Software development one - Design

Riyazdeen Abdul Waajid

5th of October 2024

Contents

1 SDLC/Dev - Software development life cycle

1.0.1 what is the Software Development life cycle?

It is a systematic process or set of phases that guide the development of software applications from conception to deployment and maintenance.

In simple words it is to make an idea to manifest into reality, this idea could also be the solution to a problem. This idea/solution will take its form over multiple phases. This solution, which is the end product must also be disributable, and used by end users.

These phases are the standard set of steps to be followed when developing a software.

The phases in the SDLC are:

1. Analysis phase
2. Design phase
3. Implementation
4. Testing
5. Deployment
6. Maintenance

The software development process is referred to as a cycle rather than a sequence because it is often iterative and repetitive. In a sequence, each step would occur once in a linear order, but in a cycle, the process loops back on itself, allowing for continuous improvement, refinement of the software.

1.1 Analysis phase - requirement gathering and analysis

Identify and understand the needs, and the requirements of the software. Within this phase, the problem is to be identified, the requirements are to be identified, and then an analysis of the problem and the requirements are to be made. Before starting any new task or development of a new software it is of utmost priority for the developer or anyone who undertakes development to understand the exact requirements that are demanded by the task.

To gather the requirements the developer must use elicitation methods, such as:

- observations
- interviews
- surveys
- research into literature

Elicitation methods are techniques used to gather, extract, and understand requirements from stakeholders (such as clients, users, and subject matter experts) during the early stages of a project. These methods help ensure that the software or system being developed aligns with the needs and expectations of those who will use or benefit from it. Elicitation methods are crucial in business analysis, system design, and project management to accurately capture functional and non-functional requirements.

Next in this phase, **the project scope must be defined, followed by documenting the functional and non-functional requirements.**

Defining the project scope establishes clear boundaries for the team, outlining the specific criteria that need to be fulfilled. Everything within the scope represents what the system and developers are responsible for. Conversely, any requirements outside this scope are not part of the project and are not the developers' responsibility. This ensures that expectations are managed and the team focuses only on the agreed-upon deliverables.

1.1.1 Functional Requirements and Non-functional requirements

Functional requirements

Functional requirements describe the specific behaviors or functions that a system must perform. The focus is on the actions and features the product will deliver. These requirements define what the system should do to fulfill its intended purpose.

A common approach for gathering and prioritizing functional requirements is based on the IEEE standards, often represented using the MoSCoW method. This format categorizes requirements based on their importance and necessity.

In the MoSCoW format, the requirements are grouped into the following categories:

1. Must: Essential features the system is required to have.
2. Should: Important features that are not critical but highly desirable.
3. Could: Features that are nice to have if time and resources permit.
4. Won't (or Would): Features that are not considered at this stage but might be included in the future.

This approach helps prioritize the most critical requirements, ensuring that the system focuses on delivering what is absolutely necessary, while also considering enhancements when possible.

Examples:

(a) The user should be able to log into the dashboard using biometric credentials.

Extra notes from gpt starts here!

Some functional requirements for a library system:

(1) User Registration and Authentication

- The system must allow users to create an account by providing personal details (e.g., name, email, ID).
- The system should provide a password recovery option.
- User Registration and Authentication Users must be able to log in using their credentials (username/password or biometric login).

(2) Book Search and Retrieval The system should display the availability status (available, checked out, reserved).

- Users must be able to search for books by title, author, genre, ISBN, or keywords.
- Users should be able to filter and sort search results by criteria like publication date or popularity.

(3) Book Borrowing The system must track the due dates for borrowed items.

- Registered users must be able to borrow books by placing a hold on available items.
- Users should be able to extend loan periods if the book is not reserved by another user.

(4) Book Reservation

- Users must be able to reserve books that are currently checked out.
- The system should notify users when a reserved book becomes available for pickup.

(5) Notifications and Alerts

- The system must notify users about overdue books via email or SMS.
- The system should send reminders for upcoming due dates.
- Users should receive alerts when reserved books are available for pickup.

