International Journal of Electronics Communication and Computer Engineering
Volume 4, Issue (6) NCRTCST-2013, ISSN 2249–071X

National Conference on Recent Trends in Computer Science and Technology (NCRTCST)-2013

Suitability Analysis of Various Software Development Life Cycle Models

Apoorva Mishra
Computer Science & Engineering
C.S.I.T, Durg, India
apoorvamishra@csitdurg.in

Deepty Dubey
Computer Science & Engineering
C.S.I.T, Durg, India
deeptydubey@csitdurg.in

***Abstract* — In this current era of software development, a large number of life cycle models are available for the systematic development of computer software and projects. SDLC models give a theoretical guide line regarding development of the software. SDLC models are very important for developing the software in a systematic manner such that it will be delivered within the time deadline and should also have proper quality. These models have their own unique characteristics and are suited to a particular situation of software development and software types. One software life cycle model may prove to be more efficient than the other one depending upon the development environment. In this paper, the work has been done to analyze the various software life cycle models from this aspect. Selecting proper SDLC allows the project managers to regulate whole development strategy of the software. Each SDLC has its advantages and disadvantages according to which we decide which model should be implemented under which circumstances. In this paper we will present a comprehensive study of different life cycle models like- waterfall model, rapid application development (RAD), prototype model, spiral model, incremental model and extreme programming (XP) .**

***Keywords* – Software Development Life Cycle (SDLC), Suitability Analysis, Phases of the Life Cycle, Milestones.**

I. Introduction

Software development life cycle (SDLC), is a way of developing the software in a smooth manner. SDLC also increases the probability of completing the software project within the time deadline and maintaining the quality of the software product as per the requirement. The System Development Life Cycle framework provides a sequence of activities for system designers and developers to follow for developing software. It is often considered as a part of system development life cycle. The software development process is divided into phases that allow a software organization to organize its work. All software projects go through the phases of requirements gathering, business analysis, system design, implementation, and quality assurance testing[1]. Employing any SDLC model is often a matter of personal choice entirely dependent on the developer. Each SDLC has its strengths and weaknesses, and each SDLC may provide better functionalities in different situations. Jovanovich D. et al. [1] presented basic principles and comparison of software development models. Rodriguez-Martinez et al. [2] focused on lifecycles frameworks models and detailed software development life cycles process and reports the results of a comparative study of Software development life cycles. One life cycle model theoretically may suite particular conditions and at the same time other model may also look fitting into the requirements but one should consider trade-off while deciding which model to choose [3]. Davis A.M. et al. [4] provided a framework that can serve as a basis for analyzing the similarities and differences among alternate life-cycle models as a tool for software engineering researchers to help describe the probable impacts of a life-cycle model and as a means to help software practitioners decide on an appropriate life-cycle model to utilize on a particular project or in a particular application area. A software development life cycle model is broken down into distinct activities and specifies how these activities are organized in the entire software development effort. In this paper a thorough study of the different life cycle models has been presented and on the basis of that a suitability analysis of different types of models for various projects has been done.

II. Phases Involved in SDLC Model

The phases involved in software development life cycle model are-

1. Requirements gathering.
2. Requirement analysis
3. high level design
4. low level design
5. Implementation
6. Testing
7. Deployment & maintenance.

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.

The common phases of an SDLC can be represented by the following diagram-

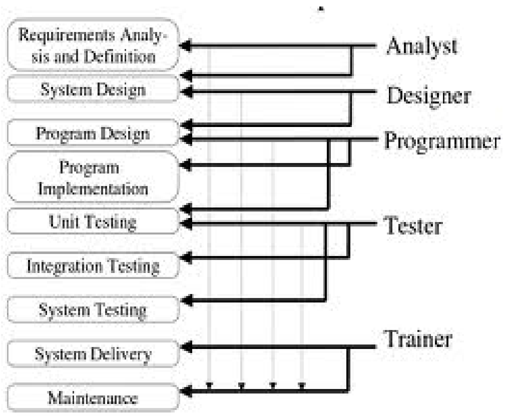

Fig.1. Phases of an SDLC model.

III. Software Development Life Cycle Models

Waterfall Model

Waterfall model was proposed by Royce in 1970 which is a linear sequential software development life cycle (SDLC) model. The various phases followed are **r**equirements analysis, design, coding, testing and implementation in such a manner that the phase once over is not repeated again and the development does not move to next phase until and unless the previous phase is completely completed**.** Hence it is not very much useful when the project requirements are dynamic in nature.

Waterfall strengths

1. Easy to understand, easy to use.
2. Provides structure to inexperienced staff.
3. Milestones are well understood.
4. Sets requirements stability.
5. Good for management control (plan, staff, track).
6. Works well when quality is more important than cost or schedule.

Waterfall weaknesses

1. All requirements must be known upfront
2. Deliverables created for each phase are considered frozen – inhibits flexibility
3. Can give a false impression of progress
4. Integration is done in one big bang manner at the end
5. Little opportunity for customer to preview the system (until it may be too late).

