Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

*IJCSMC, Vol. 2, Issue. 5, May 2013, pg.262 – 267*

SURVEY ARTICLE

# A Survey on Software Development Life Cycle Models

**T Bhuvaneswari[1], S Prabaharan[2]**

[1]PG Student, Department of Computer Science and Engineering, VMKV Engineering College, India

[2]Associate Professor, Department of Computer Science and Engineering, VMKV Engineering College, India

[1] *basweety4@gmail.com*

***Abstract— This study deals with a vital and important thing in computer software development. It is concerned with the software management processes that examine the area of software development through the development models, which are known as software development life cycle. It represents the development models namely Waterfall model, Iterative model, V-shaped model, Spiral model, Extreme programming, Iterative and Incremental Method, Rapid prototyping model, The Chaos Model, Adaptive Software Development (ASD), The Agile Software Process (ASP), Crystal, Dynamic System Development Method (DSDM), Feature Driven Development (FDD), Rational Unified Process (RUP), SCRUM, Wisdom, The Big Bang Model. These models have advantages and disadvantages as well. Therefore, the main objective of this study is to represent different models of software development and make comparison between them to show the features and defects of each model.***

***Key Terms: - Software Engineering; Software Development; Software Process Model, SDLC; Software Life Cycle Model***

## I. INTRODUCTION

No one can deny the importance of computer in our life, especially during the present time. In fact, computer has become indispensable in today's life as it is used in many fields of life such as industry, medicine, commerce, education and even agriculture. It has become an important element in the industry and technology of advanced as well as developing countries. Now-a-days, organizations become more dependent on computer in their works as a result of computer technology. Computers considered a time-saving device and its progress helps in executing complex, long, repeated processes in a very short time with a high speed. In addition to using computer for work, people use it for fun and entertainment. Noticeably, the number of companies that produce software programs for the purpose of facilitating works of offices, administrations, banks, etc., during the previous four decades. Moreover, the aim of software engineering is to construct programs of high quality.

## II. SOFTWARE PROCESS MODELS

A Software Development Life Cycle (SDLC) is a construction imposed on the development of a software product. It is frequently considered a subset of systems development life cycle. There are a number of models for such processes, each describing approaches to a range of tasks or activities that take place during the process. This section represents the following models of software development and makes comparison between them to show the features and defects of each model**.**

- *Waterfall model.*
- *Iterative model.*

 

- *V-shaped model.*
- *Spiral model.*
- *Extreme programming.*
- *Incremental Method*
- *Rapid prototyping model*
- *The Chaos Model*
- *Adaptive Software Development (ASD)*
- *The Agile Software Process (ASP)*
- *Crystal Model*
- *Dynamic System Development Method (DSDM)*
- *Feature Driven Development (FDD)*
- *Rational Unified Process (RUP)*
- *SCRUM*
- *Wisdom*
- *The Big Bang Model.*

*A. The Waterfall Model*

The waterfall model [1] is the classical model of software engineering. This model is one of the oldest models and is widely used in government projects and in many major companies. As this model emphasizes planning in early stages, it ensures design flaws before they develop. In addition, its intensive document and planning make it work well for projects in which quality control is a major concern**.** The model begins with establishing system requirements and software requirements and continues with architectural design, detailed design, coding, testing, and maintenance. **Advantages:** Easy to understand and implement. It reinforces good habits such as define-before-design, design-before-code. It identifies deliverables and milestones, Document driven, Published documentation standards, Works well on mature products and weak teams. **Disadvantages:** Idealized doesn't match reality well. It doesn't reflect iterative nature of exploratory development. It is unrealistic to expect accurate requirements so early in project. Software is delivered late in project, delays discovery of serious errors. It is difficult to integrate risk management. It is expensive to make changes to documents. For small teams and projects, the cost is more and significant administrative overhead.

*B. Iterative Model*

An iterative life cycle model [3] does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model. **Advantages:** In iterative model we are building and improving the product step by step, we can track the defects at early stages. This avoids the downward flow of the defects. In iterative model we can get the reliable user feedback. In iterative model less time is spent on documenting and more time is given for designing. **Disadvantages:** Each phase of iteration is rigid with no overlaps, Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire life cycle.

