristics of a successful system.

Software Engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software because it integrates significant mathematics, computer science and practices whose origins are in Engineering. Various processes and methodologies have been developed over the last few decades to improve software quality, with varying degrees of success. However it is widely agreed that no single approach will prevent project overruns and failures in all cases. Software projects that are large, complicated, poorly-specified, and involve unfamiliar aspects, are still particularly vulnerable to large, unanticipated problems. A software development process is a structure imposed on the development of a software product.

There are several models for such processes, each describing an approach test to a variety of tasks or activities that take place during the process. It aims to be the standard that defines all the tasks required for developing and maintaining software. These classic software life cycle models usually include some version or subset of the following activities:

* Planning and Visualization
* Requirement Analysis
* Software Modeling and Design
* Coding
* Testing and integration
* Deployment and Maintenance

**Fig. 3.1 SDLC**

And so far, there exist many SDLC models, one of which is the Agile model which will be explained in the next session.

**3.1.1 AGILE Software Development**

Agile development is based on the idea of incremental and iterative development, in which the phases within a development life cycle are revisited over and over again. It iteratively improves software by using customer feedback to converge on solutions.

In agile development, rather than a single large process model that is implemented in conventional SDLC, the development life cycle is divided into smaller parts, called “increments” or “iterations”, in which each of these increments touch on each of the conventional phases of development. According to Agile Manifesto, the major factors of agile factors include the following four:

1. Early customer involvement

2. Iterative development

3. Self-organizing teams

4. Adaptation to change

There are currently six methods that are identified as agile development methods, which are Crystal methodologies, dynamic software development method, feature-driven development, lean software

development, scrum, and extreme programming.

Agile, also called an iterative life cycle model as it does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.

## **Main agile software development lifecycle phases**

The agile lifecycle is a structured series of stages that a product goes through. It consists of six phases:

### **Meet & Plan for requirements determination**

Stakeholders conduct an overall project assessment to determine the time and resources required for the development process. At the same stage, the owner assesses the risks and prioritizes the various functions depending on their business value.

1. **Design**

The software owner meets with the software development team and introduces them to the requirements outlined in the first step. The group then discusses the sequence for introducing functions and identifies the essential tools – the programming language, syntax libraries, and basic frameworks. At the same stage, software development teams can prototype the expected user interface.

1. **Development and coding**

After agreeing on the plan with the customer, the team develops the product itself. The product is delivered in stages, in separate sprints, each designed to improve the current version of the product. The initial release is likely to undergo many changes to provide improved functionality and new features.

Each cycle includes testing, and the final product must also undergo final testing. For this phase, you can use Scrum and the Kanban methodology, the development process based on individual tasks.

1. **Integration and testing**

At this point, the product becomes available to consumers, so the team must conduct a series of tests to ensure that the software is fully functional. If potential bugs or flaws are found, the developers will fix them immediately. At this stage, they also collected consumer feedback.

1. **Implementation and releasing**

The software is now fully deployed and available to customers. This action puts him in the maintenance phase. During this phase, the software development team provides ongoing support to keep the system running smoothly and fix any new bugs. Over time, further iterations are possible to update an existing product or add other functionality.

1. **Feedback**

That is the last stage of the Agile development cycle. After completing all the previous stages of development, the development team presents to the owner the result achieved in meeting the requirements. After that, the Agile software development phases start over – either with a new iteration or moving to the next stage and scaling Agile.

**Fig. 3.2 Agile model phases**

**Advantages:** The most important advantages of this model are the ability to respond to the changing requirements of the project. This ensures that the efforts of the development team are not wasted, which is often the case with the other methodologies. The changes are integrated immediately, which saves trouble later. There is no guesswork between the development team and the customer, as there is face to face communication and continuous inputs from the client. The documents are to the point, which does not leave any space for ambiguity. The culmination of this is that high quality software is delivered to the client in the shortest period of time and leaves the customer satisfied.

**Disadvantages:** If the projects are smaller projects, then using the agile model is certainly profitable, but if it is a large project, then it becomes difficult to judge the efforts and the time required for the project in the software development life cycle. Since the requirements are ever-changing, there is hardly any emphasis, which is laid on designing and documentation. Therefore, chances of the project going off the track easily are much more. The added problem is if the customer representative is not sure, then the project going off track increases manifold. Only senior developers are in a better position to take the decisions necessary for the agile type of development, which leaves hardly any room for newbie programmers, until it is combined with the senior’s resources.