(6) Catalog Management (For Admins/Librarians)

- Librarians must be able to add, update, or remove books from the system catalog.
- The system should support bulk uploading of book data through CSV files or other formats.
- The system must allow the classification of books by genres, authors, and categories.

(7) Fine Calculation and Payment

- The system must calculate overdue fines based on the number of late days.
- Users must be able to view their fine history and pay fines online.

(8) User Account Management

- Users must be able to view their borrowing history.
- The system must allow users to update their personal information.
- Users should be able to view current borrowed books, due dates, and reservation statuses.

(9) Library Hours and Information

- The system should display library hours, upcoming events, and holidays.
- Users should be able to view the locations and contact information for different library branches.

(10) Reports and Analytics (For Admins)

- The system must generate reports on book circulation, user activity, and overdue items.
- Librarians should be able to view analytics on popular books, genres, and user borrowing patterns.

Extra notes from gpt end here!

Non-functional requirement

Non-functional requirement specifies the quality attributes or the constrains the system must satisfy.

Non-functional requirements (NFRs) specify the criteria that can be used to judge the operation of a system, rather than specific behaviors. These requirements define how a system performs its functions rather than what functions it performs. They encompass various quality attributes and constraints that affect the user experience, performance, and overall system characteristics.

Meaning of Non-Functional Requirements

Non-functional requirements describe the quality attributes of a system, such as:

- Performance: How quickly the system responds to user actions.
- Usability: How easy and intuitive the system is for users.
- Reliability: The system's ability to function under specific conditions without failure.
- Scalability: The capacity of the system to handle increased loads or expand in capability.

- Security: Measures to protect the system from unauthorized access or breaches.
- Maintainability: The ease with which the system can be modified or updated.
- Avalibility: The accessibility of the system to the users over a period of time.

Examples of Non-Functional Requirements:

1. Performance Requirements
    - The system should respond to user queries within 2 seconds under normal load conditions.
    - The system must support 100 concurrent users without performance degradation.
2. Usability Requirements
    - The user interface should be intuitive and easy to navigate, requiring no more than three clicks to access any feature.
    - The system must provide accessibility features for users with disabilities (e.g., screen reader compatibility).
3. Reliability Requirements
    - The system should have an uptime of 99.9
    - The system must recover from failures within 5 minutes.
4. Scalability Requirements
    - The system should be able to scale horizontally by adding new servers to handle increased user load.
    - The application should support a 50
5. Security Requirements
    - All user data must be encrypted both at rest and in transit.
    - The system must implement multi-factor authentication for user logins.
6. Maintainability Requirements
    - The system should allow for easy integration of new features with minimal disruption to existing functionality.
    - The codebase must be documented in compliance with established coding standards.
7. Compatibility Requirements
    - The application should be compatible with the latest versions of major web browsers (e.g., Chrome, Firefox, Safari).
    - The system must integrate seamlessly with existing internal systems and databases.
8. Legal and Regulatory Compliance
    - All user data handling must adhere to industry standards for privacy and security.

Importance of Non-Functional Requirements

(a). User Experience:

They significantly impact how users perceive the quality and reliability of the system.

(b). System Quality:

NFRs help ensure that the system meets the desired performance, usability, and security standards.

(c). Risk Management:

Addressing NFRs early in development can reduce risks associated with performance bottlenecks, security vulnerabilities, and user dissatisfaction.

(d). Long-term Viability:

They help ensure that the system remains maintainable and adaptable over time as technology and user needs evolve.

In summary, non-functional requirements are critical for the overall success of a system, influencing how well it meets user expectations and performs under various conditions.

Normally, developers use use case diagrams to represent the functional requirements of a system

Use case diagrams are simple visual tools that show how users interact with a system. They help explain what the system can do from the user's point of view. The main parts of a use case diagram are actors, which are the users or things that use the system, and use cases, which are the actions they can take, like searching for a book, borrowing a book, or signing up as a new user. The system boundary is a box that shows what is included in the system. For example, in a library management system, the actors could be a "User" and a "Librarian," while the use cases might include searching for books and returning borrowed ones. Use case diagrams are useful because they help everyone understand how users will interact with the system, making it easier to plan and develop the needed features.

