NAME: IENGEM GAADI

SOFTWARE DEVELOPMENT LIFE CYCLE

1. Planning

In the planning phase, programmers evaluate the terms of the project. This includes calculating labor and material costs, creating a timetable with target goals and creating the project's teams and leadership structure.

Planning can also include feedback from stakeholders. Stakeholders are anyone who stands to benefit from the application. Try to get feedback from potential customers, developers, subject matter experts, and sales reps.

The programme requirements should clearly define the scope and purpose of the application. It plots the course and provisions the team to effectively create the software. It also

sets boundaries to help keep the project from expanding or shifting from its original purpose.

2. Define Requirements

Defining requirements is considered part of planning to determine what the application is supposed to do and its requirements. For example, a social media application would require the ability to connect with a friend. An inventory program might require a search feature.Requirements also include defining the resources needed to build the project. For example, a team might develop software to control a custom manufacturing machine. The machine is a requirement in the process.

3. Design

The Design phase models the way a software application will work. Some aspects of the design include:

Architecture – Specifies programming language, industry practices, overall design, and use of any templates or boilerplate

User Interface – Defines the ways customers interact with the software, and how the software responds to input

Platforms – Defines the platforms on which the software will run, such as Apple, Android, Windows version, Linux, or even gaming consoles

Programming – Not just the programming language, but including methods of solving problems and performing tasks in the application

Communications – Defines the methods that the application can communicate with other assets, such as a central server or other instances of the application

Security – Defines the measures taken to secure the application, and may include SSL traffic encryption, password protection, and secure storage of user credentials

Prototyping can be a part of the Design phase. A prototype is like one of the early versions of software in the Iterative software development model. It demonstrates a basic idea of how the application looks and works. This “hands-on” design can be shown to stakeholders. Use feedback o improve the application. It’s less expensive to change the Prototype phase than to rewrite code to make a change in the Development phase.

4. Implementation

This is the actual writing of the program. A small project might be written by a single developer, while a large project might be broken up and worked by several teams. Use an Access Control or Source Code Management application in this phase. These systems help developers track changes to the code. They also help ensure compatibility between different team projects and to make sure target goals are being met.

The coding process includes many other tasks. Many developers need to brush up on skills or work as a team.

Finding and fixing errors and glitches is critical. Tasks often hold up the development process, such as waiting for test results or compiling code so an application can run. SDLC can anticipate these delays so that developers can be tasked with other duties.

5. Documentation

Software developers appreciate instructions and explanations. Documentation can be a formal process, including wiring a user guide for the application. It can also be informal, like comments in the source code that explain why a developer used a certain procedure. Even companies that strive to create software that's easy and intuitive benefit from the documentation.

Documentation can be a quick guided tour of the application's basic features that display on the first launch. It can be video tutorials for complex tasks. Written documentation like user guides, troubleshooting guides, and FAQ's help users solve problems or technical questions.

6. Testing

It’s critical to test an application before making it available to users. Much of the testing can be automated like security testing. Other testing can only be done in a specific environment – consider creating a simulated production environment for complex deployments. Testing should ensure that each function works correctly. Different parts of the application should also be tested to work seamlessly together—performance test, to reduce any hangs or lags in processing. The testing phase helps reduce the number of bugs and glitches that users encounter. This leads to a higher user satisfaction and a better usage rate.

7. Deployment

In the deployment phase, the application is made available to users. Many companies prefer to automate the deployment phase. This can be as simple as a payment portal and download link on the company website. It could also be downloading an application on a smartphone.

Deployment can also be complex. Upgrading a company-wide database to a newly-developed application is one example. Because there are several other systems used by the database, integrating the upgrade can take more time and effort. Even after the initial launch of your app, you'll need to maintain some staff on hand to handle any customer complaints and to rollout occasional fixes and updates.

Updates are larger changes to your app's code or programming and should be undertaken after collecting a bunch of similar user feedback.

Upon collecting that feedback, you can basically restart the app development cycle again – come up with a solution for problems people are experiencing with your app, wireframe that solution, test it with a prototype version of the live app, then build in the fix and deploy it to live users.

As you can see, the application development cycle never really ends. But this also ensures that your app will be as effective and functional as possible.

