ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

COMPARATIVE STUDY: WATERFALL V/S AGILE MODEL

Ms.Akshita Dubey*, Ms.Amisha Jain, Ms.Aditi Mantri
* Acropolis Institute of Technology and Research, Indore (M.P), India
Acropolis Institute of Technology and Research, Indore (M.P), India
Acropolis Institute of Technology and Research, Indore (M.P), India

ABSTRACT

This investigation deals with a vital and critical matter in computer world. It is concerned with the development process of software. This target can be achieved with the help of various software development models available. The development models are tools that allow us to correctly follow the steps to create software that meets a business need. There are different SDLC models with their respective pros and cons. In IT world all these methodologies are incorporated .Any model is implemented by taking in context every aspect and proper confidentiality and integrity. Availability controls are planned and built into software application right through the software lifecycle. Currently, the use of, awareness in, and controversy about agile methodologies have realized dramatic growth. We have described the attributes of some traditional and agile methodologies that are far and wide used in software development. We have also discussed the strengths and weakness between the two opposing methodologies and provided the challenges associated with implementing agile processes in the software industry.

KEYWORDS: waterfall model, agile model.

INTRODUCTION

Software has been an indispensable part of modern society for a long time. Several software development methodologies are in use today. Various companies have their own modified tactics for their software enhancement but the majority speaks about two kinds of methodologies: heavyweight and lightweight. The software development life cycle (SDLC) is the process consisting of a sequence of planned phases to develop or to amend the software products. In this monograph an overview of SDLC models is discussed. Heavyweight methodologies, the most conventional way of software development, assert their support to comprehensive planning, thorough documentation, and extensive design. On the other hand, the lightweight methodologies also known as agile modeling has gained considerable recognition from the software engineering society in the last few years. SDLC models should be chosen on the basis of requirements and size of any project. The objective of SDLC is to produce high trait software that will satisfy the needs of customer and also provide a clear-cut idea about the development phases to the customer as well as the developer. The traditional life cycle is essentially sequential. Some stages focus on the early part of the project, while others occur toward the end. To meet the demands of the current environment, a new SDLC model needs to allow developers to perform some tasks in different stages concurrently. In this world of neck to neck competition, system developers need the flexibility to respond rapidly to environmental opportunities and threats even in the midst of the project, for the project to be successful. SDLC is a structure/basic building block defining functions performed in individual steps.

Typically SDLC is completed in following stages:

1. Planning and analysis of requirements.
2. Defining requirements
3. Designing the software architecture.
4. Developing the product
5. Testing it.
6. Deployment in market & maintenance.

Characterstics of SDLC:

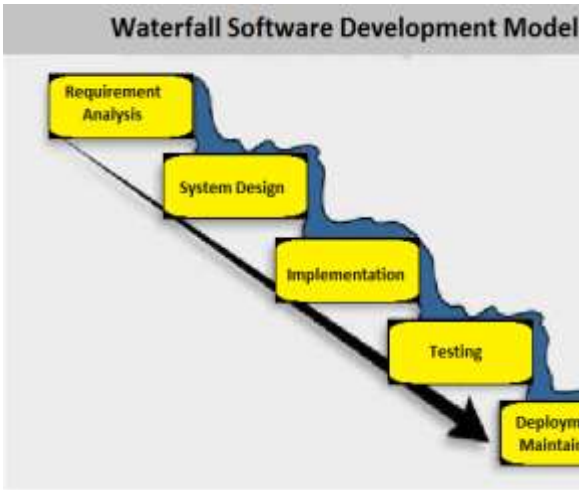
- Minimal expenditure
- Flexibility and feedback.
- Simultaneous tasks.
- Thorough analysis.

WATERFALL MODEL

The waterfall model is a sequential, down-flow model often used in software development processes, it is called so because all the phases of Analysis, Design, Production/Implementation, Construction, Testing,

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

and Maintenance are executed one by one and flow downwards like a waterfall. Waterfall model is a sequential and incremental development model. The oldest of the SDLC's and the finest known.



The chronological segments in Waterfall model are:

Requirement analysis and information gathering: From top to bottom all the necessities of the system to be developed are captured in this fragment and documented in a prerequisite specification document.

