Software Engineering Task 1 2024

Software Development Life Cycle (SDLC)

Q: Explain the different phases of SDLC (e.g., requirements analysis, design, implementation, testing, maintenance).

While there are various methods for software development, an SDLC assures each stage yields the highest quality results. The 7 different generalised phases are planning/requirements analysis, design, implementation, testing, deployment, maintenance and evaluation.

Planning/Requirements Analysis

During this stage, the project lead works with the client to shape a detailed picture of the product (Almeida, 2023) and its end goals. This phase leads to the eventual estimation of the costs and deadlines for a project and also a complete list of software specifications made in accordance with the requirements of the application.

Design

In this phase, the software engineers involved come up with the best possible software solutions after consulting the requirements and specifications produced in the stage that occurs prior. It is within this stage that the decisions to use any pre-existing modules or technologies are made and the identification of necessary tools (*What Is SDLC? - Software Development Lifecycle Explained*, n.d.) takes place.

Implementation

This phase is where the actual programming starts and a codebase is populated by a singular developer or a team, depending on the project size. Developers in this phase use a set of explicit guidelines to build the actual application. Efficiency is a key aspect of this stage as teams dedicate themselves to producing a functional and coherent application.

Testing/Integration

Often occurring in tandem with the previous phase, this stage involves application of various testing methods, including unit testing and integration testing to guarantee the quality of

the code and validate its security and performance. When done parallel to the implementation phase, code can be written, tested for bugs and integrated with the application, after a certain sprint or time block.

Deployment

Once all bugs have been eliminated from an application, the deployment phase begins. While making sure that the software utilised is up-to-date (*What Is SDLC? - Software Development Lifecycle Explained*, n.d.) and secure enough for real-world use, the deployment team transfers it from the development environment to a live one, which is often an app store or an online platform.

Maintenance

Once the software that was meticulously developed using the above phases of the SDLC framework is in active use by a client, the ongoing maintenance stage is commenced. This phase involves the resolving of customer issues, system updates and any other post-deployment bugs. User behaviour in regards to the application is steadily monitored to help identify ways to refine the application in future revisions.

Evaluation

The evaluation phase of the SDLC builds on the maintenance stage, by determining if the project met cost and timeline objectives. In this phase, members of the development team look to document what worked in the development process, taking into consideration any mistakes made, to improve on in future development projects.

Software Engineering Task 1 2024

Q: Discuss why following an SDLC is essential for successful software development.

Although unstructured development approaches seem more enticing in the short term, following an SLDC ends in the smooth production of high quality software. The use of an SDLC enables efficient project management, by providing clarity to both the developers, and the stakeholders overlooking the project.

The planning, requirements analysis and design stages plot a clear path for the development of an application, giving the development team a clear understanding of business requirements in the implementation phase. Furthermore, an SDLC ensures product quality through rigorous and systematic testing in the testing phase, including unit, integration and acceptance testing routines. This in turn, allows for the early detection (Bhatt, n.d.) of an issues in an application, giving developers ample opportunity to handle them before the application is sent out to market.

Finally, the gradual progression followed by any SDLC, especially the Agile family of approaches, enables a higher level of flexibility and adaptability (Bhatt, n.d.) in the development of an application. Not only does following an SDLC mean that a project evolves continuously, in iterations, or ‘sprints’, but it also mitigates any risks and increases customer fulfilment in doing so. As an example, a large banking corporation, looking for an application for their users to interact with, would need to follow an SDLC to make certain that their high performance and reliability requirements are met by the eventual result, and that there are no errors present.

By utilising an SDLC, a smoother development process will take place, where the executives of the banking corporation are well-informed with occurrences at each. All in all, following an SDLC intensifies product quality, adds to flexibility in the development process and enhances project management.

Software Engineering Task 1 2024

Programming Paradigms

Q: Describe imperative, declarative, and object-oriented programming paradigms.

Imperative

The imperative paradigm facilitates the use of a procedural way to execute programming logic. Relying on control flow to implement programs, this approach applies well-defined instructions (Shakya, 2023) to yield end results. A weather application's pseudocode is written like so in this paradigm;

1. TAKE location input
2. CALL current weather API
3. PARSE JSON from API
4. RETURN relevant aspects of JSON
5. DISPLAY information from JSON

Declarative

Declarative programming is carried out in a manner where the expected end-result of the program is described beforehand. A program based on this paradigm is carried out by the programming language's implementation and compiler (Chang, 2022) to achieve the desired outcome. After adapting the above weather application to this approach, the pseudocode is written like so:

1. TAKE location input
2. RECEIVE data from weather API
3. DISPLAY data

Object-Oriented

The object-oriented approach shapes code into reusable 'blueprints' known as classes (Doherty, 2024) to be instantiated later on. Classes are composed of attributes and methods - information fields and operations to perform with them - abstracting a significant portion of logic. A program defining a dog class with attributes for its name, breed and birthday and a method that calculates its age using the birthday and the current date adopts this paradigm.

Q: Explain how Python supports these paradigms.

Python, being a multi-paradigm language, supports all three paradigms without enforcing the use of any one in particular.

Its control flow features of conditionals (if, elif and else) and loops (for and while) allow developers to achieve tasks in a step-by-step format. For example, implementation of a program to calculate the sum of a list of integers in Python, utilising the imperative approach, works with a for loop that iterates through the list and adds each integer to a predefined variable.

Python also facilitates the declarative paradigm by providing the programmer with predefined functions to execute programs. The sum() function in Python is an example of such a function, allowing for the addition of a list of integers without using the procedural instructions defined above, all in one function call.

