International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 02 | Feb-2018 www.irjet.net p-ISSN: 2395-0072

A Comparative Study on Software Development Life Cycle Models

Prof. Supriya Madhukar Salve[1], Prof. Syed Neha Samreen[2], Prof. Neha Khatri-Valmik[3]

[123]Assistant Professor, Dept. of Computer Science and Engineering,
P. E. S. College of Engineering, Aurnagabad-431001, Maharashtra, India

---

Abstract – *The Software Development Life Cycle (SDLC) provides a systematic way for building and delivering software applications. SDLC process is used by the software industry to design, develop and test high quality softwares. The SDLC is used to produce high-quality software that meets customer expectations, software completion within time and estimates cost.*

Key Words: SDLC, Waterfall Model, Spiral Model

1. INTRODUCTION

The Software Development Life Cycle (SDLC) provides a systematic way for building and delivering software applications. SDLC process is used by the software industry to design, develop and test high quality softwares. The SDLC is used to produce high-quality software that meets customer expectations, software completion within time and estimates cost. SDLC is a process followed for a software project, within a software organization. SDLC defines a methodology for improving the quality of software and the overall development process. The figure1 shows various stages of a SDLC.

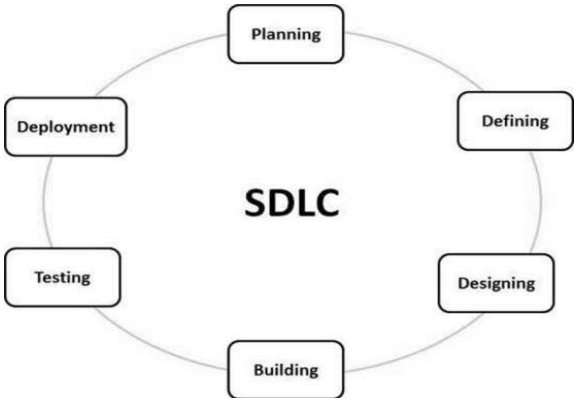

Figure1: Software Development Life Cycle

2. SDLC MODELS

There are different software development life cycle models. Each process model follows a series of steps distinctive to its type for successful software development. Following are the some popular SDLC models used in the industry:

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Prototyping Model

2.1 Waterfall Model

When requirements of a problem are reasonably well understood and when work flows in a linear fashion, Waterfall model is used. E.g. changes in accounting software. It uses a systematic and sequential approach to software development that begins with customer specification of requirement, planning, modeling, construction and deployment as shown in figure 2. The various phases in waterfall model are as follows:

1. Requirement Analysis: This phase defines the needed information, functions, behavior, and performance of the software by communication between vendor and customer. The outcome of this phase is Software Requirement Specification (SRS) document.
2. Design: on the basis of SRS obtained in previous phase the vendor converts the requirements into different modules and prepares a basic level of design document. The output of this phase is data structures, software architecture, interface, algorithmic details.
3. Coding/ Implementation: Actual implementation is done here. The outcome of this phase is source code, database and user documentation.
4. Testing: A team performs testing on the end product. Different types of testing methods can be applied here.
5. Deployment and Maintenance: The software is deployed at client side and if any problem is found is fixed in this phase.

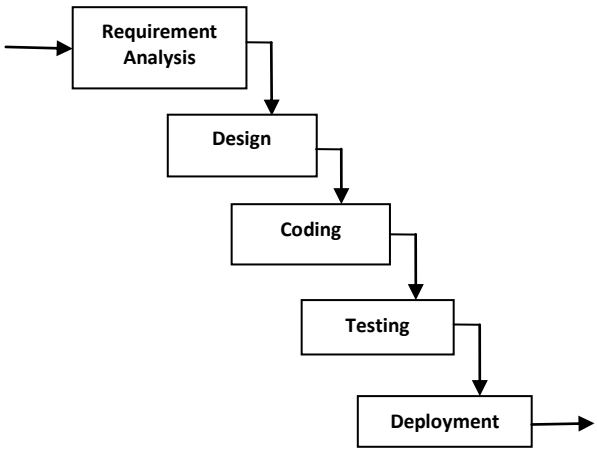

Figure 2: The Waterfall Model

Each phase has specific deliverables. One Phase is processed and completed at a time. This model is suitable for smaller projects when requirements are clear. It reinforces the notions of "define before design" and "design before code". In this model, adjusting scope during the life cycle can kill a project. It does not produce any working software during SDLC. There is a high amount of risk and uncertainty. This is poor model for complex and object-oriented projects, long projects and high risk of changing requirements.

When to use Waterfall Model:

1. When requirements are very well known.
2. Product definition is stable.
3. Technology is understood.
4. New version of an existing product.
5. Changing an existing product to a new platform.

2.2. Incremental Model:

There are circumstances when software requirements are reasonably well defined, but the overall scope of the development effort prevents a purely linear process. There may be need to provide a limited set of software functionality to users quickly and then refine and expand on that functionality in later releases. In such cases, a process model that is designed to produce the software is incremental model. Figure 3 shows incremental model.

