IOSR Journal of Engineering (IOSRJEN)
ISSN: 2250-3021 Volume 2, Issue 7(July 2012), PP 21-24
www.iosrjen.org

Comparative Study of Prototype Model For Software Engineering With System Development Life Cycle

Rajendra Ganpatrao Sabale, Dr. A.R. Dani
Student of Ph.D., Singhania University, Pacheri Bari, Dist. Jhunjhunu (Rajasthan), India
International Institute of Information Technology, Pune (Maharashtra), India

Abstract: - *The concept of system lifecycle models came into existence that emphasized on the need to follow some structured approach towards building new or improved system. Many models were suggested like waterfall, prototype, rapid application development, V-shaped etc. In this paper, we focus on the comparative analysis of these Software Development Life Cycle Models. With the growing operations of organizations, the need to automate the various activities increased. So, it was felt that some standard and structural procedure or methodology be introduced in the industry so that the transition from manual to automated system became easy.*

I. Introduction

There are several models for such processes, each describing approaches to a variety of activities that take place during the process. ISO 12207 is an ISO standard for describing the method of selecting, implementing and monitoring the life cycle for software. Any software development process is divided into several logical stages that allow a software development company to organize its work efficiently in order to build a software product of the required functionality within a specific time frame and budget. Rodriguez-Martinez et al. 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 that permits a plausible explanation of their evolution in terms of common, distinctive, and unique elements as well as of the specification rigor and agility attributes. For it, a conceptual research approach and a software process lifecycle meta-model are used. Jovanovich D. et al. presented basic principles and comparison of software development models. First part is the presentation of development models and second part introduces a practical approach to implement one of the Software development models. Finally, the problem of determining the most suitable Software development model in the case of developing PC applications. Davis A.M. et al. 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. Sharma B. et al. presented the Comparative Analysis of Software Process Improvement models. Software process improvement is recognized as an important part of the software development life cycle. This study also provided simulation of the existing models like Capability Maturity Model, ISO etc. and analyzes each model along with their importance and drawbacks. Maglyas A. et al. described that the size and complexity of software development projects are growing, at the same time proportion of successful projects is still quite low. The objective of study is to compare two existing models of success prediction i.e. The Standish Group and McConnell models and to determine their strengths and weaknesses and the results show that The Standish Group has a tendency to overestimate the problems in a project. McConnell predicts successful projects pretty well but underestimates the percentage e of unsuccessful projects. Rothay et al. provided a brief review of traditional SDLCs, they related how the use of traditional software development models is numerous and often regarded as the proper and disciplined approach to the analysis and design of software applications. Osborn et al. discussed traditional SDLC techniques and how over time the phases of these approaches have become enshrined in a development cycle that includes defining requirements, designing a system to meet those requirements, coding and testing. A Software Development Life Cycle Model is a set of activities together with an ordering relationship between activities performed in a manner that satisfies the ordering relationship that will produce desired product. SDLC Model is an abstract representation of a envelopment process. In a software development effort the goal is to produce high quality software. The development process is, therefore, the sequence of activities that will produce such software. 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 response to traditional approaches to software development, new lightweight methodologies have appeared. A high percentage of software development efforts have no process and might best be described as a chaotic “code and fix” activity. Light SDLC techniques are compromise between no process and too much process. The nine types of lightweight SDLC methodologies

are Adaptive Software Development (ASD), Agile Software Process (ASP), Dynamic System Development Method (DSDM), Extreme Programming (XP), Feature Driven Development (FDD), Rational Unified Process (RUP) etc.

Activities involved Software Development life cycle model:

Problem solving in software consists of these activities:

1. Understanding the problem
2. Deciding a plan for a solution
3. Coding the planned solution
4. Testing the actual program

For large systems, each activity can be extremely complex and methodologies and procedures are needed to perform it efficiently and correctly. Furthermore, 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 basic activities or phases to be performed for developing a software system are

1. Determination of System's Requirements
2. Design of system
3. Development (coding) of software
4. System Testing

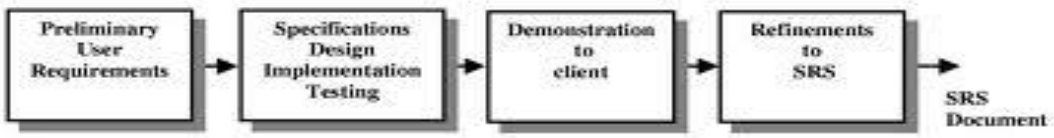

Fig. 2 Prototyping model

In this feasibility analysis phase, the feasibility of the project is analyzed, and a business proposal is put forth with a very general plan for the project and some cost estimates. Once the business proposal is accepted or the contract is awarded, the development activities begin starting with the requirement analysis.