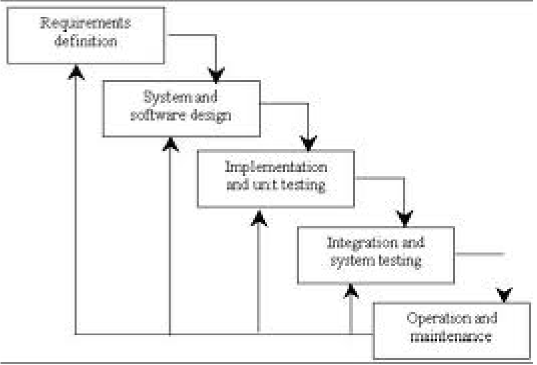
Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Fig.2. Waterfall Model

Spiral Model

Spiral model adds risk analysis and 4gl RAD prototyping to the waterfall model. Each cycle involves the same sequence of steps as the waterfall process model.

Spiral Quadrant:

Objectives: functionality, performance, hardware/ software interface, critical success factors, etc. Alternatives: build, reuse, buy, sub-contract, etc. Constraints: cost, schedule, interface, etc. Study alternatives relative to objectives and constraints. Identify risks (lack of experience, new technology, tight schedules, poor process, etc. Resolve risks (evaluate if money could be lost by continuing system development. In develop next-level product the typical activities are Create a design, Review design, Develop code, Inspect code,Test product. In plan next phase the typical activities are Develop project plan, Develop configuration management plan, Develop a test plan, Develop an installation plan.

Spiral Model Strengths

1. Provides early indication of insurmountable risks, without much cost
2. Users see the system early because of rapid prototyping tools
3. Critical high-risk functions are developed first
4. The design does not have to be perfect
5. Users can be closely tied to all lifecycle steps
6. Early and frequent feedback from users

Spiral Model Weaknesses

1. Time spent for evaluating risks is too large for small or low-risk projects
2. The model is complex
3. Risk assessment expertise is required
4. Spiral may continue indefinitely
5. Developers must be reassigned during non-development phase activities

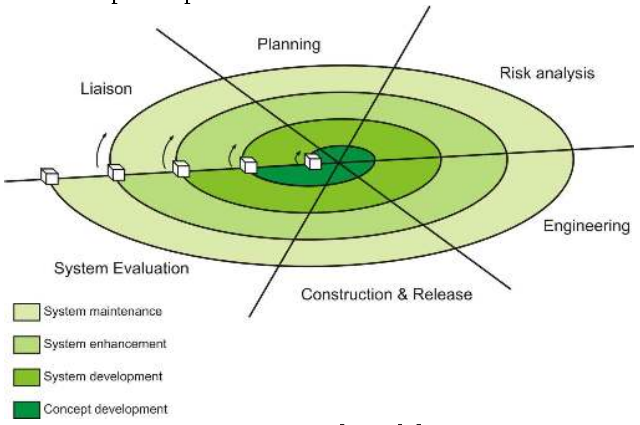

Fig.3. Spiral Model

RAD Model

Phases of RAD are Requirements planning phase, User description phase, Construction phase, Cutover phase. If requirements are well understood and project scope is constrained, the Rapid application development (RAD) figure 4 process enables a development team to create a "fully functional system" within very short time periods (e.g., 60 to 90 days).

RAD Strengths

1. Reduced cycle time and improved productivity with fewer people means lower costs
2. Time-box approach mitigates cost and schedule risk
3. Customer's involvement throughout the complete cycle minimizes risk of not achieving customer satisfaction and business needs
4. Uses modeling concepts to capture information about business, data, and processes.

RAD Weaknesses

1. Risk of never achieving closure
2. Hard to use with legacy systems
3. Requires a system that can be modularized
4. Developers and customers must be committed to rapid-fire activities in an abbreviated time frame.

RAD can be implemented in projects for which time line is aggressive, risk is not so high and the size of the project is Small or medium. RAD can prove to be very effective for projects with time limit between 60 to 90 days.

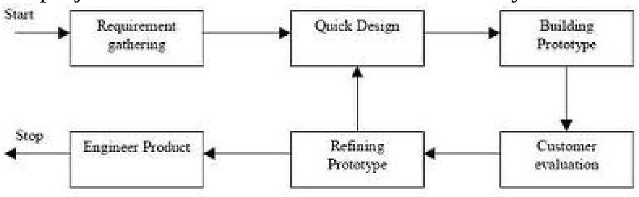

Fig.4. RAD Model

Incremental Model

In incremental model we construct a partial implementation of a total system, then slowly keep on adding further functionality. The incremental model prioritizes requirements of the system and then implements them in groups. Each subsequent release of the system adds functionality to the previous release, until all designed functionalities have been implemented.

Incremental Model Strengths

1. Develop high-risk or major functions first
2. Each release delivers an operational product
3. Customer can respond to each build
4. Uses "divide and conquer" breakdown of tasks
5. Initial product delivery is faster

Incremental Model Weaknesses

1. Requires good planning and design
2. Requires early definition of a complete and fully functional system to allow for the definition of increments
3. Well-defined module interfaces are required (some will be developed long before others)

Prototype Model

Software prototyping is the process of creating an incomplete model of the future full-featured software. The processes involved are, Identify basic requirements, Develop Initial prototype, Review with customers users, Revise and enhance.