1.1.2 Feasibility study

A feasibility study is an analysis conducted to determine the viability of a proposed project or solution. It evaluates whether the project is practical and achievable within a given timeframe, budget, and resource constraints. The goal of a feasibility study is to assess various aspects of a project to help stakeholders make informed decisions about whether to proceed, modify, or abandon the idea.

Key Components of a Feasibility Study

1. Technical Feasibility:

   Assesses whether the technology and resources needed to implement the project are available and whether the project can be developed using existing technology. If the databases that are required exist, that the languages that exist can meet the requirements, and all the other technical factors must be considered in accordance to the requirements.

2. Economic Feasibility:

   Evaluates the cost-effectiveness of the project. This includes an analysis of the expected costs, potential financial benefits, and overall return on investment.

   (a) Expenditure

      i. Direct expenditure

         Example: Such as the salaries of the developers of a company, the price of the hardware

      ii. Indirect expenditure

         Example: When an newly released software's servers crash due to the unexpectedly heavy traffic.

   (b) Revenue

      i. Direct revenue

         Example: such as the product sales, service fees, subscription fees, etc.

ii. Indirect revenue

Example: Saving time

3. Operational Feasibility:

   Examines how well the proposed project aligns with the organization's existing operations and whether it can be integrated into current processes.

   Example: An organization wants to shift from recording data on paper to using a computer system. However, the software developed by the team is too complex and difficult to navigate, with many unnecessary features. This complexity can overwhelm the employees, who are not accustomed to using complicated software. In this case, the software is not operationally feasible. For individuals who are not very familiar with computers, the developer must design the software to be easy to use with a small learning curve, ensuring it fits smoothly into their workflow.

4. Legal Feasibility:

   Determines if the project complies with legal and regulatory requirements, including permits, licenses, and environmental regulations.

5. Organisational Feasibility:

   Refers to the assessment of whether an organization has the necessary resources, structure, skills, and culture to successfully implement a proposed project. It focuses on evaluating internal capabilities to determine if the organization can support the development, deployment, and operation of the project. This type of feasibility is critical in ensuring that the project aligns with the organization's goals and that the necessary human, technical, and management resources are available

Importance of a Feasibility Study

1. Informed Decision-Making: Helps stakeholders understand the potential challenges and benefits, allowing for more informed decisions.

2. Risk Mitigation: Identifies potential risks and obstacles early in the planning process, enabling the team to address them proactively.

3. Resource Allocation: Provides insight into the resources required for the project, helping organizations allocate their time, money, and personnel effectively.

4. Stakeholder Confidence: Demonstrates due diligence to stakeholders and investors, increasing their confidence in the project.

Conclusion

In summary, a feasibility study is an essential step in project planning that assesses various factors to determine whether a project is worth pursuing. By thoroughly analyzing the technical, economic, operational, legal, and scheduling aspects, organizations can make better-informed decisions that contribute to successful project outcomes.

At the end of the first phase, after completing all the above steps, it will result in an document called an SRS document also know as the Software Requirement Specification. This is the outcome of your first phase, after the requirement gathering and the feasibility analysis.

1.2 System design - Design phase.

the second phase is the System design. In this phase all theu architects and the senior members in a development team will refer to the SRS and then come up with a design for the software inaccordance to the requirement.

The SRS (Software requirement specification) is given to the architects and senior managers who will the be creating the high level design that outlines the system architecture and component

This is the most important phase as this is where the developer develops the design for the high level code of the software.

Create a high level desing that outlines the system architecture and components. Define data structures, interfaces, algorithms and overall system behaviour.

The tools that are used:

- UML diagrams
  - Class Diagrams
  - Sequence Diagrams
  - ER Diagrams
  - DFD
  - Flow charts

at the end of this phase the developer will complete an document called the Software Design Document.