**3.2 Chosen Methodology**

The Scrum approach is our chosen methodology as the main idea of Scrum is that systems development involves several environmental and technical variables (e.g. requirements, time frame, resources, and technology) that are likely to change during the process. This makes the development process unpredictable and complex, requiring flexibility of the systems development process for it to be able to respond to the changes. Scrum helps to improve the existing engineering practices (e.g. testing practices) in an organization, for it involves frequent management activities aiming at consistently identifying any deficiencies or impediments in the development process as well as the practices that are used.

In scrum approach responsibilities are divided based on roles and there are six identifiable roles in Scrum that have different tasks and purposes during the process and its practices: Scrum Master, Product Owner, Scrum Team, Customer, User and Management.#

**3.2.1 Roles and responsibilities**

**Scrum Master**

Scrum Master is a new management role introduced by Scrum. Scrum Master is responsible for ensuring that the project is carried through according to the practices, values and rules of Scrum and that it progresses as planned. Scrum Master interacts with the project team as well as with the customer and the management during the project. He is also responsible for ensuring that any impediments are removed and changed in the process to keep the team working as productively as possible.

**Product Owner**

Product Owner is officially responsible for the project, managing, controlling and making visible the Product Backlog list. He is selected by the Scrum Master, the customer and the management. He makes the final decisions of the tasks related to product Backlog, participates in estimating the development effort for Backlog items and turns the issues in the Backlog into features to be developed.

**Scrum Team**

Scrum Team is the project team that has the authority to decide on the necessary actions and to organize itself in order to achieve the goals of each Sprint. The scrum team y is involved, for example, in effort estimation, creating the Sprint Backlog, reviewing the product Backlog list and suggesting impediments that need to be removed from the project.

**User**

Users participate in the tasks related to product Backlog items for the system being developed or enhanced.

**Management**

Management is in charge of final decision making, along with the charters, standards and conventions to be followed in the project. Management also participates in the setting of goals and requirements. For example, the management is involved in selecting the Product Owner, gauging the progress and reducing the Backlog with Scrum Master.

Scrum does not require or provide any specific software development methods/practices to be used. Instead, it requires certain management practices and tools in the various phases of Scrum to avoid the chaos caused by unpredictability and complexity.

**3.2.2 Scrum Practices**

Scrum does not require or provide any specific software development methods/practices to be used. Instead, it requires certain management practices and tools in the various phases of Scrum to avoid the chaos caused by unpredictability and complexity.

In the following, the description of Scrum practices is given:



**Fig. 3.3 Practices of Scrum**

**Product Backlog :** Product Backlog defines everything that is needed in the final product based on current knowledge. Thus, Product Backlog defines the work to be done in the project. It comprises a prioritized and constantly updated list of business and technical requirements for the system being built or enhanced. Backlog items can include, for example, features, functions, bug fixes, defects, requested enhancements and technology upgrades. Also issues requiring solution before other Backlog items can be done are included in the list. Multiple actors can participate in generating Product Backlog items, such as customer, project team, marketing and sales, management and customer support.

This practice includes the tasks for creating the Product Backlog list, and controlling it consistently during the process by adding, removing, specifying, updating, and prioritizing Product Backlog items. The Product Owner is responsible for maintaining the Product Backlog.

**Sprint:** Sprint is the procedure of adapting to the changing environmental variables (requirements, time, resources, knowledge, technology etc.). The Scrum Team organizes itself to produce a new executable product increment in a Sprint that lasts approximately thirty calendar days. The working tools of the team are Sprint Planning Meetings, Sprint Backlog and Daily Scrum meetings. The Sprint with its practices and inputs is illustrated in Figure 3.3.

**Sprint Planning:** meeting A Sprint Planning Meeting is a two-phase meeting organized by the Scrum Master. The customers, users, management, Product Owner and Scrum Team participate in the first phase of the meeting to decide upon the goals and the functionality of the next Sprint. The second phase of the meeting is held by the Scrum Master and the Scrum Team focusing on how the product increment is implemented during the Sprint.

**Sprint Backlog:** Sprint Backlog is the starting point for each Sprint. It is a list of Product Backlog items selected to be implemented in the next Sprint. The items are selected by the Scrum Team together with the Scrum Master and the Product Owner in the Sprint Planning meeting, on the basis of the prioritized items (see 3.3) and goals set for the Sprint. Unlike the Product Backlog, the Sprint Backlog is stable until the Sprint (i.e. 30 days) is completed. When all the items in the Sprint Backlog are completed, a new iteration of the system is delivered.

