SOFTWARE DEVELOPMENT LIFE CYCLE MODEL

The required Software Development Life Cycle Model for online game streaming application called GStream.

19 SEPTEMBER 2023

HD SECTOR

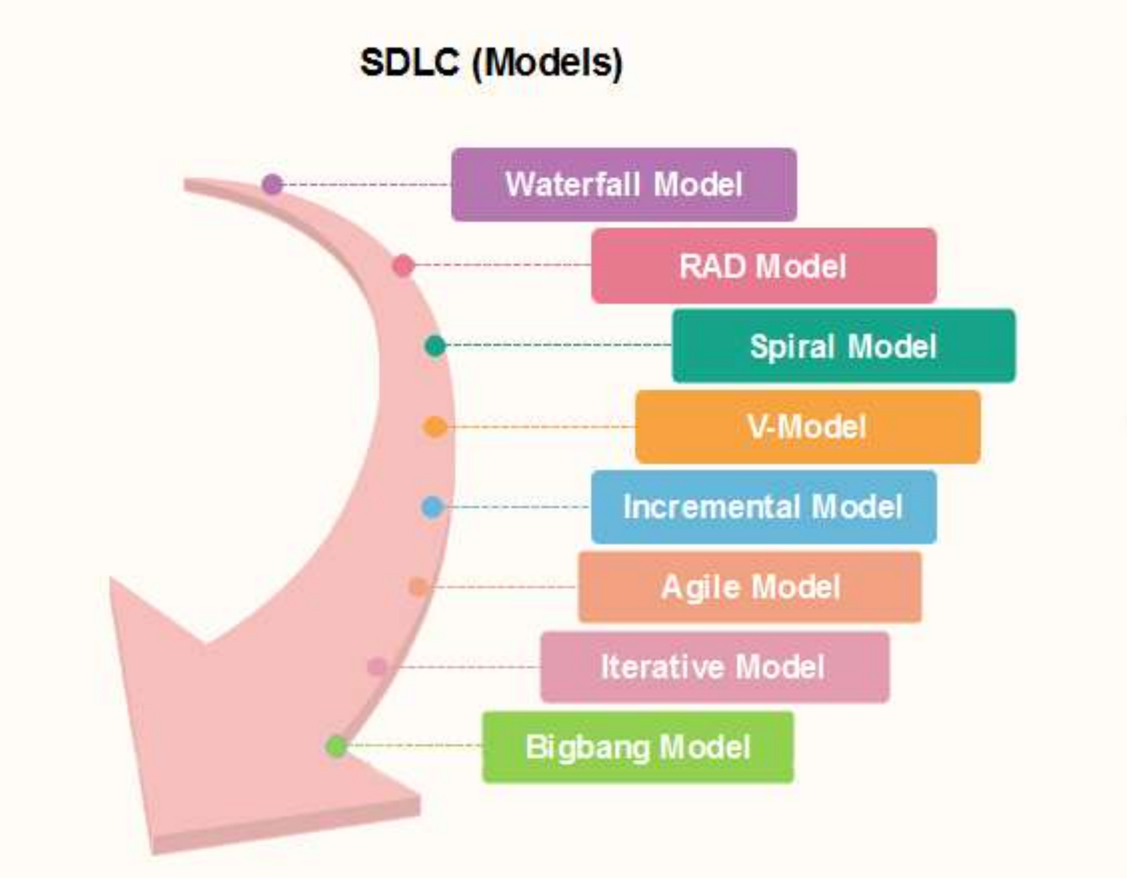
Contents

1. SDLC Models

Software Development life cycle (SDLC) is a spiritual model used in project management that defines the stages include in an information system development project, from an initial feasibility study to the maintenance of the completed application.

There are different software development life cycle models specify and design, which are followed during the software development phase. These models are also called "Software Development Process Models." Each process model follows a series of phase unique to its type to ensure success in the step of software development.

Here, are some important phases of SDLC life cycle:



Waterfall Model

The waterfall is a universally accepted SDLC model. In this method, the whole process of software development is divided into various phases.

The waterfall model is a continuous software development model in which development is seen as flowing steadily downwards (like a waterfall) through the steps of requirements analysis, design, implementation, testing (validation), integration, and maintenance.

Linear ordering of activities has some significant consequences. First, to identify the end of a phase and the beginning of the next, some certification techniques have to be employed at the end of each step. Some verification and validation usually do this mean that will ensure that the output of the stage is consistent with its input (which is the output of the previous step), and that the output of the stage is consistent with the overall requirements of the system.