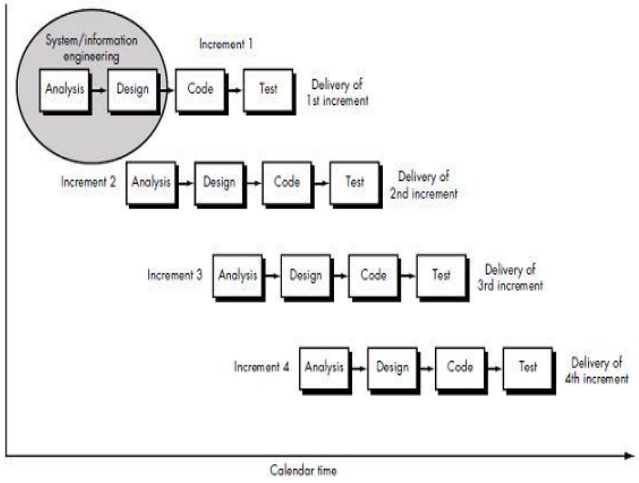

Figure 3: incremental model

This model combines elements of waterfall model applied in an iterative fashion. In figure the incremental model applies linear sequences in a staggered manner as calendar time progresses. Each linear sequence produces deliverable "increments" of the software. The first increment of an incremental model is often a core product that is basic requirements are implemented and but many supplementary features are not implemented. The core product is used by customer. As a result a plan is developed for next increment. The plan focuses on the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment until complete product is produced.

When to use incremental Model:

1. Risk, funding, schedule, program complexity or need for early realization of benefits.
2. Most of the requirements are known but are expected to evolve over time.
3. A need to get basic functionality to the market early.
4. On projects which have lengthy development schedules.
5. It is useful when staffing is unavailable for a complete implementation. Early increments can be implemented with fewer people.
6. When high risk to develop the whole system at once

2.3. Prototype Model:

Due to changing business and product requirement, tight market deadlines completion of software product is almost impossible, but a limited version of can be introduced to meet competitive or business pressure. So when set of core product requirements are well understood, but details are yet to be defined then evolutionary models are used. A set of general objectives are taken from customer, but detailed input, processing and output requirements are not known to customer. In other cases, the developer is unsure of the efficiency of an algorithm, the adaptability of an operating system etc, then prototyping is the best approach as shown in figure 4.

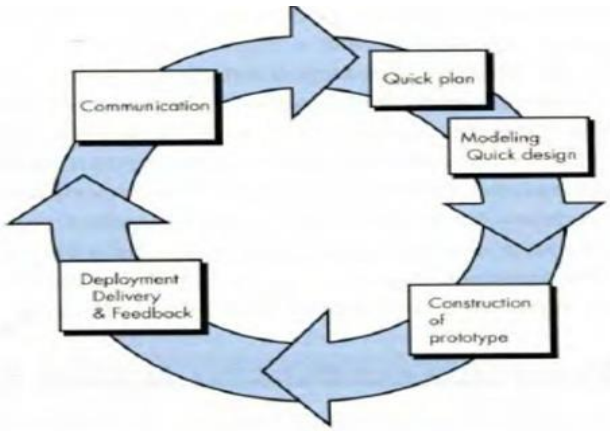

Figure 4: Prototype model

The first phase is communication in which requirements are gathered by having communication with customer. Then quick plan is developed to implement the stated requirements. Further, modeling and quick design is made. After this phase the actual construction of prototype is done which is then deployed and delivered to customer. The feedback is taken from customer; changes are made into software and software development is carried out. It also allows the software engineer some insight into the accuracy

of initial project estimates and whether the deadlines and milestones proposed can be successfully met. The limitation of this model is that the developers may lose focus on the real purpose of the prototype and compromise the quality of the product. For example, they may employ some of the inefficient algorithms or inappropriate programming languages used in developing the prototype.

When to use Prototyping Model:

1. Prototyping is very effective in the analysis and design of on-line systems.
2. Systems with little user interaction, such as batch processing or systems that mostly do calculations benefit little from prototyping. Sometimes, the coding needed to perform the system functions may be too intensive and the potential gains that prototyping could provide are too small.
3. Prototyping is especially good for designing good human-computer interfaces. "One of the most productive uses of rapid prototyping to date has been as a tool for iterative user requirements engineering and human-computer interface design."

2.4. Spiral Model

It is invented by Dr. Barry Boehm in 1988 while working at Thompson Ramo Wooldridge (TRW) Automotive. This model follows an iterative nature of prototyping model and systematic approach of waterfall model. This is used when requirements are not well understood and risks are high. Outer spiral take on a classical waterfall model approach after requirements are defined, but permit iterative growth of the software. It operates as a high risk-driven model. A go/ no-go decision occurs after each complete spiral in order to react to risk determination. It requires considerable expertise in risk assessment. This model serves as realistic model for large scale projects. Each cycle follows same sequence of steps as waterfall as shown in figure 5.