**Daily Scrum meetings:** Daily Scrum meetings are organized to keep track of the progress of the Scrum Team continuously and they also serve as planning meetings: what has been done since the last meeting and what is to be done before the next one. Also problems and other variable matters are discussed and controlled in this short (approximately 15 minutes) meeting held daily. Any deficiencies or impediments in the systems development process or engineering practices are looked for, identified and removed to improve the process. The Scrum Master conducts the Scrum meetings. Besides the Scrum team also the management, for example, can participate in the meeting.

**Sprint Review:** meeting On the last day of the Sprint, the Scrum Team and the Scrum Master present the results (i.e. working product increment) of the Sprint to the management, customers, users and the Product Owner in an informal meeting. The participants assess the product increment and make the decision about the following activities. The review meeting may bring out new Backlog items and even change the direction of the system being built.

Our scrum process includes three phases: pre-game, development and post-game

**The pre-game phase** includes two sub-phases: Planning and Architecture/High level design. Planning includes the definition of the system being developed. A Product Backlog list (see 3.3) is created containing all the requirements that were currently known in our system. The requirements can originate from our targeted Users which are the kids who are diagnosed with ADHD and their parents. The requirements are prioritized and the effort needed for their implementation is estimated. The product Backlog list is constantly updated with new and more detailed items, as well as with more accurate estimations and new priority orders. Planning also includes the definition of the project tools and other resources, risk assessment and controlling issues and training needs. At every iteration, the updated product Backlog is reviewed by our team so as to gain our commitment for the next iteration.

The following table shows a part of our backlog when our team was still working on the project :

|  | **Product backlog items** | **Estimation of difficulty** | **Priority** | **Sprint** | **Status** | **Estimation of effort** |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | Data Flow Diagram | low | 5 | 1 | Done | 4 |
| **2** | Use Case Diagram | low | 5 | 1 | In Progress | 3 |
| **3** | Sequence Diagram | medium | 4 | 1 | To Do | 10 |
| **4** | Entity Relationship Diagram | low | 5 | 1 | In Progress | 6 |
| **5** | welcome page | low | 7 | 1 | Done | 3 |
| **6** | Learning Phases | medium | 3 | 5 | To Do | 1 |
| **7** | MCQ Test Phases | low | 3 | 5 | To Do | 6 |
| **8** | Parents Registration | low | 9 | 1 | Done | 5 |
| **9** | Login | low | 10 | 1 | Done | 3 |
| **10** | Game Design | high | 1 | 2 | To Do | 72 |
| **11** | Splash Screen | medium | 6 | 4 | To Do | 12 |
| **12** | Menu Screen | medium | 6 | 4 | To Do | 8 |
| **13** | Game Categories | high | 1 | 2 | To Do | 10 |
| **14** | Memory game design | high | 1 | 2 | To Do | 6 |
| **15** | Order game design | high | 1 | 2 | To Do | 4 |
| **16** | Math game design | high | 1 | 3 | To Do | 7 |
| **17** | Alphabet collecting game design | high | 1 | 3 | To Do | 13 |
| **18** | Finding missing alphabet game design | high | 1 | 3 | To Do | 13 |
| **19** | Scoring System | high | 2 | 4 | To Do | 11 |
| **20** | Educational Content | low | 3 | 3 | To Do | 5 |
| **21** | Game Shop | low | 8 | 6 | To Do | 2 |
| **22** | Reporting System | medium | 5 | 4 | To Do | 8 |
| **23** | Database | high | 3 | 6 | To Do | 48 |
| **25** | Game Theme | medium | 9 | 6 | To Do | 10 |

**In the architecture phase**: the high level design of the system including the architecture is planned based on the current items in the Product Backlog. In case of an enhancement to an existing system, the changes needed for implementing the Backlog items are identified along with the problems they may cause. A design review meeting is held to go over the proposals for the implementation and decisions are made on the basis of this review. In addition, preliminary plans for the contents of releases are prepared.

**The development phase:** (also called the game phase) is the agile part of the Scrum approach. This phase is treated as a "black box" where the unpredictable is expected. The different environmental and technical variables (such as time frame, quality, requirements, resources, implementation technologies and tools, and even development methods) identified in Scrum, which may change during the process, are observed and controlled through various Scrum practices during the Sprints of the development phase. Rather than taking these matters into consideration only at the beginning of the software development project, Scrum aims at controlling them constantly in order to be able to flexibly adapt to the changes. In the development phase the system is developed in Sprints. Sprints are iterative cycles where the functionality is developed or enhanced to produce new increments. Each Sprint includes the traditional phases of software development: requirements, analysis, design, evolution and delivery (Figure 3.3) phases. The architecture and the design of the system evolve during the Sprint development. One Sprint is planned to last from one week to one month. There may be, for example, three to eight Sprints in one systems development process before the system is ready for distribution. Also there may be more than one team building the increment.