8. Operations and Maintenance

At this point, the development cycle is almost finished. The application is done and being used in the field. The Operation and Maintenance phase is still important, though. In this phase, users discover bugs that weren't found during testing. These errors need to be resolved, which can spawn new development cycles.

In addition to bug fixes, models like Iterative development plan additional features in future releases. For each new release, a new Development Cycle can be launched.

SDLC Models & Methodologies Explained

Waterfall

The Waterfall SDLC model is the classic method of development. As each phase completes, the project spills

over into the next step. This is a tried-and-tested model, and it works. One advantage of the Waterfall model is each phase can be evaluated for continuity and feasibility before moving on. It's limited in speed, however, since one phase must finish before another can begin.

Agile

The AGILE model was designed by developers to put customer needs first. This method focuses strongly on user experience and input. This solves much of the problems of older applications that were arcane and cumbersome to use. Plus, it makes the software highly responsive to customer feedback. Agile seeks to release software cycles quickly, to respond to a changing market. This requires a strong team with excellent communication. It can also lead to a project going off-track by relying too heavily on customer feedback.

Iterative

In the Iterative development model, developers create an initial basic version of the software quickly. Then they review and improve on the application in small steps (or iterations). This approach is most often used in very large applications. It can get an application up and functional quickly to meet a business need. However, this process can exceed its scope quickly and risks using unplanned resources.

DevOps

The Devops security model incorporates operations – the people who use the software – into the development cycle. Like Agile, this seeks to improve the usability and relevance of applications. One significant advantage of this model is the feedback from actual software users on the design and implementation steps. One drawback is that it requires active collaboration and communication. Those additional costs can be offset by automating parts of the development process.

Other Models

Many other SDLC models are essentially a variant of these core processes. Organizations use LEAN manufacturing processes for software development. V-shaped development is a type of Waterfall that implements testing, verification, and validation. Spiral development may pick and choose models for each step in the development process.

Benefits of SDLC

SDLC provides a number of advantages to development teams that implement it correctly.

Clear Goal Descriptions

Developers clearly know the goals they need to meet and the deliverables they must achieve by a set timeline, lowering the risk of time and resources being wasted.

Proper Testing Before Installation

SDLC models implement checks and balances to ensure that all software is tested before being installed in greater source code.

Clear Stage Progression

Developers can’t move on to the next age until the prior one is completed and signed off by a manager.

Member Flexibility

Since SDLCs have well-structured documents for project goals and methodologies, team members can leave and be replaced by new members relatively painlessly.

Perfection Is Achievable

All SDLC stages are meant to feed back into one another. SDLC models can therefore help projects to iterate and improve upon themselves over and over until essentially perfect.

No One Member Makes or Breaks the Project

Again, since SDLCs utilize extensive paperwork and guideline documents, it's a team effort and losing even one major member will not jeopardize the project timeline.

Best Practices Of Software Development

In addition to the models and stages of software development, there are a few other helpful practices. These can be applied to part or all of the development cycle.

Source Control

Source Control is a security plan to secure your working code. Implement Source Control by keeping the code in a single location, with secure and logged access. This could be a physical location where files are stored and accessed in a single room in the building. It could also be a virtual space where users can log in with an encrypted connection to a cloud-based development environment.

Source Control applications include a change management system to track work done by individuals or teams. As with any storage, use a backup system to record development progress in case of a disaster.

Continuous Integration

Continuous Integration evolved out of a case of what not to do. CI works to make sure each component is compatible through the whole development cycle. Before CI, different teams would build their own projects independently. This created significant challenges at the end when developers stitched the application together. Continuous Integration

ensures all teams use similar programming languages and libraries, and helps prevent conflicts and duplicated work.

SDLC Management Systems

A software development cycle management system works to control and manage each step of the development cycle. Management Systems add transparency to each phase and the project as a whole. They also add analytics, bug-tracking, and work management systems. These metrics can be used to improve parts of the cycle that aren't running efficiently.

Conclusion: The Process for Software Development

SDLC shows you what's happening, and exactly where your development process can improve.

Like many business processes, SDLC aims to analyze and improve the process of creating software. It creates a scalable

view of the project, from day-to-day coding to managing production dates.