*C. V-Shaped Model*

Like waterfall model, the V-Shaped life cycle [2] is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing is emphasized in this model more than the waterfall model. The testing procedures are developed early in the life cycle before any coding is done, during each of the phases preceding implementation. Requirements begin the life cycle model just like the waterfall model. Before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in requirements gathering. The high-level design phase focuses on system architecture and design. An integration test plan is created in this phase in order to test the pieces of the software systems ability to work together. **Advantages:** Simple and easy to use. Each phase has specific deliverables. Higher chance of success over the waterfall model due to the early development of test plans during the life cycle. Works well for small projects where requirements are easily understood. **Disadvantages:** Very rigid like the waterfall model. Little flexibility and adjusting scope is difficult and expensive. Software is developed during the implementation phase, so no early prototypes of the software are produced. This model does not provide a clear path for problems.

 

*D. Spiral model*

The spiral model [4] 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 spiral, starting in the planning phase, requirement is gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. Requirements are gathered during the planning phase. 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. Software is produced in the engineering phase, along with testing at the end of the phase. The evaluation phase allows the customer to evaluate the output of the project to data before the project continues to the next spiral. In the spiral model, the angular component represents progress, and the radius of the spiral represents cost. **Advantages:** High amount of risk analysis. Good for large and mission-critical projects. 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. It doesn't work well for smaller projects.

*E. Extreme programming.*

An approach to development [5] based on the development and delivery of very small increments of functionality. It relies on constant code improvement, user involvement in the development team and pair wise programming. It can be difficult to keep the interest of customers who are involved in the process. Team members may be unsuited to the intense involvement that characterizes agile methods. Prioritizing changes can be difficult where there are multiple stakeholders. Maintaining simplicity requires extra work. Contracts may be a problem as with other approaches to iterative development. **Advantages:** Lightweight methods suit small-medium size projects. Produces good team cohesion and emphasises final product and Iterative. Test based approach to requirements and quality assurance. **Disadvantages:** Difficult to scale up to large projects where documentation is essential and needs experience and skill if not to degenerate into code-and-fix. Programming pairs is costly.

*F. Incremental Method*

It is developed to overcome the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during development of earlier parts or versions of the system [6]. **Advantages:** Produces business value early in the development life cycle· Better use of scarce resources through proper increment definition. Can accommodate some change requests between increments. More focused on customer value than the linear approaches. Problems can be detected earlier. **Disadvantages:** Requires heavy documentation. Follows a defined set of processes, defines increments based on function and feature dependencies. It requires more customer involvement than the linear approaches. Partitioning the functions and features might be problematic. Integration between iteration can be an issue if this is not considered during the development.

*G. Rapid prototyping model or Rapid Application Development (RAD)*

A rapid prototype [7] is a working model that is functionally equivalent to a subset of the product. Because the working prototype has been validated through interaction with the client, the resulting specification will be correct. Verification is needed in specification, planning, and design. In implementation and integration, testing is needed. An essential aspect of a rapid prototype is in the word *rapid*. We can combine waterfall and rapid prototyping, by using rapid prototyping to find out the client's requirements. **Advantages:** RAD reduces the development time and reusability of components help to speed up development. All functions are modularized so it is easy to work with. **Disadvantages:** For large projects RAD require highly skilled engineers in the team. Both ends Customer and developer should be committed to complete the system in a much abbreviated time frame. If commitment is lacking RAD will fail. RAD is based on Object oriented approach and if it is difficult to modularize the project the RAD may not work well.

*H. The Chaos Model*

The Chaos model [8] combines a linear problem solving loop with fractals to describe the complexity of software development. The linear problem-solving loop involves four different stages: problem definition, technical development, solution integration, and status quo. Fractals describe the structure between different parts of a project. The Chaos model differs from other models in that it imposes little organization on the development process; rather, it allows many organizations to evolve. This allows the Chaos model to apply in many complex situations. The structure of a simple problem is different from the structure of a more complex

 

problem. In general, we break complex problems into simpler sub problems. We use this reductions approach to deal with problems that are too large to handle otherwise. Yet, stating that the recursive structure is too simple.

*I. Adaptive Software Development (ASD)*

Adaptive Software Development (ASD) as a framework from which to address the rapid pace of many software projects [9]. ASD is grounded in the science of complex adaptive systems theory and has three interwoven components: the Adaptive Conceptual Model, the Adaptive Development Model, and the Adaptive Management Model. In contrast to the typical waterfall (plan, build, implement) or the iterative (plan, build, revise) life cycles, the adaptive development life cycle (speculate, collaborate, learn) acknowledges the existence of uncertainty, change and does not attempt to manage software development using precise prediction and rigid control strategies.