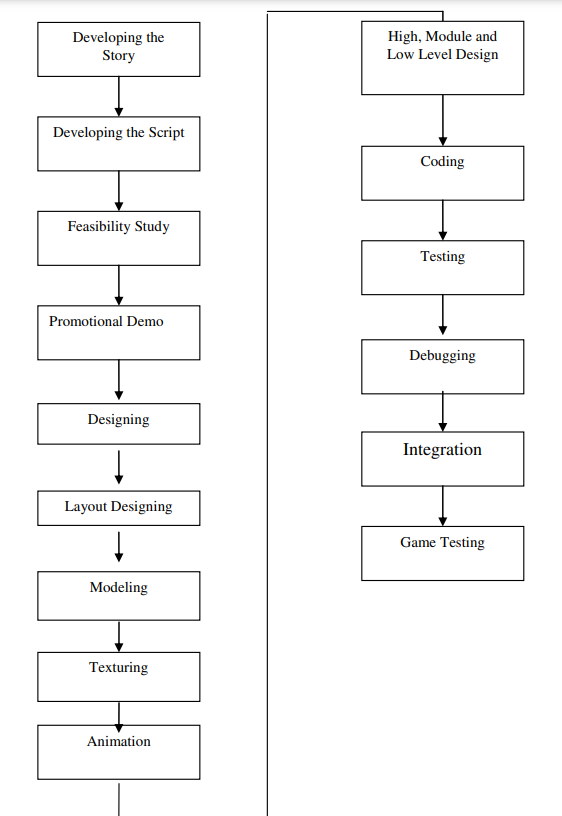
The development phase will be illustrated with all details in the next chapters.

**The post-game phase:** contains the closure of the release. This phase is entered when an agreement has been made that the environmental variables such as the requirements are completed. In this case, no more items and issues can be found nor can any new ones be invented. The system is now ready for the release and the preparation for this is done during the post-game phase, including the tasks such as documentation, integration and system testing to release a complete usable software

**3.2.3 Game Development Phases**

The birth of video games has slowly shifted the meaning of traditional games into a digitalized multimedia game. The term ‘games’ refers to the meaning of video games. Nowadays, games can be played on almost any device, and that is why developing games can be a profitable industry. To support the growth of the gaming industry, several original equipment manufacturers (OEM) publicly distribute their software development kit (SDK) and application programming interface (API) to attract people to become “indie developers” . According to Pressman, games are a kind of software which provides entertainment . However, game development using only software development life cycle (SDLC) faces several challenges. While SDLC is a systematic process of engineering to develop software, games are not purely a product of pure engineering. Game also is not just pure art, a creation of creativity and imaginative thinking, but game is more like a craft, created from the combination of interleaving, multidisciplinary aspects, from art, music, programming, acting, and the management and integration of those aspects. Therefore, a game development requires specific guidelines which govern its development process, the game development lifecycle (GDLC).

The phases in the game development life cycle are discussed in the following sections and the flow is shown in figure 3.4.



**Fig. 3.4 Phases in Games Development**

**Developing the story**

To start with, a core theme is developed with a well defined storyline. The story is generally targeted at a specific age group. Every story and game starts with an idea or a concept. This idea is then developed further. Characters are decided, situations, events and places are created, relationships determined and so on. Following that, various conflict and resolution scenarios are added. All the stories follow a similar structure. Apparently the translation of a story into a game is mere conflict resolution that can be easily achieved based on the designer’s creativity.

**Developing the script**

The story is then fine tuned to obtain a tight script that ensures a seamless flow of the game from one level to the other level. The better the script the better the game will be. The script writing needs a great deal of imagination and creativity. The key to writing a good script is to capture the imagination of the childrens. The characters and the backdrops are created and decided during this phase.

