A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model

Adel Alshamrani[1] and Abdullah Bahattab[2]

[1] Ira A. Fulton Schools of Engineering at Arizona State University
Tempe, AZ, 85281, USA
[1] King Abdul Aziz University, Faculty of Computing and Information Technology-North Branch,
Jeddah, 22245, KSA
[2] College of Telecommunications and Electronics (CTE),
Jeddah, 21533, KSA

Abstract

The computer has become indispensable in today's life, and it is widely used in many fields of life such as commerce, education, industry…etc. The computer saves time in regarding to help solving complex, long, repeated processes in a short time and high speed. As the software programs need to handle these features, many companies produce software programs to facilitate the works for administrations, banks, offices, etc. Moreover, software has been in used for analyzing information or solving problems for more than four decades. Creating a suitable work to develop programs of high quality is the main goal of the software engineering. Usually, clients seek the assistance from computer and software engineers to solve and handle their problems. There are various models have been widely in used to develop software products. Common models will be described in this paper.

Keywords*: SDLC Models, Software Engineering, Waterfall model, Spiral model. Iterative model.*

1. Introduction

Software development life cycle or SDLC for short is a methodology for designing, building, and maintaining information and industrial systems. So far, there exist many SDLC models, such as the Waterfall model, which comprises five phases to be completed sequentially in order to develop a software solution; another model called the Spiral model, which is visualized as a process passing through some number of iterations. Finally, the incremental model is any combination of both iterative design or iterative method and incremental building model for software development. It has seven phases, and they are as follows: Planning, requirements, analysis, implementation, deployment, testing, and evaluation [1, 3]. In effect, SDLC has been investigated by many researchers and numerous models have been proposed where their acknowledged strengths and weaknesses are presented. The Waterfall, spiral, incremental, rational unified process (RUP), rapid application development (RAD), agile software development, and rapid prototyping are few to mention as successful SDLC models. Moreover, all SDLC models that have been suggested share basic properties. They all consist of a sequence of phases or steps that must be followed and completed by system developers and designers in order to achieve developed systems and deliver required products. However, in this paper, strengths and weaknesses of The Waterfall, Spiral, and Incremental/Iterative models will be discussed and a brief comparison of other aspects will conclude the rest of the paper.

2. Waterfall Model

The Waterfall Model is the oldest and the most well-known SDLC model. This model is widely used in government projects and in many major companies. The special feature of this model is its sequential steps. It goes downward through the phases of requirements analysis, design, coding, testing, and maintenance. Moreover, it ensures the design flaws before the development of a product. This model works well for projects in which quality control is a major concern because of its intensive documentation and planning [5].Stages that construct this model are not overlapping stages, which means that the waterfall model begins and ends one stage before starting the next one.

The following steps give a brief description about the waterfall process:

1. Requirement: Is a description of a system behavior to be developed. Usually, it is the information provided by clients. Hence, it establishes the agreement between the clients and the developers for the software specifications and features. In short, requirements are gathered, analyzed and then proper documentation is prepared, which helps further in the development process."
2. High Level design: The gathered information from the previous phase is evaluated and a proper implementation is formulated. It is the process of planning and problem solving for a software solution. It deals with choosing the appropriate algorithm design, software architecture design, database

conceptual schema, logical diagram design, and data structure definition [4, 5].

3. Coding: In this phase the whole requirements will be converted to the production environment.
4. Testing: This phase deals with the real testing and checking of the software solutions that have been developed to meet the original requirements. Also, it is the phase where the bugs and system glitches are found, fixed up, and refined.
5. Maintenance: After the software is already released, it may need some modifications, improvements, errors correction, and refinement accordingly. Thus, this phase is the process of taking care of such concerns.

3. Spiral Model

The spiral model is a software development process combines elements of both design and prototyping in stages for the sake of combining the advantages of top-down and bottom up concepts. It is a meta-model, which means that it can be used by other models [5, 6]. In addition, it focuses on risk assessment and minimizing project risk. This is can be achieved by breaking a project into smaller segments, which then provide more ease-of-change during the development process, as well as providing the opportunity to evaluate risks and weigh consideration of project continuation throughout the life cycle. In this model, the development team starts with a small set of requirements and then goes through each development phase (except Installation and Maintenance) for those set of requirements. Therefore, the development team has a chance to learn new lessons from the initial iteration (via a risk analysis process). Also, the team will add functionality for additional requirements in ever-increasing "spirals" until the application is ready for the installation and maintenance phase. In this model, each iteration prior to the production version is called a prototype of the application [7, 8, 9, 10].