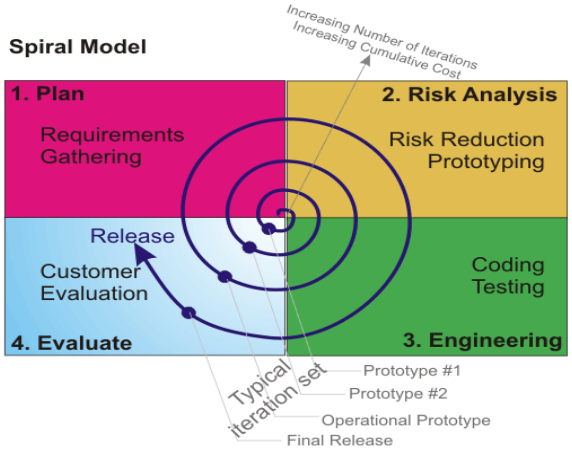

Figure 5: Spiral Model

When to use Spiral Model:

1. When cost and risk evaluation is important.
2. For medium to high risk projects
3. Long term project commitment unwise because of potential changes to economic priorities.
4. Users are not sure of their needs.
5. Requirements are complex.
6. New product line.
7. Significant changes are expected. eg. Research and Exploration.
8. For a typical shrink-wrap application, the spiral model might mean that you have a rough-cut of user elements (without the polished / pretty graphics) as an operable application, add features in phases, and, at some point, add the final graphics.
9. The spiral model is used most often in large projects. The US military has adopted the spiral model for its Future Combat Systems program.

3. Comparison of various Models:

SDLC Model	Advantages	Disadvantages
1. Waterfall Model	1. Easy to understand, easy to use. 2. Plan and Do Model. 3. Provide structure to inexperienced staff. 4. Milestones are well understood. 5. Sets requirement stability. 6. Suitable for smaller projects where requirements are well understood.	1. Once an application is in testing phase, it is very difficult to go back and change something that was not correct. 2. No working software is produced until late during the life cycle. 3. High amount of risk and uncertainty. 4. Poor model for large projects. 5. Not suitable for unclear and changing requirements. 6. Lack of involvement of customer. 7. Accuracy of estimation. 8. Unable to accommodate iteration.

2. Incremental Model	1. It is iterative model. 2. It focuses on the delivery of an operational product with each increment. 3. Customer can respond to each iteration. 4. It develops high risks or major functions first. 5. It uses "Divide and conquer" method. 6. Initial product delivery is faster. 7. Customer gets important functionality early. 8. A risk of changing requirements is reduced.	1. It requires good planning and design. 2. It requires early definition of a complete and fully functional system to allow for increments. 3. Well defined module interfaces are required. 4. Total cost of the complete system is not lower.
3.Prototyping Model	1. It follows an evolutionary and iterative approach. 2. Used when requirements are not well understood. 3. Serves as a mechanism identifying software requirements. 4. Focuses on those aspects of software that are visible to customer/user. 5. Feedback is used to refine the prototype.	1. The customers see a "working version" of software and buy the prototype after few fixes are made. 2. Developers often make implementation compromises to get the software running quickly. E.g. language choice, user interfaces, operating system choice, inefficient algorithm. 3. Increases complexity of the overall system. 4. Lesson learned- a. Define the rules for prototype before it is built. b. Plan to discard the prototype and engineer the actual production software with a goal toward quality.
1) 4. Spiral Model	1. High amount of risk analysis is done. Hence, avoidance of risk is enhanced. 2. Good for large and mission critical projects. 3. Strong approval and documentation control. 4. Additional functionality added in later date. 5. Software is produced early in software life cycle.	1. Can be costly model to use. 2. Risk analysis requires highly specific expertise. 3. Project success is highly dependent on risk analysis phase. 4. Doesn't work well for smaller projects.

4. CONCLUSION

Based upon the need of the software project suitable software development lifecycle model can be used. The Waterfall life cycle model is very simple and easy to use model which can be used for smaller projects. The Incremental model applies a series of iterations to the Waterfall model and used when there is need to provide a limited set of software functionality to users quickly and then refine and expand on that functionality in later releases . Prototyping is very effective in the analysis and design of on-line systems. Systems with little user interaction, such as batch processing or systems that mostly do calculations benefit little from prototyping. The Spiral life cycle model builds upon the Waterfall and Incremental models and focuses on risk analysis. Spiral mosel is used when cost and risk evaluation is important.

REFERENCES

[1] Roger Pressman, "Software Engineering: A Practitioner Approach", 7th Edition.

[2] Adel Alshamrani, Abdullah Bahattab, "A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model", IJCSI International Journal of Computer Science Issues, Volume 12, Issue 1, No 1, January 2015.

BIOGRAPHIES

Ms. Supriya Madhukar Salve has a teaching experience of more than 10 years and currently working as Assistant Professor at PESCOE, Aurangabad.

Ms. Neha Syed Samreen has more than 4 years of experience and currently working as Assistant Professor at PESCOE, Aurangabad.

Ms Neha Khatri-Valmik has teaching experience of more than 8 years and currently working as Assistant Professor at PESCOE, Aurangabad.