Python also adapts to an object-oriented paradigm by supporting the creation and modification of classes. The ‘class’ keyword creates a class, whose attributes and methods are defined using the ‘self’ keyword. A programmer could therefore implement a custom class of ‘Student’ with attributes of ‘name’ and ‘year’ using these keywords in Python.

Software Engineering Task 1 2024

Software Testing

Q: Define unit testing, integration testing, and acceptance testing.

Unit Testing

Unit testing assesses the functionality of a small, isolated block of code (*What Is Unit Testing? - Unit Testing Explained*, n.d.) which is normally a function or a method. A unit test written for a function that adds two numbers, e.g multiply(a, b) will execute the function with multiple test values for a and b to evaluate its correctness. This form of testing is essential at a low-level.

Integration Testing

Integration testing occurs once individual code blocks pass the unit test phase, ready to be integrated with other units. Any communication issues between separate software modules (*What Is Integration Testing? Definition, Examples, How-To*, n.d.) are highlighted. For example, in Big Bang integration testing, all system modules are tested as one to validate the cohesion of an application. Overall, integration testing is vital for assuring functionality between two or more units of a software system.

Acceptance Testing

Acceptance testing tests an application against previously set business requirements of a client (Jain, 2024), including the functional (technical) and non-functional (design-related) requirements decided in the beginning. Examples of such testing methods are User Acceptance Testing (UAT) and Business Acceptance Testing (BAT), which assess an application's compliance with user and client needs. In doing so, acceptance testing makes sure that an application meets initially defined requirements criteria.

Software Engineering Task 1 2024

Q: Discuss the importance of testing in software development.

Software testing is a vital stage in the production of software, allowing for the timely detection of issues in code and resulting in a flawless final product. Testing is a broad term containing multiple different processes, including unit testing, integration testing and acceptance testing, which ensure the validation of an application's functionality from the simplest components to more complex ones.

Occurring initially, unit testing warrants the smaller functions are checked before integration, for example, a cost function defined as 'def calculate_cost(product)' would have to be verified with a unit test, to make sure that when deployed, the correct prices are shown. Integration testing, by detecting communication issues between modules, serves as a crucial stage in producing coherent and well integrated code.

In a shopping application, if another function, 'check_availability', calls the 'calculate_cost' function, an associated integration testing scheme would evaluate that the right data types are being used in the function call (integers, in this case) and that the interaction between the two functions is as expected. The most important aspect of the testing phase is acceptance testing, which would thoroughly validate the finished shopping application and make sure that it fulfils what it was intended to do, i.e allow shoppers to browse products and sellers to see sales.

Generally, software testing is extremely important in any software development project as through various phases of testing at different levels, it results in a polished and high quality application.

Version Control

Q: Explain the purpose of version control systems (e.g., Git).

The purpose of version control systems is to track and manage (*What Is Version Control*, n.d.) changes made to an application's source code over time. Version control systems have become increasingly popular in fast-paced development environments to aid software developers in dealing with mistakes made by 'turning back the clock' (*What Is Version Control*, n.d.) to past versions of application code. Many such systems, like Git, also encourage collaboration amongst development teams by storing elaborate histories of each member's changes to source code, so multiple developers can write to the same Git repository. In all simplicity, version control serves as a back-up for previously written source code to allow for continued development without fearing irreversible changes to code.

Q: Describe how version control helps manage code changes.

All version control systems have features that aid in the management of changing source code. A long term change history (*What Is Version Control*, n.d.) for each file in a project monitors its creations, deletion and modifications, allowing developers to review older, conflicting versions of code to improve code quality. VCSs also cater to collaborative workflows with independent and merge-able branches to enable error-free development. For example, if a team using VCS technology like Git encounters errors in their code, they have the ability to revert back and make appropriate changes to the code base. Similarly, if the developers worked from different parts of the world, Git will verify that each person's contributions exist on multiple branches. By utilising an overall change history and the concept of branches, version control effectively manages changes made to an application's code base.

References

Almeida, J. (2023, July 28). *SDLC Guide: Key Stages and Models in Software Development*. DistantJob. Retrieved March 3, 2024, from https://distantjob.com/blog/software-development-life-cycle-stages-and-models/

Bhatt, T. (n.d.). Wikipedia. Retrieved March 25, 2024, from https://www.intelivita.com/au/blog/benefits-of-software-development-life-cycle/

Chang, D. (2022, July 18). *Declarative vs imperative programming: 5 key differences*. Educative.io. Retrieved March 10, 2024, from https://www.educative.io/blog/declarative-vs-imperative-programming

Doherty, E. (2024, January 24). *What is object-oriented programming? OOP explained in depth*. Educative.io. Retrieved March 10, 2024, from https://www.educative.io/blog/object-oriented-programming

Jain, S. (2024, January 8). *Acceptance Testing - Software Testing*. GeeksforGeeks. Retrieved March 4, 2024, from https://www.geeksforgeeks.org/acceptance-testing-software-testing/

Shakya, U. (2023, August 24). YouTube: Home. Retrieved March 10, 2024, from https://programiz.pro/resources/imperative-vs-declarative-programming/

What is Integration Testing? Definition, Examples, How-to. (n.d.). Katalon. Retrieved March 4, 2024, from https://katalon.com/resources-center/blog/integration-testing

What is SDLC? - Software Development Lifecycle Explained. (n.d.). AWS. Retrieved March 16, 2024, from https://aws.amazon.com/what-is/sdlc/

What is Unit Testing? - Unit Testing Explained. (n.d.). AWS. Retrieved March 4, 2024, from https://aws.amazon.com/what-is/unit-testing/

What is version control. (n.d.). Atlassian. Retrieved March 11, 2024, from https://www.atlassian.com/git/tutorials/what-is-version-control