The following steps give a brief description about the Spiral model phases:

1. Planning: This phase includes the understanding of the system requirements by conducting continuous communications between the customers and the system analysts.
2. Risk Analysis: In this phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of this phase.
3. Development/Engineering: In this phase the software is produced along with the testing.
4. Evaluation Phase: This allows the customer to evaluate the output of the project before the project continues to the next spiral or next round.

4. Iterative and Incremental Model

This model combines elements of the waterfall model in an iterative fashion. Moreover, each linear sequence produces deliverable increments of the software. The basic requirements are addressed in the first increment, and it is the core product, however, many supplementary features (some known, others unknown) remain undeliverable at this increment. This model constructs a partial implementation of a total system. Then, it slowly adds increased functionality. Therefore, each subsequent release will add a function to the previous one until all designed functionalities are implemented [7, 8, 9, 10].

5. Comparison of the three SDLC Models (Waterfall, Spiral, and incremental)

As we have already mentioned above, there are many SDLC models each of which has different level of risk, budget, estimated completion timeline, and benefits to cope with the project requirements. In addition, some models are preferred over others in regard to the size of the project either large or small while other models being preferred due to their flexibility to allow rapid changes throughout the whole life cycle of the software development [1, 2, 5, 6]. Thus, developers have to consider various aspects before choosing the SDLC model to implement the required system. They must know the strengths and weaknesses of each model, and when to use the appropriate model. Therefore, the tables (1 and 2) provide some helpful information, which shows the comparison between the three SDLC models in regard to their strengths, weaknesses, other aspects, and when to use each.

6. Conclusion

In this research, we concluded that there are many existing models for developing systems based on clients' requirements and the size of projects. Some models are preferred over the others due to their properties and how they match the clients' needs. The waterfall model, spiral model, and incremental model may have same shared properties, but they still have different advantages and disadvantages for the development of systems, so each model tries to eliminate the disadvantages of the previous model. In the future work, we are planning to extend this research to add other models and some models might be simulated using some tools.

Table 1: Strengths and Weaknesses Comparison of Waterfall, Spiral, Incremental SDLC Models.

Model/feature	Strengths	Weaknesses	When to Use
Waterfall	• Easy to understand and implement. • Widely used and known. • Define before design, and design before coding. • Being a linear model, it is very simple to implement. • Works well on mature products and provides structure to inexperienced teams. • Minimizes planning overhead. • Phases are processed and completed one at a time.	• All requirements must be known upfront • Inflexible. • Backing up to solve mistakes is difficult, once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage. • A non-documentation deliverable only produced at the final phase. • Client may not be clear about what they want and what is needed. • Customers may have little opportunity to preview the system until it may be too late. • It is not a preferred model for complex and object-oriented projects. • High amounts of risk and uncertainty, thus, small changes or errors that arise in the completed software may cause a lot of problems.	• When quality is more important than cost or schedule. • When requirements are very well known, clear, and fixed. • New version of existing product is needed. • Porting an existing product to a new platform
Spiral	• High amount of risk analysis. • Software is produced early in the software life cycle. • Strong approval and documentation control. • Additional functionality can be added at a later date. • Project monitoring is very easy and effective. • Concerned people of a project can early review each phase and each loop as well because of	• Cost involved in this model is usually high. • Risk assessment expertise is required. • Amount documentation required in intermediate stages makes management of a project very complex. • Time spent for evaluating risks for small or low-risk projects may be too large. • Time spent for planning, resetting	• For medium to high-risk projects. • When risk evaluation and costs are important. • When significant changes are expected.

