Volume No 03, Special Issue No. 01, March 2015 ISSN (online): 2348 – 7550

A COMPARISON BETWEEN DIFFERENT TYPES OF SOFTWARE DEVELOPMENT LIFE CYCLE MODELS IN SOFTWARE ENGINEERING

Mr. Ashish Kumar Gupta

Assistant Professor, Dept. of C.S.E., I.T.S. Engineering College, Greater Noida, U.P., (India)

ABSTRACT

This research deals with a vital and important issue in computer world. At present the software systems can not be built with mathematical or physical certainty so they are not perfect upto the level. 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. In this research paper the comparison between various software development models has been carried out. Each and every model of SDLC has its own advantages and limitations so in this research we have to describe each model on behalf of various important features. Various SDLC models like waterfall, iterative, prototype model and spiral model were suggested.

Keywords: Software Development Life Cycle, Phase of SDLC Models, Software Development Process, Comparison, Four Models.

I INTRODUCTION

SDLC stands for Software Development Life Cycle. A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and lifecycle management of an application or piece of software.

Software engineering is the study and an application of engineering to the design, development, and maintenance of software. Typical formal definitions of software engineering are: "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software". It is the application of engineering to software because it integrates significant mathematics, computer science and practices whose origins are in Engineering.

There are various processes and methodologies have been developed over the last few decades to improve software quality, with varying degrees of success. However, it is widely agreed that no single approach that will prevent project failures in all cases.

A software development process is a structure imposed on the development of a software product. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process. Software Engineering processes are composed of many activities, notably the following:

- Requirements Analysis
- Specification
- Software architecture
- Implementation
- Testing
- Documentation
- Training and Support
- Maintenance

1.1 General Software Process Models are

Even though there are number of models each software Development Company adopts the best-suited model, which facilitates the software development process and boosts the productivity of its team members.

- Waterfall model: Separate and distinct phases of specification and development.
- Prototype model.
- Rapid application development model (RAD).
- Evolutionary development: specification, development and validation are interleaved.
- Incremental model.
- Iterative model.
- Spiral model.
- Component-based software engineering : The system is assembled from existing components.

There are many variants of these models e.g. formal development where a waterfall-like process is used, but the specification is formal that is refined through several stages to an implementable design.

II DIFFERENT PHASES OF SDLC

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced according to the design which is called development phase. After coding and development the testing verifies the deliverable of the implementation phase against requirements. The basic activities or phases to be performed for developing software system are-

- Determination of System's Requirements
- Design of system
- Development (coding) of software
- System Testing

Fig1. Software Development Life Cycle

III SOFTWARE DEVELOPMENT MODELS

3.1 Waterfall Model

The waterfall model is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production / Implementation and Maintenance.

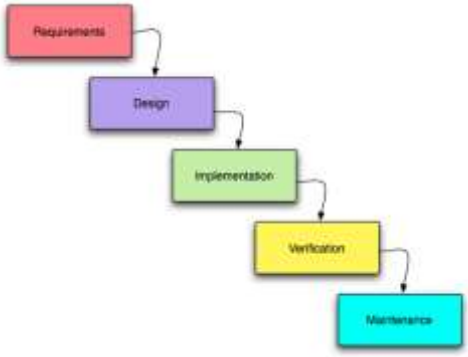

Fig2. Waterfall Model

1. **Basic Principles**
   - Problems can be solved more easily if they are more clearly defined.
   - Large amounts of code are more traceable if they are structured.
   - A good project life-cycle plan improves the development process.
   - System documentation is a byproduct of the development process, and is not done later, as an afterthought.
2. **Advantages of Waterfall Model**
   - This model is simple and easy to understand and use.
   - It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
   - In this model phases are processed and completed one at a time. Phases do not overlap.
   - Waterfall model works well for smaller projects where requirements are very well understood.

3. Disadvantages of Waterfall Model

- 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.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.

3.2 Iterative Model

An iterative life cycle model 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.

1. Basic Principles

- Manage requirements not tasks, based on use cases and nonfunctional requirements.
- Manage to meet business goals, due dates and budgets. Be willing to change requirements to fit these, not the other way around.

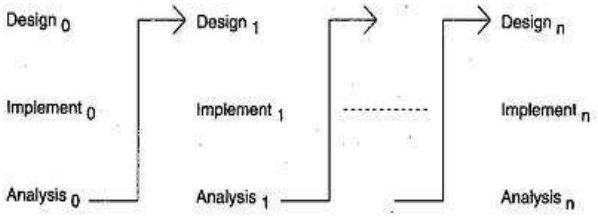

Fig3. Iterative Model

- Begin with a simple implementation of a subset of the requirements that demonstrates key aspects of the system.
- Design around isolated, easy-to-find modules that group small sets of related requirements. Complete or re-code one module per iteration.
- Work in short cycles (1-6 weeks) composed of overlapping phases: requirements, design, programming, testing.
- During the iteration, the external customer or project manager cannot change the scope for that iteration, but the development team may change the scope by dropping features if the end date will not be met.
- Any difficulty in design, coding and testing a module should signal the need for redesign or re-coding.
- Modifications should become easier to make as the iterations progress. If not, redesign is needed.

2. Advantages of Iterative Model

- It is much better model of the software process.
- It allows feedback to proceeding stages.

- It can be used for project wherein the requirements are not well understood.

3. Disadvantages of Iterative Model

- Each phase of an 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 lifecycle.
- No clear milestones in the development process.

3.3 Prototyping Model

The basic idea here is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system.