RAD Model

RAD or Rapid Application Development process is an adoption of the waterfall model; it targets developing software in a short period. The RAD model is based on the concept that a better system can be developed in lesser time by using focus groups to gather system requirements.

- Business Modeling
- Data Modeling
- Process Modeling
- Application Generation
- Testing and Turnover

Spiral Model

The spiral model is a risk-driven process model. This SDLC model helps the group to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc. The spiral technique is a combination of rapid prototyping and concurrency in design and development activities.

Each cycle in the spiral begins with the identification of objectives for that cycle, the different alternatives that are possible for achieving the goals, and the constraints that exist. This is the first quadrant of the cycle (upper-left quadrant).

The next step in the cycle is to evaluate these different alternatives based on the objectives and constraints. The focus of evaluation in this step is based on the risk perception for the project.

The next step is to develop strategies that solve uncertainties and risks. This step may involve activities such as benchmarking, simulation, and prototyping.

V-Model

In this type of SDLC model testing and the development, the step is planned in parallel. So, there are verification phases on the side and the validation phase on the other side. V-Model joins by Coding phase.

Incremental Model

The incremental model is not a separate model. It is necessarily a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC process is repeated, with each release adding more functionality until all requirements are met. In this method, each cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.

Agile Model

The Agile Model is an incremental and iterative process of software development. It defines each iteration's number, duration, and scope in advance. Every iteration is considered a short "frame" in the Agile process model, which mostly lasts from two to four weeks.

Agile Model divides tasks into time boxes to provide specific functionality for the release. Each build is incremental in terms of functionality, with the final build containing all the attributes. The division of the entire project into small parts helps minimize the project risk and the overall project delivery time.

Iterative Model

It is a particular implementation of a software development life cycle that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete. In short, iterative development is a way of breaking down the software development of a large application into smaller pieces.

Big bang model

Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come.

This model works best for small projects with smaller size development team which are working together. It is also useful for academic software development projects. It is an ideal model where requirements are either unknown or final release date is not given.

Prototype Model

The prototyping model starts with the requirements gathering. The developer and the user meet and define the purpose of the software, identify the needs, etc.

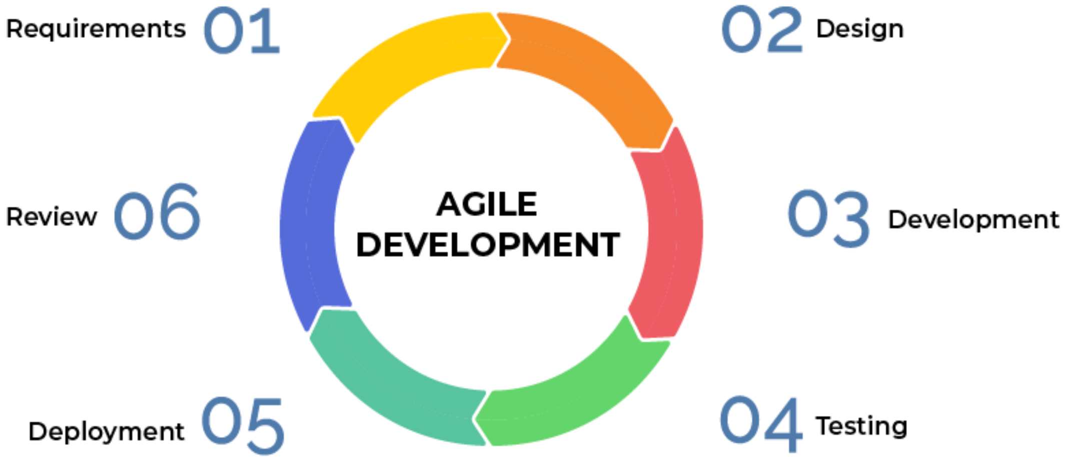
A '**quick design**' is then created. This design focuses on those aspects of the software that will be visible to the user. It then leads to the development of a prototype. The customer then checks the prototype, and any modifications or changes that are needed are made to the prototype.

Looping takes place in this step, and better versions of the prototype are created. These are continuously shown to the user so that any new changes can be updated in the prototype. This process continue until the customer is satisfied with the system. Once a user is satisfied, the prototype is converted to the actual system with all considerations for quality and security.

2. The suitable SDLC Model for the specific Application Development

Agile is based on the adaptive software development methods, whereas the traditional SDLC models like the waterfall model is based on a predictive approach. Predictive teams in the traditional SDLC models usually work with detailed planning and have a complete forecast of the exact tasks and features to be delivered in the next few months or during the product life cycle. Unlike the traditional approaches of SDLC, Agile approaches are precise and customer friendly. Users/Customers have the opportunity to make modifications throughout project development phases

Agile uses an adaptive approach where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed.