System Design: System Design helps in specifying hardware and also provide a virtual overview of the system/software to be designed.

Implementation: With inputs from above phase, the system is first moduled in small programs called units, which are integrated in the coming step. Each program is tested on a particular scale to which is referred to as Unit Testing.

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

Deployment of system: The small units are merged into one functional unit and are tested. Once the testing is finished, the product is set up in the customer environment or released into the market.

Maintenance: There are some concerns which come up in the client atmosphere. To repair those issues patches are released. Also to enhance the artifact some better versions are released. Maintenance is done to bring these changes in the customer environment.

Pros

- Prerequisite is clear before development commences.
- Each phase is accomplished in specified period of time after that it moves to next phase.
- As it is a linear model, it's simple to employ.
- The quantity of resources required to employ this model are nominal.
- Each phase appropriate documentation is followed for the eminence of the development.

Cons

- Sarcastically, the biggest shortcoming is one of its greatest benefits. You cannot go back a step; if the design phase has gone erroneous, things can get very problematical in the implementation phase.
- Often, the client is not very precise of what he exactly wants from the software. Any changes that he reveals in between, may cause a lot of confusion.
- Small alterations or errors that come up in the completed software may cause a lot of problems.

Best use of waterfall methodology can be done when:

1. There is a clear depiction of what the final product should be.
2. Clients won't have the ability to change the extent of the project once it has begun.
3. Characterization is more important than speed.

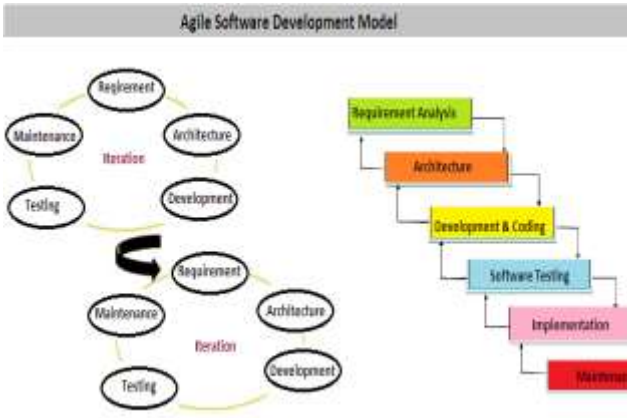
Agile model:

Agile thought process had commenced early in the software development and started becoming trendy with time due to its elasticity and adaptability.

Iterative approach is taken and working software build is conveyed after each iteration. Each build is incremental in terms of characteristics; the final build possesses all the features obligated by the customer.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

The most admired agile methods include Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995).



Following are the agile platform principles:

- Individuals and interactions - In agile process, self-organization and enthusiasm are important, as are interactions like co-location and pair programming.
- Working software - Demo working software is considered the best resource of communication with the customer to understand their condition, instead of just relying on documentation.
- Customer collaboration - As the necessities cannot be gathered absolutely at the time of foundation of the project due to various aspects, continuous customer interaction is very vital to get proper product requirements.
- Responding to change - agile development is paying attention on quick reaction to change and incessant development.

Pros:

- Functionality can be developed rapidly and verified side by side.
- Good model for environments that change progressively.
- Little or no planning required, development and deployment are carried out side by side.

Cons:

- Hard to maintain and sustain the needs, because any of the step is not finished actually.
- Documentation is very less hence individual dependencies are increased.
- Depends heavily on interaction with costumer so if costumer is not cleared then it will create ambiguity between the developers.

When should you use Agile methodology?

1. When rapid fabrication is more important than the eminence of the product.
2. When clients will be proficient to change the scope of the project.
3. When there isn't a clear image of what the final creation should look like.
4. When you have skilled developers who are adaptable and capable to think independently.
5. When the product is proposed for an industry with rapidly varying standards.