**Feasibility study**

This study is needed to assess the feasibility of a game. In addition it helps to freeze the requirements, the scope, profitability and other inferences of the proposed game. The areas to be reviewed within the feasibility study are overall analyses of the requirements, the pricing, technical, organizational, cultural and legal issues and the schedule of the project. This is a phase where project leaders, project design and development personnel and research group will be working together to create a game as a complete product. The programmers analyze and explain the programming limitations to the project managers. The outcome of the feasibility study is the feasibility study report, which comprises two parts - management summary and technical specification. The findings and recommendations are tabulated in the report.

**Promotional Demo**

A plan for the promotional demo for the game is prepared and is executed. Promotional demo is needed to attract potential customers. At this stage, since the game is not implemented the live demonstration is not possible. Hence a basic trial version is created for promotions.

**Designing**

The three main designing elements are the character design, background design and sets & props design. Computer games utilize characters to a large extent. In Character Design, while designing a convincing character for games, emphasis is given to characters with strong personalities and who can be of visual interest. This designing phase requires various techniques like use of visual design themes, silhouettes, poses, facial expressions and behavior of the characters.

**Layout design**

In this stage the layout for the entire game and for all the levels is designed. The layout represents the various sets, the passage ways with hurdles and the escape routes.. Every type of game design map will have certain constraints. Covering all the constraints in the map is quite challenging.

**Modeling**

Modeling is like character modelings, rules modeling. Models have to provide enough data so that the finer details can be rendered in an effective manner. There are many advanced techniques which are not very expensive, and they offer lots of features for creating models. One such advanced technique is the use of normal mapping.

**Texturing**

Texturing is a technique for adding detail, providing surface texture or color so that a realistic look can be given to the characters, sets and props. It is very common in almost all 3D applications.

**Animation**

After designing, modeling and texturing stages, the characters have to be animated to represent the sequence of actions in the game. Because the flow of the game is not linear, mesh methodology is used to animate a character.

The animation phase includes motion capture (mocap) animation and keyframe animation. In mocap animation each and every movement of the characters or objects enacted by humans or trained animals with either optical or magnetic sensors placed on them , is captured and recorded to create a series of scenes. These movements are later applied to the models. The only disadvantage is that it requires use of expensive camera/magnetic systems and very specific hardware and software. Key frames are snapshots of an image at a single point in time. Keyframe animation is the cycling of key frames to give the illusion of movement.

**High and Low level of Design**

The high level design gives the overall design of the game by identifying all the elements. It shows the abstract view of the system and it hides some of the details. The design also gives the relationships between the various modules involved in the development process in the form of data flow, flow charts, data structures etc., In module design high level design is segregated into various modules. The low level design details the high level design (i.e) the logic behind each of the modules is defined .

**Coding**

Coding is an important part of the game development process. This phase includes not only coding for making the gaming interactive but also for implementing the interface design. Various game engines are available in the market to code a game and execute the code and edit the game. There are separate sets of tools for making 2D games and 3D games. The 2D games tools include, GameMaker, Multimedia Fusion 2, Construct, Flixel, FlashPunk and Stencyl. 3D games tools are Unity , UDK , XNA , BlitzMax , jMonkeyEngine and Torque to name a few. Most games today on the PC platform are written using C++.

**3.2.4 How to limit the amount of iteration during video gaming development?**

Iteration was mentioned and tackled in a subtle manner and that behavior can be considered expected, since iteration is normalized in the video game industry. steadily the mentality of overworking in the video game industry is becoming a staple. Alas, there were many video game developers who nevertheless, without condemning iteration or identifying it as a problem, proposed effective solutions, like Kitase for FF7 Remake. These solutions are not mere assumptions or hopeful thinking, they are based on actual facts and have been put to practice and proved to be successful in limiting or eliminating iteration in rather unique ways.

These methods, depending on the situation, demand the reliance of developers either to already established franchises or to successful mechanics and aesthetics found in previous, unrelated video games. For instance, sequels of the games can just build upon the previous titles mechanics and expand on them. Another solution is that a good enough game can be published and then the sequels can use the exact same mechanics and design choices until the last one reaches the fullest potential that was initially envisioned for the first title. These solutions greatly accelerate the development and publishing pace and allow the companies to put in the market more products of a good quality at a faster pace.

Furthermore, another alternative solution is to increase the interaction of developers and customers to such an extent that developers will constantly seek how customers react and respond to choices about the game during the development. The concept of alliance with the customers may seem paradoxical at first, since it is not a practice that the majority of companies do, but this new practice offers many advantages. More importantly, constant interactivity with the customers can provide feedback that will prevent the need of retesting or trying to guess the tastes of the audience, thus limiting the need of iteration.

