Volume : 4 | Issue : 6 | June 2015 • ISSN No 2277 - 8179

Research Paper

A Comparative Analysis of Software development life cycle Models

Engineering

KEYWORDS : Software development Process, SDLC, phase of SDLC models

Rashmika K. Vaghela Lecturer, R.C.Technical Institute, Sola,Ahmedabad

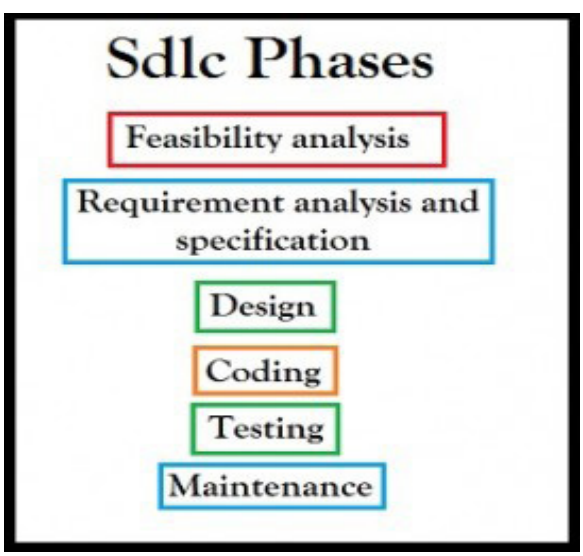
ABSTRACT

There are various software development life cycle models used for developing software. SDLC provides a systematic way for developing software. Each SDLC model has it's own advantages and disadvantages according to which we decide which model is suitable for development under which condition. In this paper the comparison of various SDLC models like waterfall, incremental, spiral and RAD model have been carried out

Introduction

Software development life cycle (SDLC) is a method by which the software can be developed in a systematic manner. SDLC maintain quality of software and increase the probability of completing the software project within the deadline. The Software development life cycle provides a sequence of activities for software development team. Any software development process is divided into several logical stages that allow a software development company to organize its work efficiently in order to build a software product of the required functionalities within specific time period and estimated budget.

Phases of Software development life cycle(SDLC)



There are six phases in every Software development life cycle model:

- Requirement gathering and analysis
- Design
- Implementation or coding
- Testing
- Deployment
- Maintenance

i) 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.

ii) 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.

iii) 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.

iv)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.

v) Deployment: After successful testing the product is delivered / deployed to the customer for their use.

vi) 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.

For large systems, each activity can be extremely complex and methodologies and procedures are needed to perform it efficiently and correctly. Furthermore, each of the basic activities itself may be so large that it cannot be handled in single step and must be broken into smaller steps. For example, design of a large software system is always broken into multiple, distinct design phases, starting from a very high level design specifying only the components in the system to a detailed design where the logic of the components is specified.

In addition to the activities performed during software development, some activities are performed after the main development is complete. There is often an installation phase, which is concerned with actually installing the system on the client's computer systems and then testing it. Maintenance is an activity that commences after the software is developed. Software needs to be maintained not because some of its Components "wear out" and need to be replaced, but because there are often some residual errors remaining in the system which must be removed later as they are discovered. Therefore, maintenance is unavoidable for software systems.

SDLC Models

Waterfall Model

The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is

Volume : 4 | Issue : 6 | June 2015 • ISSN No 2277 - 8179

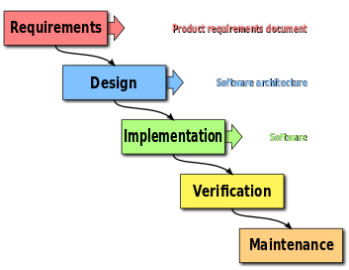
very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

Waterfall Model design
Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model.



Water fall Model
The sequential phases in Waterfall model are:

Requirement Gathering and analysis: All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.

System Design: The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.

Implementation: With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

Integration and Testing: All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

Deployment of system: Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.

Maintenance: There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model phases do not overlap.

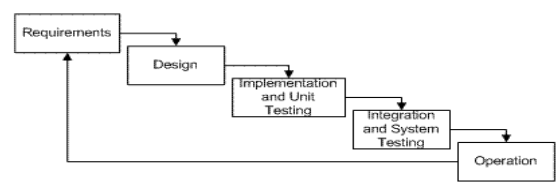
Advantages
The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Disadvantages
The disadvantage of waterfall development is that it does not allow for much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

Incremental Model
The incremental model is an intuitive approach to the waterfall model. Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle. Cycles are divided up into smaller, more easily managed iterations. Each iterations passes through the requirements, design, implementation and testing phases.

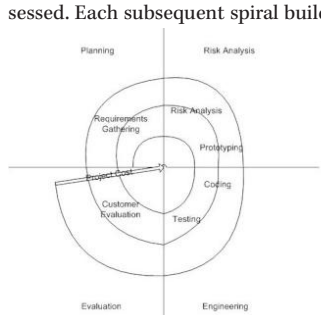
A working version of software is produced during the first iteration, so you have working software early on during the software life cycle. Subsequent iterations build on the initial software produced during the first iteration.