Comparison between Waterfall and Agile Methodologies:

Waterfall is defined as a sequential development model with clearly defined deliverables for every phase. Many industry practitioners are strict in performing audit reviews to ensure that the project has satisfied the input criteria before continuing to the next phase. while, agile model is based on the adaptive software development methods. It is flexible as well as lucid i.e.it gives freedom to the client demanding for the software. Based on some renown features a table comparing waterfall and agile model is given below:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

MODEL/FEATURES	WATERFALL MODEL	AGILE MODEL
Requirement Specifications	Beginning	Frequently changed
Understanding Requirements	Well Understood	Well Understood
Cost	Low	Very High
Guarantee of Success	Low	Very High
Resource Control	Yes	No
Cost Control	Yes	Yes
Simplicity	Simple	Intricate
Risk Involvement	High	Reduced
Expertise Required	High	Very high
Changes Incorporated	Difficult	Difficult
Risk Analysis	Only at beginning	Yes
User Interaction	Only at beginning	High
Overlapping Phases	No Such Phase	Yes
Flexibility	Rigid	Highly Flexible
Maintenance	Least Glamorous	Promote Maintenance Ability
Integrity &Security	Vital	Obvious
Reusability	Limited	Reusable
Interface	Minimal	Model-driven
Documentation &Training required	Vital	Yes
Time Frame	Long	Least possible

AGILE	WATERFALL
Architecture is informal and incremental.	Architecture is very well documented &finalized before coding starts.
Developers share possession of the code.	Each developer is responsible for one area.
Continuous integration	Integration executed at one end or after milestone
Focus is on completing stories (functionalities) in short iterations	Focus is on completing modules (parts of the architecture) at large milestones
Relies on engineering practices (TDD, refactoring design patterns…)	Doesn't necessarily rely on engineering practices
Light process and documentation	Heavy process and documentation
Requires cross-trained developers, familiar with all vital technologies.	Relies on a small group of architects/ designers to overview the complete code, the rest of the team can be very specialized.
Main roles: Developer	Main role: architect, developer
Open door policy. Developers are encouraged to talk directly with business, QA & management at any time. Everyone's point of view is considered.	Only a few developers, & some architects can contact some business people. Communication happens mainly at the beginning of the project & at the signposts.

CONCLUSION

This was about the SDLC models and the scenarios in which these SDLC models are used. The information in this paper will help the project developers to decide which SDLC model would be suitable for their project and it would also help the developers and testers to understand basics of the development model being used for their project. We have discussed both the popular SDLC models in the industry, both traditional and modern. This paper also gives an insight into the pros and cons and the practical applications of the SDLC models discussed. Waterfall is traditional SDLC model and is of sequential type. Sequential means that the next phase can start only after the completion of first phase. Such models are suitable for projects with very clear product requirements and where the requirements will not change dynamically during the course of project completion. Agile is the most popular model used in the industry. Agile introduces the concept of fast delivery to customers using prototype approach. Agile divides the project into small iterations with specific deliverable features. Customer interaction is the backbone of Agile methodology, and open communication with minimum documentation are the typical features of agile development environment.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

ACKNOWLEDGEMENT

We would like to gratefully and sincerely thank Mr. Deepak Agrawal for his guidance and understanding. His mentorship was paramount in providing a well rounded experience consistent our long-term career goals. He encouraged us to not only grow as an experimentalist and an engineer but also as independent thinker. We would also like to thank CSE department, Acropolis Institute of Technology &Research for their assistance and guidance in making of this research paper and giving us the platform to do so. Finally and importantly, we would like to thank our parents. Their support, encouragement and quiet patience helped us a lot in making of this report.