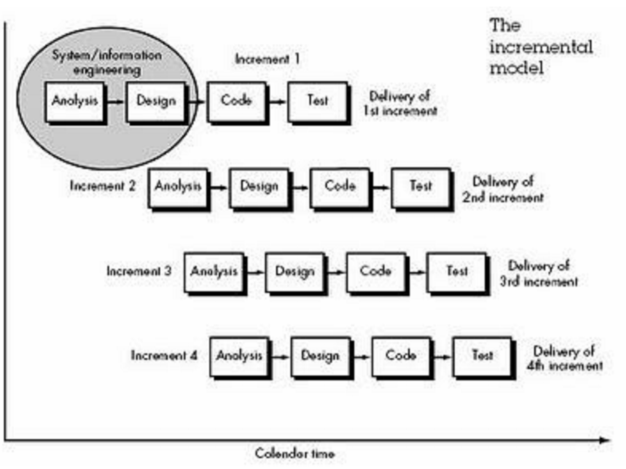

Fig.5. Incremental Model

The types of prototypes are,Throwaway prototype: process of creating a rapid model (demonstration or marketing), Evolutionary prototype: build a system then refine., Incremental: final product built as separate prototypes. As the cost of change increases with time the prototype model provides the opportunity to the customer to give feedback at an early stage which saves a lot of cost.

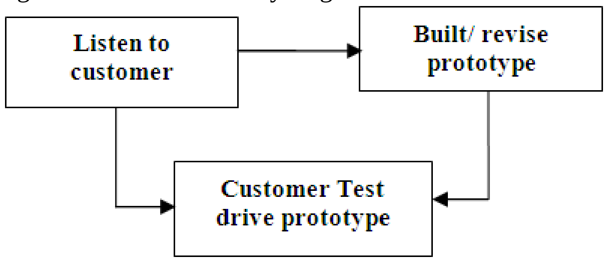

Fig.6. Prototype Model

Benefits of the model are, Misunderstandings between software users and developers are exposed, Missing services may be detected and confusing services may be identified, A working system is available early in the process, The prototype may serve as a basis for deriving a system specification, The system can support user training and system testing.

XP MODEL

In XP the programming is done in pairs. It involves test driven development, continuous planning, change and delivery. No overtime is required.

XP strengths

1. Lightweight methods suit small-medium size projects
2. Produces good team cohesion
3. Emphasises final product

XP weaknesses

1. Difficult to scale up to large projects where documentation is essential
2. Needs experience and skill if not to degenerate into code-and-fix
3. Pair programming is costly.

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. Different SDLC models have their unique characteristics and requirements. On the basis of these aspects, this paper presents an analysis survey on the suitability of various SDLC models on the situation of its use for software project development. 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.

Table 1: Suitability Analysis of Various SDLC Models

SDLC	Type of Situation in which it is Suitable
Waterfall	-Fixed requirements. -Work proceeds to completion in a linear manner. -Without any time aggressiveness. -It can not handle High Risks Project
XP	-Small projects. -Full time user representative -It will not work if the team will not be able to predict the cost and schedule
RAD	-Time line is aggressive. -Risk is not so high. -Small to medium size projects.
Prototype	-Developers get to build something immediately. -It cannot handle high level of risks
Incremental	-High technical risks. -Time line is aggressive.
Spiral	-Large –scale system and software. -Level of Reliability is High. -where reaction to Risks at each evolutionary level is required.

V. Conclusion

There are many SDLC models such as, Waterfall, RAD, spiral, incremental, XP, Prototype etc. used in various organizations depending upon the conditions prevailing in it. All these different software development models have their own advantages and disadvantages. In the Software Industry, the hybrid of all these methodologies is used i.e with some modification. In this paper, the work has been done to analyze the various SDLC models with respect to their suitability for development of various types of software projects depending upon the development environment. Selecting the correct life cycle model is extremely important in a software industry & the suitability analysis of the SDLC models according to the project type will help in the selection of the proper SDLC model for a particular project.

References

[1] 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.
[2] 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.
[3] Roger Pressman, titled "Software Engineering - a practitioner's approach".
[4] 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
[5] Sanjana Taya, Shaveta Gupta, "Comparative Analysis of Software Development Life Cycle Models".
[6] Vishwas Massey, K.J Satao, " Comparing Various SDLC Models And The New Proposed Model On The Basis Of Available Methodology".
[7] Klopper, R., Gruner, S., & Kourie, D. "Assessment of a framework to compare software development methodologies".
[8] Fowler, M. (2000), "Put Your Process on a Diet", Software Development".
[9] Sandeep Kumar et al, "Ontology Development and Analysis for Software Development Life Cycle Models".
[10] Manish Sharma, "A Survey of project scenario impact in SDLC models selection process".