*J. The Agile Software Process (ASP)*

The Agile Software Process (ASP) was first proposed at the 1998 [10] unlike traditional software process models based on volume, the ASP is time-based and quickly delivers software products. The model accomplishes this by integrating lightweight processes, modular process structures, and incremental and iterative process delivery. The ASP methodology offers five major contributions to the field. These include: A new process model with a time-based inaction mechanism. A software process model that provides evolutional delivery. A software process architecture that integrates concurrent and asynchronous processes. ASP is a complex process and is therefore more vulnerable to disruption than are other lightweight and traditional SDLC methodologies. Benefits of the ASP process are its ability to efficiently manage large-scale software development efforts. Evidence of this is the 75 percent reduction in development cycle time realized by Fujitsu when ASP was employed to manage a major communication software project.

*K. Crystal Model*

The Crystal family of lightweight SDLC methodologies is the creation of Alistair Cockburn [17]. Crystal is comprised of more than one methodology because of Cockburn's belief that differing project types require differing methodologies. Project types are classified along two lines: the number of people on the development team and the amount of risk (e.g. a 30 person project that is at risk to lose discretionary money requires a different methodology than a four person life-critical project).Crystal methodologies are divided into colours-coded bands. "Clear" Crystal is the smallest and lightest. "Yellow", "Orange", "Red", "Maroon", "Blue", and "Violet" follow for use with larger groups using more complex methodologies.

*L. Dynamic System Development Method (DSDM)*

The Dynamic Systems Development Method (DSDM) is a framework [12] used to control software development projects with short timelines. It was developed in 1994 by a consortium formed by a group companies in Great Britain. The methodology begins with a feasibility study and business study to determine if DSDM is appropriate. The rest of the process consists of three interwoven cycles. These are functional model iteration, design and build iteration, and implementation. The underlying principles of DSDM include frequent deliveries, active user communication, empowered development teams, and testing in all phases of a project. DSDM is different than traditional approaches in that requirements are not fixed. Project requirements are allowed to change based upon a fixed timeline and fixed project resources. This approach requires a clear prioritization of functional requirements. Emphasis is also put on high quality and adapting to changing requirements. It has the advantage of a solid infrastructure (similar to traditional methodologies), while following the principles of lightweight SDLC methods.

*M. Feature Driven Development (FDD)*

Feature Driven Development (FDD) is a model-driven short-iteration software development process [13]. The FDD process starts by establishing an overall model shape. This is followed by a series of two-week "design by feature, build by feature" iterations. FDD consists of five processes: develop an overall model, build a features list, plan by feature, and design by feature, and build by feature. There are two types of developers on FDD projects: chief programmers and class owners. The chief programmers are the most experienced developers and act as coordinator, lead designer, and mentor. The class owners do the coding. One benefit of the simplicity of the FDD process is the easy introduction of new staff. FDD shortens learning curves and reduces the time it takes to become efficient. Finally, the FDD methodology produces frequent and tangible results. The method uses small blocks of user-valued functionality. In addition, FDD includes planning strategies and provides precision progress tracking.

 

*N. Rational Unified Process (RUP)*

The Rational Unified Process (RUP) works well with cross-functional projects [16]. RUP contains six best practices: manage requirements, control software changes, develop software iteratively, use component-based architectures, visually model, and verify quality. RUP is a process framework and can be used in either a traditional (e.g. waterfall style) or a lightweight manner. One example of the model's flexibility is the do process developed by Robert Martin. The dX process is identical XP and is a fully compliant instance of RUP. The process was designed for developers that have to use RUP, but would prefer to use XP. Finally, although RUP was originally intended to help manage software projects, its flexible design makes it applicable to large e-business transformation projects. After applying a few critical augmentations to the process, RUP can effectively provide a framework for enterprise-wide e-business transformation.

*O. SCRUM*

A SCRUM is a Rugby team of eight individuals [14]. The team acts together as a pack to move the ball down the field. Teams work as tight, integrated units with a single goal in mind. In a similar manner, the SCRUM software development process facilitates a team focus. SCRUM is a light SDLC methodology for small teams to incrementally build software in complex environments. SCRUM is most appropriate for projects where requirements cannot be easily defined up front and chaotic conditions are anticipated. SCRUM divides a project into sprints (iterations) of 30 days. Functionality is defined before a sprint begins. The goal of the process is to stabilize requirements during a sprint.

