A
Seminar report
on

SDLC

Submitted in partial fulfillment of the requirement for the award of
Degree of Computer Science

SUBMITTED TO:

www.studymafia.org

SUBMITTED BY:

www.studymafia.org

Preface

I have made this report file on the topic **SDLC** , I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic.

My efforts and wholehearted co-corporation of each and everyone has ended on a successful note. I express my sincere gratitude to ……..who assisting me throughout the preparation of this topic. I thank him for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it.

Introduction

SDLC, Software Development Life Cycle is a process used by software industry to design, develop and test high quality software's. The SDLC aims to produce a high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software development process.
- The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process.
- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

What is SDLC?

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

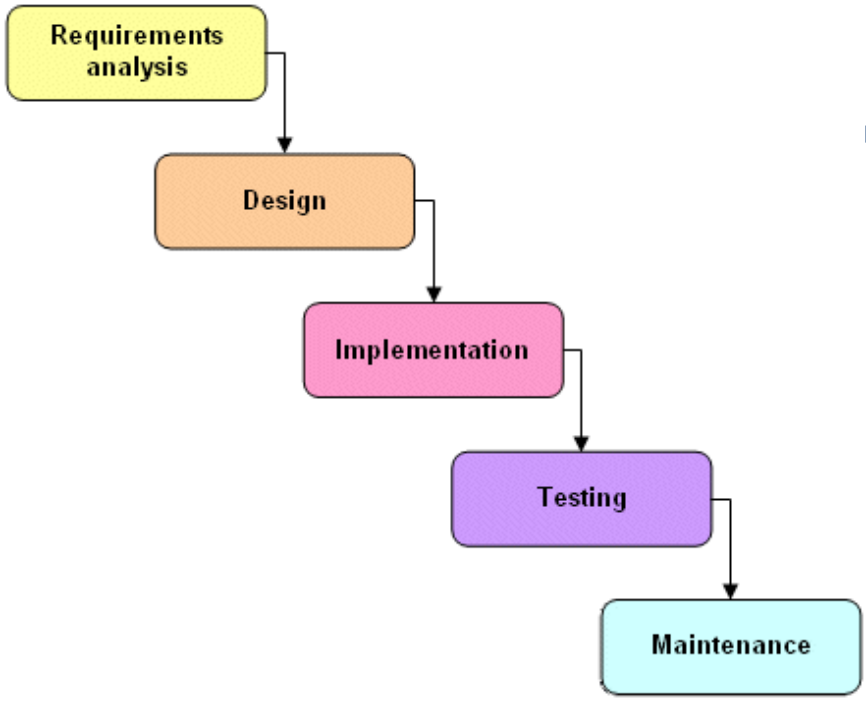
History

The **Systems Life Cycle (SLC)** is a type of methodology used to describe the process for building information systems, intended to develop information systems in a very deliberate, structured and methodical way, reiterating each stage of the life cycle. The systems development life cycle, according to Elliott & Strachan & Radford (2004), "originated in the 1960s,to develop large scale functional business systems in an age of large scale business conglomerates. Information systems activities revolved around heavy data processing and number crunching routines".

Several systems development frameworks have been partly based on SDLC, such as the Structured Systems Analysis and Design Method (SSADM) produced for the UK government Office of Government Commerce in the 1980s. Ever since, according to Elliott (2004), "the traditional life cycle approaches to systems development have been increasingly replaced with alternative approaches and frameworks, which attempted to overcome some of the inherent deficiencies of the traditional SDLC.

Software Development Life Cycle phases?

1. Requirement gathering and analysis
2. Design
3. Implementation or coding
4. Testing
5. Maintenance



1) Requirement gathering and analysis:

Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake holders. Meetings with managers, stake holders and users are held in order to determine the requirements like; who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase. After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied.

Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

2) Design:

In this phase the system and software design is prepared from the requirement specifications which were studied in the first phase. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

3) Implementation / Coding:

On receiving system design documents, the work is divided in modules/units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of the software development life cycle.

4) Testing:

After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. During this phase unit testing, integration testing, system testing, acceptance testing are done.

5) Maintenance

Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

Model of SDLC

Waterfall

Advantages

- Simple goal.
- Simple to understand and use.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are Well documented.
- Easy to manage. Each phase has specific deliverable and a review.
- Works well for projects where requirements are well understood.
- Works well when quality is more important then cost/schedule.
- Customers/End users already know about it.

Disadvantages

- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- No working software is produced until late in the life cycle.
- Risk and uncertainty is high with this process model.
- Adjusting scope during the life cycle can end a project
- Not suitable for complex projects
- Not suitable for projects of long duration because in long running projects requirements are likely to change.
- Integration is done as a "big-bang" at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.
- Users can only judge quality at the end.
- Attempt to go back 2 or more phases is very costly.
- Percentage completion of functionality can not be determined in mid of the project because every functionality is undergoing some phase.
- Very risky, since one process can not start before finishing the other.

Incremental

Advantages

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during an iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment operational product is delivered.
- Issues, challenges & risks identified from each increment can be utilized/applied to the next increment.

Disadvantages

- More resources may be required.
- Although cost of change is lesser but it is not very suitable for changing requirements.
- More management attention is required.
- Each phase of an iteration is rigid with no overlaps.
- System architecture or design issues may arise because not all requirements are gathered. up front for the entire life cycle.
- Does not allow iterations within an increment.
- Defining increments may require definition of the complete system.

Evolutionary

Advantages

- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- During life cycle software is produced early which facilitates customer evaluation and feedback.

Disadvantages

- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which a risk is.
- Can be costly to use.
- Highly skilled resources are required for risk analysis.
- Project's progress is highly dependent upon the risk analysis phase.

Spiral

Advantages

- Changing requirements can be accommodated.
- Allows for extensive use of prototypes.
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided in to smaller parts and more risky parts can be developed earlier which helps better risk management.

Disadvantages

- Management is more complex.
- End of project may not be known early.
- Not suitable for small or low risk projects (expensive for small projects).
- Process is complex.
- Spiral may go indefinitely.
- Large number of intermediate stages requires excessive documentation.

RAD (Rapid Application Development)

Advantages

- Time to deliver is less.
- Changing requirements can be accommodated.
- Progress can be measured.
- Cycle time can be short with use of powerful RAD tools.
- Productivity with fewer people in short time.
- Use of tools and frameworks.

Disadvantages

- Management complexity is more.
- Resource requirements may be more.
- Suitable for systems that are component based and scalable.
- Suitable only when requirements are well known.
- Requires user involvement throughout the life cycle.
- Suitable for project requiring shorter development times.

Extreme/Agile Development

Advantages

- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.

Disadvantages

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.

Conclusion

Software Development Life Cycle (SDLC) is the process of developing information systems through analysis, planning, design, implementation, integration maintenance and testing of software applications. SDLC is also known as information systems development or application development. The development of quality software involves the usage of a structured approach towards the design and development of the end product. In a nutshell, the success of the SDLC process for building a successful software system rests on the following:

- Scope Restriction
- Progressive Enhancement
- Pre-defined Structure
- Incremental Planning at each of the stages

If each of these steps can be followed in its entirety, most of the risks that evolve in the software development life cycle can be mitigated. This article has provided a comprehensive explanation of the important and widely used Software Development Life Cycle (SDLC) phases and its models. It has also provided the advantages and disadvantages of each of these models.