	rapid prototyping tools. • Early and frequent feedback from users • Suitable to develop a highly customized product. • Provides early indication of insurmountable risks.	objectives, doing risk analysis, and prototyping may be excessive. • Project's success is highly dependent on the risk analysis phase.	• When users are not exactly sure what their needs.
Incremental/ **Iterative.**	• Develop high-risk or major functions first. • Risk is spread across smaller increments instead of concentrating in one large development. • Lessons learned at the end of each incremental delivery can result in positive revisions for the next increment. • Customers get important functionality early, and have an opportunity to respond to each build. • Each release delivers an operational product. • Initial product delivery is faster. • Reduces the risk of failure and changing the requirements.	• Requires good planning and design. • Requires early definition of a complete and fully functional system to allow for the definition of increments. • The model does not allow for iterations within each increment.	• On low to medium-risk projects. • A need to get basic functionality to the market early • On projects which have lengthy development schedules. • On a project with new technology, allowing the user to adjust to the system in smaller incremental steps rather than leaping to a major new product. • When it is high risky to develop the whole system at once.

Table 2: Comparison of SDLC models (Waterfall, Spiral, and Iterative model)

Model/Feature	**Waterfall**	**Spiral**	**Incremental/Iterative**
Specification of All the Requirements in the beginning	Yes	Not all and Frequently Changed	Not all and Frequently Changed
Long term project	Inappropriate	Appropriate	Appropriate
Complex Project	Inappropriate	Appropriate	Appropriate
Frequently Changed Requirements	Inappropriate	Appropriate	Appropriate
Cost	Not costly	Costly	Costly
Cost estimation	Easy to estimate	Difficult	Difficult
flexibility	Not	Less flexible	Flexible
Simplicity	Simple	Intermediate	Intermediate
Supporting high risk projects	Inappropriate	Appropriate	Appropriate
Guarantee of Success	Less	High	High
Customer Involvement	Low	Low, After Each Iteration	High, After Each Iteration
Testing	Late	At the end of each phase	After every Iteration
Maintenance	Least maintainable	Yes	Maintainable
Ease of Implementation	Easy	Complex	Easy

References

[1] www.en.wikipedia.org/wiki/Systems_development_life-cycle
[2] http://www.craiglarman.com/wiki/downloads/misc/history-of-iterative-larman-and-basili-ieee-computer.pdf
[3] Craig Larman and Victor Basili, "*Iterative and Incremental Development: A Brief History"*, IEEE Computer, 2003.
[4] Y. Bassil "*A simulation model for the waterfall software development life cycle. International Journal of Engineering & Technology*", 2(5): 1-7, 2012.
[5] Nabil Mohammed Ali Munassar and A. Govardhan, "*A Comparison Between Five Models Of Software Engineering",* IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 5, , pp. 94 – 101, September 2010.
[6] Ashwini Majumdar, Gayatri Masiwal, P.M.Chawan, "Analysis of Various Software Process Models" *International Journal of Engineering Research and Applications,* Vol. 2, No. 3, 2012, pp. 2015-2021.
[7] N. Munassar and A. Govardhan, "*A Comparison Between Five Models Of Software Engineering*", IJCSI International Journal of Computer Science Issues, vol. 7, no. 5, 2010.
[8] Ian Sommerville, Software Engineering, Addison Wesley, 9th ed., 2010.
[9] Jim Hurst, *"Comparing Software Development Life Cycles*," SANNS Software Security, 2014.
[10] Sanjana Taya and Shaveta Gupta, "*Comparative Analysis of Software Development Life Cycle Models*," IJCST Vol. 2, Issue 4, Oct . - Dec. 2011

Adel Alshamrani has obtained his Bachelor in Computer Science from Umm Al-Qura University, Saudi Arabia in 2007, and has obtained his Master degree in Computer Science from La Trobe University, Australia in 2010. Adel is a lecturer in the Department of Computer Science and Information Technology at King Abdul Aziz University. Now, he is pursuing his Ph.D in Computer science (Information Assurance) at Arizona State University, USA.

Abdullah Bahattab has obtained his B.S., Masters, and Ph.D. in Computer Science from King Abdulaziz University, KSA in 1989, Western Michigan University, USA in 1995, and Illinois Institute of Technology, USA in 2000, respectively. He worked as the Dean of College of Telecommunications and Electronics (CTE), Jeddah, KSA. He is an arbitration committee member of International Chamber of Commerce (ICC). His research interests are in Computer networks, routing, switching, wireless networks, and E-learning researches. He got two patents from the USA patent office. Because of the first patent, he got an honor letter from King Abdullah bin Abdulaziz. He is the author of two books and co-author of a book.