Incremental Life Cycle Model
Advantages
- Generates working software quickly and early during the software life cycle.
- More flexible – less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Easier to manage risk because risky pieces are identified and handled during its iteration.
- Each iteration is an easily managed milestone.

Disadvantages
- Each phase of an iteration is rigid and do not overlap each other.
- Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

Spiral Model
The spiral model is similar to the incremental model, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spirals, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral.



Spiral Model

Planning Phase: Requirements are gathered during the planning phase. Requirements like 'BRS' that is 'Business Requirement Specifications' and 'SRS' that is 'System Requirement specifications'.

Risk Analysis: In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

Engineering Phase: In this phase software is developed, along with testing at the end of the phase. Hence in this phase the development and testing is done.

Evaluation phase: This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

Advantages
- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the software life cycle.

Disadvantages
- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

RAD Model

RAD model is Rapid Application Development model. It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

The phases in the rapid application development (RAD) model are:

Business modeling: The information flow is identified between various business functions.

Data modeling: Information gathered from business modeling is used to define data objects that are needed for the business.

Process modeling: Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.

Application generation: Automated tools are used toconvert process models into code and the actual system.

Testing and turnover: Test new components and all theinterfaces.

Advantages
- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.

Disadvantages
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

As there are various models of software development life cycle, each has its own advantages and disadvantages depending upon which we have to decide, which model we should choose. For instance if the requirements are known before hand and well understood and we want full control over the project at all time, then we can use waterfall model. Incremental model is at the heart of a cyclic software development process. It starts with an initial planning and ends with deployment with the cyclic interactions in between. Easier to test and debug during a smaller iteration. Easier to manage risk because risky pieces are identified and handled during its iteration. Spiral model is good for large and mission critical projects where high amount of risk analysis is required like launching of satellite. RAD Model is flexible and adaptable to changes as it incorporates short development cycles i.e. users see the RAD product quickly. It also involves user participation thereby increasing chances of early user community acceptance and realizes an overall reduction in project risk. The comparison of the different models is represented in the following table on the basis of certain features.

FEATURES	WATERFALL	INCREMENTAL	SPIRAL	RAD
Requirement specifications	Beginning	Beginning	Beginning	Time boxed release
Cost	Low	Low	Expensive	Low
Simplicity	Simple	Intermediate	Intermediate	Very simple
Risk involvement	High	Easily manageable	Low	Very low
Flexibility to change	Difficult	Easy	Easy	Easy
User Involvement	Only at beginning	Intermediate	High	Only at the beginning
Maintenance	Least	Promotes maintainability	Typical	Easily maintained
Duration	Long	Very long	Long	Short
Guaranty of success	Low	Good	High	High
Client Satisfaction	Low	High	High	High

Volume : 4 | Issue : 6 | June 2015 • ISSN No 2277 - 8179

Conclusion

There are many SDLC models such as, Waterfall, RAD, spiral, incremental etc. used in various organizations depending upon the conditions prevailing there. The Waterfall model provides base for other development models. All these different software development models have their own advantages and disadvantages. In this paper we have compared the different software development life cycle models on the basis of certain features like- Requirement specifications, Risk involvement, User involvement, Cost etc. on the basis of these features for a particular software project one can decide which of these software development life cycle models should be chosen for that particular project. Selecting the correct life cycle model is extremely important in a software industry as the software has to be delivered within the time deadline & should also have the desired quality and within estimated cost. This study will make the process of selecting the SDLC model easy& very effective for software development process.

REFERENCE

1. Roger Pressman, titled "Software Engineering - a practitioner's approach" | | 2. Sanjana Taya, Shaveta Gupta, "Comparative Analysis of Software Development Life Cycle Models". | | 3. M. Davis, H. Bersoff, E. R. Comer, "A Strategy for Comparing Alternative Software Development Life Cycle Models", Journal IEEE Transactions on Software Engineering ,Vol. 14, Issue 10, 1988 | | 4. https://www.google.co.in/?gfe_rd=cr&ei=471VVL6BcnM8gemzYCQCw&gws_r=ssl#q=advantage+ and+disadvantage+of+incremental+model | | 5. M. Davis, H. Bersoff, E. R. Comer, "A Strategy for Comparing Alternative Software Development Life Cycle Models", Journal IEEE Transactions on Software Engineering ,Vol. 14, Issue 10, 1988 | 6. Fowler, M. (2000), "Put Your Process on a Diet", Software Development". | | 7. http://software-security.sans.org/resources/paper/cissp/comparison- software-development-lifecycle-methodologies | | 8. Jovanovich, D., Dogsa, T."Comparison of software development models," Proceedings of the 7th International Conference on, 11-13 June 2003, ConTEL 2003, pp. 587-592. | |