*P. Wisdom*

The White-water Interactive System Development with Object Models [15] addresses the needs of small development teams who are required to build and maintain the highest quality interactive systems. The Wisdom methodology has three key components: A software process based on user-centered, evolutionary, and rapid-prototyping model. A set of conceptual modelling notations that support the modelling of functional and non-functional components. A project management philosophy based on tool usage standards and open documentation. Wisdom is comprised of three major workflows: requirements workflow, analysis workflow, and design workflow. In addition, the methodology is based on seven models and uses four types of diagrams. Task flow plays an important role in Wisdom and corresponds to a technology-free and implementation-independent portrayal of user intent and system responsibilities.

*Q. The Big Bang Model*

The Big- Bang Model [16] is the one in which huge amount of people or money is put together, a lot of energy is expended and out comes the perfect software product or it doesn't. The beauty of this model is that it's simple. There is little planning, scheduling, or formal development process. All the effort is spent developing the software and writing the code. It is an ideal process if the product requirements aren't well understood and the final release date is flexible. It is also important to have flexible customers, too, because they won't know what they're getting until the very end.

*R. Code and Fix*

"Code and fix" development [11] is not so much a deliberate strategy and schedule pressure on software developers without much of a design in the way, programmers immediately begin producing code. At some point, testing begins (often late in the development cycle), and the inevitable bugs must then be fixed before the product can be shipped.

## III. CONCLUSION

In this paper various software development life cycle models are studied and compared. The Waterfall model provides base for other development models. The advantages and disadvantages of enhanced models such as Iteration model, V-shaped model, Spiral model, Extreme programming, Evolutionary Prototyping Model, Iterative and Incremental Method, Rapid prototyping model, The Chaos Model, Adaptive Software Development (ASD), The Agile Software Process (ASP), Crystal, Dynamic System Development Method (DSDM), Feature Driven Development (FDD), Rational Unified Process (RUP), SCRUM, Wisdom, The Big Bang Model, and code and fix models are compared.

 

REFERENCES

[1] Royce, Winston, "Managing the Development of Large Software Systems", Proceedings of IEEE WESCON 26, 1970.
[2] Kevin Forsberg and Harold Mooz, "The Relationship of System Engineering to the Project Cycle," in Proceedings of the First Annual Symposium of National Council on System Engineering, October 1991: 57–65.
[3] Craig Larman and Victor R. Basili,"Iterative and Incremental Development: A Brief History". IEEE Computer (IEEE Computer Society) 36 (6): 47–56. Doi:10.1109/MC.2003.1204375. ISSN 0018-9162. Retrieved 2009-01-10.], June 2003.
[4] Boehm B, "A Spiral Model of Software Development and Enhancement", ACM SIGSOFT Software Engineering Notes", "ACM", 11(4):14-24, August 1986.
[5] Mohamed Sami Abd EI-Satar" Software Development Life Cycle Models and Methodologies", 2012, http://melsatar.wordpress.com
[6] Craig Larman and Victor Basili, Iterative and Incremental Development: A Brief History, IEEE Computer, June 2003.
[7] C. Melissa McClendon, Larry Ragout, Gerri Akers: The Analysis and Prototyping of Effective Graphical User Interfaces. October 1996.
[8] ACM Digital Library, The chaos model and the chaos cycle, ACM SIGSOFT Software Engineering Notes, Volume 20 Issue 1, and Jan 1995.
[9] Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, High smith, J.A., and 2000 New York: Dorset House, 392pp, ISBN 0-932633-40-4.
[10] International Conference on Software Engineering in Kyoto Japan (Aoyama, 1998a).
[11] McConnell, Steve. "7: Lifecycle Planning". Rapid Development. Redmond, Washington: Microsoft Press.
[12] Rietmann: DSDM in a bird's eye view, DSDM Consortium, p. 3-8 (2001).
[13] Coad, P., Lefebvre, E. & De Luca, J. (1999). Java Modeling In Color With UML: Enterprise Components and Process. Prentice Hall International. (ISBN 0-13-011510-X).
[14] Linda Rising and Norman S.Janoff, AG Communication Systems, "The Scrum Software Development Process for Small Teams, IEEE Software July/August 2000.
[15] Nuno Jardim Nunes, João Falcão e Cunha" Wisdom - Whitewater Interactive System Development with Object Mode Version 4.0 / 21 April 2000.
[16] Wollack, Edward J. "Cosmology: The Study of the Universe". Universe 101: Big Bang Theory. NASA, 2010
[17] Ernest Mnkandla, "About Software Engineering Frameworks and Methodologies", IEEE AFRICON 2009.

 