REFERENCES

1. Winston Royce, "Managing the Development of Large Software Systems", Proc. Westcon, IEEE CS Press, 1970,
2. pp. 328-339.
3. 2 Standish Group International, Inc., "Chaos Chronicles", 1994,
4. http://www1.standishgroup.com//sample_research/chaos_1994_1.php
5. 3 Kent Beck, "Extreme Programming Explained: Embrace Change", Addison-Wesley, 2000, pp. 18-19.
6. 4 Ken Schwaber, Mike Beedle, "Agile Software Development with Scrum", Prentice Hall, 2001, pp. 89-94
7. 5Standish Group International, Inc., "Chaos Chronicals", 1994,
8. http://www1.standishgroup.com//sample_research/chaos_1994_1.php
9. 6 I. Nonaka, H. Takeuchi, "The New New Product Development Game", Harvard Business Review, January 1986,
10. pp. 137-146.
11. 7 For more on how lean development influences agile software development, see: Mary Poppendieck, Tom
12. Poppendieck, "Lean Software Development An Agile Toolkit", Addison-Wesley, 2003.
13. 8 Craig Larman, Victor R. Basili, "Iterative and Incremental Development: A Brief History", Computer, IEEE CS
14. Press, June 2004, p. 48.
15. 9 I. Nonaka, H. Takeuchi, "The New New Product Development Game", Harvard Business Review, January 1986,
16. pp. 137-146.
17. 10 The Agile Manifesto is online at http://www.agilemanifesto.org/
18. 11 Barry Boehm, "Software Engineering Economics", Prentice Hall PTR, 1981.
19. 12 Kent Beck, "Extreme Programming Explained: Embrace Change", Addison-Wesley, 2000.
20. 13 Reexamining the Cost of Change Curve, year 2000, by Alistair Cockburn; XP Magazine, September 2000.
21. 14 Examining the Cost of Change Curve, by Scott Ambler; Agile Modeling Essays excerpted from the book "The
22. Object Primer, 3rd ed.: Agile Model-Driven Development with UML2"; by Scott Ambler, Cambridge University
23. Press, 2004.
24. 15 Martin Fowler, "Is Design Dead", http://martinfowler.com/articles/designDead.html , 2004.
25. 16 Ken Schwaber, Mike Beedle, "Agile Software Development with Scrum", Prentice Hall, 2001, pp. 100-101
26. 17 In fact, it is debatable whether Quality is really an adjustable factor. As professionals, software developers find it
27. very objectionable when asked to skimp on quality. Surgeons or lawyers would be sued for malpractice, and for the
28. same ethical implications software developers resent this charge. Software developers should always aim for high
29. quality software, period.
30. 18 Kent Beck, "Extreme Programming Explained: Embrace Change", Addison-Wesley, 2000, pp. 15-19.
31. 19 The Scrum Agile method has its roots in the Nonaka-Takeuchi article referenced above.
32. 20 Mary Poppendieck, Tom Poppendieck, "Lean Software Development An Agile Toolkit", Addison-Wesley, 2003,
33. p 28.
34. 21 Jim Johnson, "ROI, It's Your Job!", Published Keynote Third International Conference on Extreme Programming
35. 2002.
36. 22 Mary Poppendieck, Tom Poppendieck, "Lean Software Development An Agile Toolkit", Addison-Wesley, 2003,
37. p 32.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

38. 23 Standish Group International, Inc., "Chaos Chronicals", 2004.
39. 24 "Standish: Project Success Rates Improved Over 10 Years", Software Magazine and Weisner Publishing,
40. http://www.softwaremag.com/L.cfm?Doc=newsletter/2004-01-15/Standish.
41. 25 Software architect Michael James has a semi-serious theory that "test-only" development will soon be feasible
42. with the advances in cheap, clustered supercomputing. Developers write only robust tests and supercomputers will
43. write and compile code until all tests pass.
44. [1] Laura C. Rodriguez Martinez, Manuel Mora ,Francisco,J.Alvarez, "A Descriptive/Comparative Study of the Evolution of Process Models of Software Development Life Cycles", Proceedings of the 2009 Mexican International Conference on Computer Science IEEE Computer Society Washington, DC, USA, 2009.
45. [2] 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.
46. [3] A. 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
47. [4] Sharma, B.; Sharma. N, "Software Process Improvement: A Comparative Analysis of SPI models", Emerging Trends in Engineering and Technology (ICETET), 2009 2nd International Conference on,16-18, 2009, pp. 1019-1024
48. [5] Maglyas, A.;Nikula, U.; Smolander, T.,"Comparison of two models of success prediction in software development projects", Software Engineering Conference (CEE-SECR), 2010 6th Central and Eastern European on 13-15 Oct. 2010, pp. 43-49.
49. [6] Osborn, C. SDLC, JAD, RAD,"Center for Information Management Studies", 2001.
50. [7] Rothi, J.,Yen, D, "System Analysis and Design in End User Developed Applications", Journal of Information Systems Education, 1989.
51. [8] Fowler, M. (2000), "Put Your Process on a Diet", Software Development".