1.3 Implementation phase - coding

Write and develop the code base upon the design specifications (the SDD at the end of phase two). Then perform unit testing to ensure each individual component work as intended.

This design is transformed into an executable code block. Thus the developer can run the program or software and check the functionalities for any errors or malfunctions.

Inorder to fulfill the requirements and produce a functional code block that meets all the requirements, in this phase all the developers are involved as well as the senior developers and the team leads, and more.

At the end of this level the developers will have an executable code, a manual, and not a deployable product. This manual will teach the end user on how to use the system or the program.

1.4 Testing phase

Conduct various testing, phases such as integration testing, system testing, and user acceptance testing. Identify and fix defects or issues found during testing.

1.4.1 The three types of testing

1. White box testing:

   The testers are allowed to see the internal code structures. In this type of testing the main advantage is that even if the code does function and produce the correct output, the testers can see the internal structures and check for errors such as code redundency, or wrong output for some of the internal functoins of the code. Usually the developers from the team that originally developed the code for the software might be present.

2. Black box testing:

   In black box testing the internal code or the structure is unkown to the tester who is testing the code. Where the tester compares the output of the system to the expected output after inserting an input. If the expected output and the actual output are not exaclty the same then that means that there is an error or an bug in the software or the system. The tester or the quality assurance team are not provided with any documents that explain the system or any type of access to the internal code.

   Example: automated testing can be incorperated to perform black box testing.

3. Gray box testing :

   This is where the tester has partial access to the code blocks and internal code.

The most important of these two types of testing is the white box and black box testing.

Within the software testing phase there is another life cycle.

1.4.2 Software Testing Life Cycle (STLCL)

1. Requirement analysis
2. Test planing
3. Test case design and development
4. Test enviroment setup
5. Test execution
6. Test cycle closure

inside a life cycle you have anothe life cycle, software testing life cycled.

each test case contain multiple parameters.

there are 6 phases in this software life cycle.

1. Requirement

Testing Phases.

1. Aceptance testin- which is divided into two parts. 1. you have an involment with the development team, where an member from a different team from the development team.

2. system testing- a black box or a white box 3. intergration testing is trying to integrate the components together and to verify that all these components are working well. done by the develepor 4. Unit testing - done by the develepors them selves 5.

6. Deployment stage. you need to follow multioplw senarios

Direct deployment - an old system and an new system. where you stop with your old system and continue with your new system. there is a high risk in this way because you will be removing the old system and if your new system is to fail then you are left with, no working system.

Parellel deployment - is to maintain both systems for a transition time, where you run both the systems in parellel. there is less risk. during the transition period there is a high cost. because you are running both systems at the same time.

pilot deployment - where you take a sample and you let them use the new system and let the others use the old system. for this pilot group this is going be a direct deployment, yet for the company this will be a parellel deployment. the cost is high and the risk is low.

Phases deployment - where, in each phase you introduce a new feature from the new system and remove the old feature from the old system, gradually introducing the new system, eventually removing the old system.

this is a good way of doing the deployment. the risk is low and a good way to handle a complex software.

7... maintainance and support

this is like an long journey. cost plan is hard to predict in the long run.

it is risky to decide to maintain a project for a long time.

Development methodologies. a way we use sdlc in real life is kown as development methodologies.

Homw woik.....read and get an idea of these models

Waterfall model –1

spiral model –2 used for high risk projects, for large projects.

V model –3 verification and validation.

agile model –4 very easyy to adapt to changes.

The RAD model.

Home work.....

do a research on the models pros and cons. SDLC/Dev

startsectionSoftware development life cycle

Is a systematic process or set of phases that guide the development of software applications from conception to deployment and maintenance.

To make the idea into something that is tangible, you need to follow multiple phases which are in between these two stages.

Standard set of steps

Analysis phase Design phase Testing phase

These phases are redone again and again every time you have to make an improvement to the software and update it. Thats why its called a cycle.