II. Comparative Analysis:

Waterfall Model is easy to manage due to the rigidity of the model as each phase has specific deliverables and a review process. It works well for smaller projects where requirements are very well understood. V-shaped Model has higher chance of success over the waterfall model due to the development of test lans during the life cycle. It works well for small projects where requirements are easily understood. CMM Model provides more detailed coverage of the product life cycle than other process-improvement products used alone. CMM provides an opportunity to eliminate the stovepipes and barriers that typically exist in different parts of an organization and that typically are not addressed by other process-improvement models. CMM, which integrates software engineering and systems engineering into product engineering, is a valuable tool for many organizations that produce software only solutions RUP Model is a complete methodology in itself with an emphasis on accurate documentation. It is proactively able to resolve the project risks associated with the client's evolving requirements. Less time is required for integration as the process of integration goes on throughout the software development life cycle. The development time required is less due to reuse of components.

Prototype Model places more effort in creating the actual software instead of concentrating on documentation. This way, the actual software could be released in advance. Prototyping requires more user involvement and allows them to see and interact with a prototype allowing them to provide better and more complete feedback and specifications. The presence of the prototype being examined by the user prevents many misunderstandings that occur when each side believe the other understands what they said. The final product is more likely to satisfy the user's desire for look, feel and performance. 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.JAD Model can be successfully applied to a wide range of projects like new systems, enhancements to existing systems, System conversions, Purchase of a system etc. In Agile Model face to face communication and continuous inputs from customer representative leaves no space for guesswork. The end result is the high quality software in least possible time duration and satisfied customer. Comparison between different SDLC models in relation to their features like requirements, cost, resource control, risk involvement, changes incorporated, time frame, interface, reusability etc.

Model / Features	**Waterfall**	**V – Shape**	**CMM**	**RUP**	**Prototype**	**Incremental**	**Spiral**	**RAD**	**JAD**	**Agile**
Requirement Specifications	Beginning	Beginning	At second level	Beginning	Frequently changed	Beginning	Beginning	Time-box released	prototyped	Frequently changed
Understanding Requirements	Well Understood	Easily understood	Easily understood	Difficult to understand	Not well understood	well understood	Well understood	Easily understood	Easily understandable	Well understood
Cost	Low	expensive	high	expensive	High	low	expensive	low	expensive	Very high
Guarantee of Success	Low	High	High	Not guaranteed	Good	High	High	Good	Low but for long period	Very high
Resource Control	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No
Cost Control	Yes	Yes	varies	Yes	No	No	Yes	Yes	Yes	Yes
Simplicity	Simple	Intermediate	intermediate	Simple and clear	simple	Intermediate	Intermediate	very simple	Simple	Complex
Risk Involvement	High	Low	Varies acc to level	Critical risks in the early stages	low	Easily manage	Low	Very low	Varies	reduced
Expertise Required	High	medium	Varies acc to level	Yes	Medium	High	high	Medium	High	Very high
Changes Incorporated	Difficult	difficult	Medium	Easy	Easy	Easy	Easy	Easy	Medium	Difficult
Risk Analysis	Only at beginning	Yes	Yes	Yes	No risk analysis	No risk analysis	Yes	Low	Yes	Yes
User Involvement	Only at beginning	at the beginning	Only at beginning	at the beginning and at the last phase	High	Intermediate	High	Only at the beginning	in the design and development	High
Overlapping Phases	No such phase	No	No	yes	Yes	No	Yes	No	No	Yes
Flexibility	Rigid	Little flexible	Highly flexible	considerable	Highly flexible	Less flexible	Flexible	High	Flexible	Highly flexible
Maintenance	Least glamorous	Least	typical	Promote maintainability	Routine *maintenance*	Promotes maintainability	Typical	Easily maintained	Rigorously at all times.	Promotes maintainability
Integrity & Security	vital	limited	Limited	Very important	Weak	Robust	High	Vital	limited	Demonstrable
Reusability	limited	To some Extent	Yes	Supports reusability of the existing classes	Weak	Yes	Yes	some extent	limited	Use Case *Reuse*
Interface	minimal	minimal	Crucial	User interface	crucial	Crucial	Crucial	minimal	crucial	*model*-driven
Documentation &Training required	vital	yes	Yes	Yes	weak	Yes	Yes	limited	limited	Yes
Time Frame	Long	acc to project size	Quite long	short *time frame*	Short	Very long	Long	Short	medium	least possible

III. Conclusion

There are many SDLC models such as Agile, RAD and Waterfall etc. used in various organizations depending upon the conditions prevailing in it like v-model gives the verification and validation for organization and it is very useful for organization. All these different software development models have their own advantages and disadvantages. Nevertheless, in the contemporary commercial software development world, the fusion of all these methodologies is incorporated. Timing is very crucial in software development. If a delay happens in the development phase, the market could be taken over by the competitor. Also if a 'bug' filled product is launched in a short period of time (quicker than the competitors), it may affect the reputation of the company.

There should be a tradeoff between the development time and the quality of the product. Customers don't expect a bug free product but they expect a user-friendly product that results in Customer Ecstasy!

Reference

[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.

[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.

[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

[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

[5] 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.

[6] Osborn, C. SDLC, JAD, RAD,"Center for Information Management Studies", 2001.

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