There are other types of interactions with customers or in other words allying with them. For instance, customers don’t have to interfere during the development of the game, a method that will alleviate the frustration of some developers who might feel that their work is compromised. Instead, customers can have the opportunity to receive the game in such a state that they can construct it to their own image, with the features they want to see. Thus, without the need of polishing the game, the developing company doesn’t have to conduct the time consuming qualitative iteration in the last stages of development. Such is the case of Bethesda, which has pushed the boundaries of interacting with the customers even further, as the company publishes games with clear problems and deficiencies (and at the same time rich in context and gameplay value) and leaves their corrections to the customers themselves by enabling the customers to do modifications to the games. Alternatively, a game can be published in fractures or episodes until gradually all the episodes will constitute a full game. In that way, the production pace of products is accelerated and the developers spend less resources for a small size product. In that regard, a small size product demands less iteration to be materialized. The developers can also evaluate the commercial reception of that small size product before they proceed to develop the other episodes or in other words before they dedicate the resources to formulate a fully fledged game.

**3.2.5 How Agile development’s iteration influences/affects the development of a game?**

This data stems from both the conducted interviews and also the collected secondary data. Through the intermediate theory, these different types of data act supplementary to each other.

It is important to note the level of understanding of Agile methodology by the video game industry’s associates. One important finding is that the Agile methodology is widely used by many developers in the video gaming industry. The notion and the secondary data also provides confirmation towards that conclusion. The theoretical framework provides data as well confirming that Agile methodology is the favorite methodology of managers in the video game industry. Agile is viewed by managers as ideal to develop video games and it turns into a staple in the industry. What also becomes staple in the industry are the characteristic features of Agile like the examined iteration in this thesis. In that sense, it becomes clear that the extensive adoption of Agile methodology results in the extensive adoption of iteration as well. This observation supports the significance and validity of that research question, by proving that the Agile methodology is becoming mainstream. The effects of Agile can be seen frequently and will be seen more often, the more developers decide to adopt it. products, the revenue in return is going to be smaller. It should be noted that project failure is not necessary a result of managerial incompetence but it’s due to the nature of iteration. Agile’s iteration commands and proclaims essentially that employees should never settle to staples and commonalities and demands constant experimentation to achieve uniqueness. This mentality goes against the best interest of any company to make profits.

According to all interviewees, iteration is not affecting the quality of creativity. The decision making in that area is mostly a collective process that takes into account both creative sensibilities and also practical issues found often in the video game industry, like whether some choices have an appeal to customers or that the creative vision is not static and constantly evolving and changing in accordance to the development of the video game. This is an important finding in this thesis, as artistic integrity constitutes the only layer of development that is not only unaffected by iteration but in return is enhanced by it. It seems to be widely accepted by developers that constant retesting and the benefit of more time adds value to the artistic vision and more chances to realize itself. This finding contradicts the findings of other researches which claim that artists resist the principles of Agile due to concerns over restricting their creativity.

**3.3 Applications of Chosen Methodology**

In this section, we briefly discuss some of the existing systems and games that are developed using agile methodology and scrum approach.

**FunPhy**

FunPhy which stands for Fun Physics and it is a mobile game application that supports fun learning for students especially for physics subjects. It was designed by using Agile methodology.

This educational game is intended for junior high schools so that it can help students to master physics. An important characteristic in the system development with Agile method is oriented to the completion of the information system and the software application as a product . Compared to processes and tools, the Agile method focuses on the interaction between individuals . In addition, a change arose during the development process gives better response than following the set plan from the first step of the project.

**Fig. 3.5 FunPhy Game**

**PRO FEEL GOLF**

Definitely, Agile is a proven way to achieve success. It helps in easy tracking of the progress and turns the whole process into a scalable one, that’s why pro feel golf game is also developed based on Agile methodology using Unity3D technology and it is a Sports-simulation Genre game for iOS & Android.



**Fig. 3.6 PRO FEEL GOLF**

**Game**

**Kids Games: For Toddlers 3-5**

It is an amazing learning and educational game for children of all ages. It has been developed using agile methodology and it is designed to help teach colors, shapes, coordination, motor skills, memory, and more

**Fig. 3.7 Kids Games**

**Antidote COVID-19**

This game is another example of the games that use agile methodology in its development process.The Antidote COVID-19 is a game to turn complex, scientific information into a fun learning experience. During the course of the game, players will learn about their immune system, pathogens, vaccines and how to protect themselves from COVID-19.



**Fig. 3.8 Antidote COVID-19**

**Game**