1. Analysis phase

Identify the problem Gather the requirement Do an analysis

You need to first identify and understand the needs. Such as the requirements.

This identification is called the elicitation phase

Elicitation phase is to implement methods such as surveys ans intervcies and observation , researdch and litriratki And questuionsareds

The Scope of the projects Inscope and outscope to identify the inscope, to handle them and to identify t;he out scppw ewhvih will not be handled buy us

To identify the functional and the nonfunctional requirements

The functional requierrnet Describes the specific behaviours or the functions that a system must perform The functionally of the program that you will bge making

Moscow standard- a way how we standardise ieee .

Must Should Would Could

Prioritizw and criticalness

The user should be able to log into the dashboard using biometric credentials.

The user who is the target audience and the action functioon which is the dash board And the format which is to be able to by using the biometric

Functional requirements for the library requirements ..

To be able to add and organise books To be able to track books that are lent To be able to generate a user identity per person To be able to return the books To be able to

Feasibility study

1. Technical Feasibility 2. Operational Feasibility 3. Economical Feasibility 1. Expendeture 1.direct expendeture 2. Indirect revenue. 2. Revenue 1. direct expendeture 2. Indirect revenue.

4. Organisational Feasibility

5. Legal Feasibility

you need to consider the legal feasibility.

Software requirment specification. this is the requirement of the first phase.

the second phase is the System design. the srs (Software requirement specification) is given to the architects and senior managers who will the be creating the high level design that outlines the system architecture and component

this is where the data structures and, interfaces and algoriths, and overall system behaviour.

the tools that are used : UML diagrams .

an artifact called the Software Desing Document.

the third chapter is the implementation chapter, where you write the actual code.

this is where you do the unit testing.

at the end of this you will have an code or program which can be executable.

the forth chpter is the "Testing"

to check wether the system outputs and the functional requirements are working properly there are three basic types of testing .

white box testing and black box testings, grey box testing

black box. where anything inside is not visible. where you do all the testing without having any access to any internal code, or any other documents. what you do is that you insert an input and then proceed to verify if the output is correct.

white box. a similtude, like a glass. you have acess to all the code and have an understanding of it. all documents are accessible.

grey box, where the code is partially accessible,

inside a life cycle you have anothe life cycle, software testing life cycled.

each test case contain multiple parameters.

there are 6 phases in this software life cycle.

1. Requirement

Testing Phases.

1. Aceptance testin- which is divided into two parts. 1. you have an involment with the development team, where an member from a different team from the development team.

2. system testing- a black box or a white box 3. intergration testing is trying to integrate the components together and to verify that all these components are working well. done by the develepor 4. Unit testing - done by the develepors them selves 5.

6. Deployment stage. you need to follow multioplw senarios

Direct deployment - an old system and an new system. where you stop with your old system and continue with your new system. there is a high risk in this way because you will be removing the old system and if your new system is to fail

then you are left with, no working system.

Parellel deployment - is to maintain both systems for a transition time, where you run both the systems in parellel. there is less risk. during the transition period there is a high cost. because you are running both systems at the same time.

pilot deployment - where you take a sample and you let them use the new system and let the others use the old system. for this pilot group this is going be a direct deployment, yet for the company this will be a parellel deployment. the cost is high and the risk is low.

Phases deployment - where, in each phase you introduce a new feature from the new system and remove the old feature from the old system, gradually introducing the new system, eventually removing the old system.

this is a good way of doing the deployment. the risk is low and a good way to handle a complex software.

7... maintainance and support

this is like an long journey. cost plan is hard to predict in the long run.

it is risky to decide to maintain a project for a long time.

Development methodologies. a way we use sdlc in real life is kown as development methodologies.

Homw woik.....read and get an idea of these models

Waterfall model –1

spiral model –2 used for high risk projects, for large projects.

V model –3 verification and validation.

agile model –4 very easyy to adapt to changes.

The RAD model.

Home work.....

do a research on the models pros and cons.