3. Phases of Agile Model

3.1 Requirements:

- In Agile, requirements are captured as user stories or features. These are short, specific descriptions of functionality that provide value to the customer.

- User stories are typically written by the product owner in collaboration with stakeholders.

3.2 Design:

- In Agile, design is an ongoing activity that happens in parallel with development. It focuses on creating just enough design to support the current set of user stories.

- Detailed design work is often deferred until it's necessary to avoid over-engineering.

3.3 Development or Iteration:

The third phase of the agile development process is development or iteration. This one is the longest of all as it carries all the bulk work. Now the development team will start working on combining all the requirements of the product gathered in the concept and inception phase. It goes under several reviews and revisions for improvement until finalized.

In the development or iteration phase, the following steps are accomplished:

- The association of the team with clients.
- Iterations and functionalities are prioritized and implemented.
- Each sprint/iteration should be closely examined and developed.
- Delivering regular working software releases.
- Ensuring product quality by testing at regular intervals.

3.4 Testing:

- Testing in Agile is continuous and integrated throughout the development process.

- Automated testing is emphasized to ensure that code changes don't introduce new bugs. Manual testing is also performed, particularly for exploratory and usability testing.

3.5 Deployment:

- Continuous Deployment (CD) is often a key practice in Agile. It means that code changes are automatically deployed to production as soon as they pass automated tests.

- This ensures that working software is consistently delivered to users.

3.6 Review

- Reviews in Agile are part of the sprint cadence. At the end of each iteration/sprint, the team holds a sprint review meeting to demonstrate the completed user stories to stakeholders.

- This is an opportunity for stakeholders to provide feedback on the delivered features.

In Agile, these phases are not strictly linear, and they often overlap. The key is to continuously iterate and refine the product. Additionally, Agile encourages adaptability and responsiveness to changing requirements, so adjustments can be made at any point in the process.

4. Why choose Agile Model

Agile development is important because it helps to ensure that development teams complete projects on time and within budget. It also helps to improve communication between the development team and the product owner. Additionally, Agile development methodology can help reduce the risks associated with complex projects.

GStream is a live streaming platform focused on gaming and esports content. For such an application, a flexible and iterative approach like Agile, with specific attention to Continuous Deployment, would be well-suited. Here's why:

1. Rapid Feature Delivery: Agile emphasizes delivering small, incremental updates frequently. This aligns with the need to continuously introduce new features, enhancements, and improvements in real-time to keep viewers and streamers engaged on platforms like G Stream..

2. User-Centric Development: Agile frameworks encourage close collaboration with users. Regular feedback from streamers and viewers can help shape the platform to better meet their needs and preferences, which is crucial for the success of a live streaming service.

3. Suitable for Bigger Project: Agile methodologies allow teams to respond to changes and new requirements quickly and easily. This is particularly important for large-scale projects, which often involve multiple stakeholders with different priorities and needs.

4. Flexibility and Adaptability: The gaming and esports industry is dynamic, with rapidly changing trends and user expectations. Agile allows you to adapt quickly to market shifts and emerging technologies, ensuring that the platform remains competitive.

5. Continuous Improvement: Agile encourages continuous improvement through retrospectives. The development team can reflect on their processes and practices to make ongoing refinements and optimizations.

6. Continuous Integration and Deployment (CI/CD): For a platform that needs to be constantly updated and improved, CI/CD is critical. Agile methodologies, especially when coupled with CI/CD pipelines, enable automated testing and deployment, reducing manual errors and speeding up the release process.

7. Time-to-Market: Agile, along with CI/CD, allows for faster time-to-market. New features and bug fixes can be deployed quickly, ensuring that the platform remains competitive and responsive to user needs.

8. Risk Management: Agile enables early detection and mitigation of risks through iterative development and continuous testing. This is important for a platform like GStream, which requires high levels of reliability and stability.

9. Transparency and Communication: Agile methodologies promote transparency through regular ceremonies like stand-up meetings, sprint reviews, and retrospectives. This ensures that all stakeholders have a clear view of progress and challenges.

10. Cross-Functional Teams: Agile teams are typically cross-functional, including members with skills in development, testing, design, and more. This is essential for building and maintaining a comprehensive platform like GStream, which requires expertise in multiple domains.

5. Contact

By Phone:

+8801828867270

Contact Us by Email

HDsector@gmail.com

On our website

www.hdsector.bd