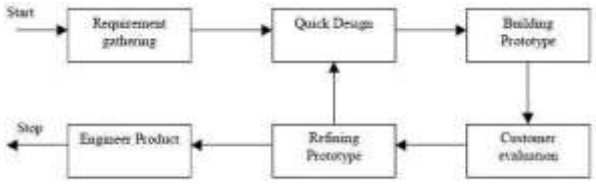

Fig4. Prototyping Model

1. Basic Principles

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Not a standalone, complete development methodology, but rather an approach to handling selected parts of a larger, more traditional development methodology.
- Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system.

2. Advantages of Prototyping Model

- Users are actively involved in the development.
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Confusing or difficult functions can be identified.

3. Disadvantages of Prototyping Model

- Possibility of causing systems to be left unfinished.
- Possibility of implementing systems before they are ready.

3.4 Spiral Model

The spiral model is a risk-driven processmodel generator for software projects. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping.

1. Basic Principles

- Focus is on cost and risk assessment throughout the life cycle.
- Useful for Long-term project commitment unwise because of potential changes to economic priorities.
- Users are unsure of their needs.
- Requirements are complex.

2. Advantages of Spiral Model

- High amount of risk analysis hence, avoidance of Risk is enhanced.

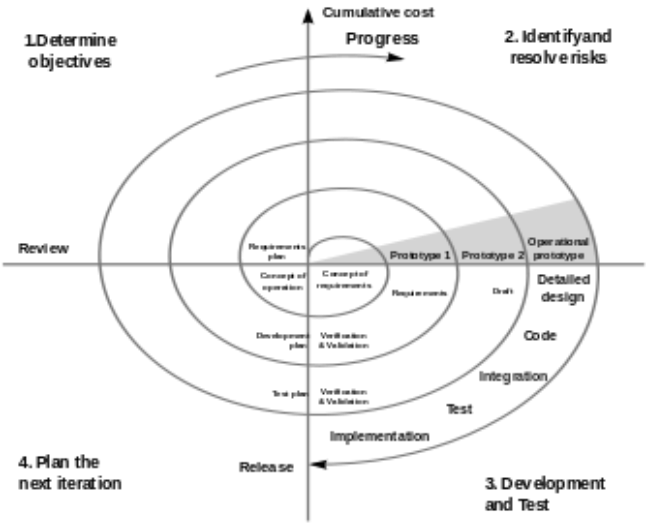

Fig5. Spiral Model

- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the software life cycle.

3. Disadvantages of Spiral Model

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

4. Comparative Analysis of SDLC Models

Features	Waterfall Model	Iterative Model	Prototyping Model	Spiral Model
Requirements Specification	Beginning	Beginning	Frequently Changed	Beginning
Cost	Low	Low	High	Expensive
Simplicity	Simple	Intermediate	Simple	Intermediate
Expertise Required	High	High	Medium	High
Risk Involvement	High	Easily Manage	Low	Low
Overlapping Phases	No	No	Yes	Yes
Flexibility	Rigid	Less Flexible	Highly Flexible	Flexible
Maintenance	Least Glamorous	Promoted Maintainability	Routine Maintenance	Typical
Reusability	Limited	Yes	Weak	High
Documentation Required	Vital	Yes	Weak	Yes
User Involvement	Only At Beginning	Intermediate	High	High
Cost Control	Yes	No	No	Yes
Resource Control	Yes	Yes	No	Yes
Guarantee of success	Less	High	Good	High

Table1. Comparison of SDLC Models

V CONCLUSION

SDLC models are tools that allow the development team to correctly follow the SDLC steps to create software that meets a business need. Each SDLC model has evolved as a new technology and has addressed weaknesses of older models. After analysis of all models through the various factors in this research, it is concluded that:

1. There are many existing SDLC models for developing systems with different requirements.
2. Waterfall model is used by various big companies for their internal projects.
3. Most commonly used models for developing systems are waterfall model and spiral model.
4. Each model has advantages and disadvantages for developing the systems, so each new model tries to eliminate the disadvantages of previous model.

REFERENCES

[1]. Ms. Shikha Maheshwari, Prof. Dinesh Ch. Jain, "A Comparative Analysis of Different types of Models in Software Development Life Cycle", ISSN: 2277 128X (online), vol.2 Issue 5, may 2012.

[2]. Nabil Mohammed Ali Munassar1 and A. Govardhan "A Comparison Between Five Models Of Software Engineering" International Journal of Computer Science Issues, Vol. 7, Issue 5, September 2010.

[3]. Ashish B.S..., Dr.Vinay C.., "Survey of Software Life Cycle Models by Various Documented Standards"IJCST Vol. 2, Iss ue 4, Oct . - Dec. 2011.

[4]. Chan, D.K.C. Leung, K.R.P.H, "Software development as a workflow process", 2-5 Page(s): 282 – 291, Dec. 1997.

[5]. Sanjana Taya "Comparative Analysis of Software Development Life" ISSN:2229-4333(print),ISSN:0976-8491(online),vol.2ISSUE 4,Oct-Dec2011.

[6]. Dr. Deepshikha Jamwal "Analysis of software Development Models", ISSN: 2229-4333(print),ISSN:09768491 (online), vol.1ISSUE 2, Decemder 2010.

[7]. Maglyas, A.; Nikula, U.; Smolander, K.,"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.

[8]. Swapanja Ingale "Comparative Study Of Software Development Model" International conference on advances in computing &management 2012.

[9]. Rothi, J.,Yen, D, "System Analysis and Design in End User Developed Applications", Journal of Information Systems Education, 1989.

[10]. S. M. Metev and V. P. Veiko, Laser Assisted Microtechnology, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

[11]. K. Schwalbe(2009), "Information Technology Project Management", 6th Edition, Cengage Learning.