Scientific Journal of Impact Factor (SJIF): 3.134

ISSN(Online): 2348-4470
ISSN(Print) : 2348-6406

International Journal of Advance Engineering and Research Development

Volume 1, Issue 11, November -2014

IMPROVING THE SOFTWARE DEVELOPMENT EXPERIENCE BY REDUCING EFFORT, COST AND RESOURCE ALLOCATION BY COMBINING VARIOUS SOFTWARE DEVELOPMENT APPROACHES

Nagajan Tarkhala[1], Prof. Chintan Kanani[2]

[1]*Computer Engineering, Darshan Institute of Engineering & Technology, nagarjun.jadeja3393@gmail.com*
[2] *Computer Engineering, Darshan Institute of Engineering & Technology, chintan.kanani@darshan.ac.in*

Abstract *— At Present all major routine work is being converted electronically, so there is a certain need about considering Software Engineering as an emerging and growing domain. It is not practically or physically possible that flow of software generation can be mapped with its mathematical output. In present all software systems are imperfect as there is always probation in development without any fixed certainty. According SDLC each and every model has the advantage and drawbacks so in this research we have to calculate the performance of each model on behalf of some important features. This paper categorizes and examines a few methods for relating or modeling how software systems are developed. It starts with environment and definitions of default software life cycle models that dictate most book talking and current software development approaches. In this paper, we are going to compare various software development models using various parameters to show the features and defects of each model. At end we are suggesting some points that should be helpful for organizations to generate software reducing manpower and resources on the same project retaining same quality to gain maximum profit from it. The major concentration of the paper is to provide software development model with less efforts and less resources with high quality output and for that we are going to combine EP, spiral and RAD approach to get better results in software generations.*

Keywords- *Software Development Process, SDLC, phase of SDLC, Spiral, EP, RAD*

I. INTRODUCTION

No one can deny the importance of computer in our life, especially during the present time. In fact, computer has become indispensible in today's life as it is used in many fields of life such as industry, medicine, commerce, education and even agriculture. Now a day, organizations become more dependent on computer in their works as a result of computer technology [1]. Computer is considered a time-saving device and its progress helps in executing complex, long, repeated processes in a very short time with a high speed. A software development process, also known as a software development life cycle (SDLC), is a structure imposed on the development of a software product. It is often considered as a subset of system development life cycle. Software Engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering g to software because e it integrates significant mathematics, computer science and practices whose origins are in Engineering [2]. Different processes and methodologies have been urbanized over the last few decades to improve software class. However, it is broadly approved that no single approach that will prevent project overruns and failures in all cases. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. A descriptive model shows the narration of how a particular software system was developed [3][1]. It may be used as the basis for accepting and improving software development processes or for building empirically grounded prescriptive models. A prescriptive model defines how a new software system should be developed. Prescriptive models are used as guidelines or frameworks to organize and structure how software development activities should be performed. Normally, it is easier and more common to eloquent a prescriptive life cycle model for how software systems should be developed. This is possible since most such models are perceptive or well reasoned. This means that many individual details that describe how a software system is built in practice can be ignored, generalized, or deferred for later consideration [4][5]. Software development teams, taking into account its goals and the scale of a particular project, and have a number of well-established software development models to choose from. Therefore, 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 [6]. So, according to the requirement, team selects approach and same as we are going to suggest that team should use more than one approach or combined approach for developing the same software. So that as a result, they need not to compare all categories of their requirements to generate model [7][10][18][20].

II. VARIOUS PHASES OF SDLC

After gone through reading of many literatures and papers, the listed out many things. In this report, the proposed approach is based on generating common software development model for all conditions. The approach or life cycle used to make the effective software step by step is known as SDLC (Software Development Life Cycle) [8][9].
SDLC is consists of many steps that as follows:

- **System Initiation**
  Here System is initiated for development i.e. it is a phase of requirement gathering where requirement is gathered by various sources from the authenticated party [11].
  .
- **Requirement Specification and Analysis**
  After gathering all the requirements related to new project, requirement must be specified and analyzed properly. Various feasibility studies have been carried out for the project as economical, technical and market feasibility has been taken into consideration. If it has been found all the way feasible then it has taken under work. [12]

- **Functionality specification and prototyping**
  After analyzing requirements, it is needed to define the functions of the project and make out the list of various functions that has been performed by the project or system. After specifying the functionality for the system, it is been needed to define a prototype for each and every function that it should work for such prototype.

- **Partitioning and Choosing**
  After defining prototype the requirement or the prototypes are grouped into modules that modules are the set of functions that are work for same category. Finally, the option is choosing the best module to start the work with the project. [13]

- **Model design and configuration specification**
  After specifying the modules i.e. after dividing project into modules now it has been needed to define development model with considering many features as cost, effort, time, resource and skill of developers etc.

- **Detailed model design specification**
  After choosing the model for the development, it is been defined perfectly to work throughout the project.

- **Model implementation**
  After defining and designing the model it needs to be implemented using any language on any standard platform.

- **Model Debugging**
  After implementing model, it is been debugged to check the possible errors that can be the consequences of the implementation phase.

- **Model Integration and Testing**
  After successfully debugging all the models, they all are integrated to make a single software system and that single software system is being tested using many testing methods [14].
  - Black box testing
    Non technical terms to be considered and software is checked according to general user of the system.
  - White box testing
    Here testing is done on the base of technical matters, the navigation and the business logic and security of the software is being tested.

- **Deployment and Installation**
  After testing application successfully, project is being deployed to the users in three versions.
  - Alpha Version - The software is tested by the company people who have developed it.
  - Beta version - Software is tested by the general public and opinions are collected about the software usage and user friendliness.
  - Full version - After success of beta and Alpha versions of software, the software is opened for public usage by touching finally.

- **Training and use**
  Before open up for the public use, public needed to be trained to use the software and generally that is been done by user manuals and usage videos [15].

- **Maintenance**

After deploying application, company needs to monitor its issues and usage and public reviews and need to go through the issues and try to solve the issues in upcoming version of the software.

III. SOFTWARE DEVELOPMENT MODELS IN BRIEF

A Programming process model is a conceptual illustration to explain the process from a particular viewpoint. There are numbers of general models for software processes, like: Waterfall model, Iterative development, Prototyping, RAD, XP, Spiral and Win-win Spiral etc [1][3][5][7][14][15].

3.1. Waterfall model

The waterfall model 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. [1][14]

3.2. Iterative model

The troubles with the Waterfall Model formed a claim for a new process of developing systems which could give faster results, require less up-front information, and propose larger flexibility. With Iterative development, the projects are divided into minute sections. This makes the team to display results earlier on in the process and get important feedback from users.[2][4]

3.3. Prototyping

The novel purpose of a prototype is to allow users of the software to evaluate developers' ideas for the plan of the ultimate product by really trying them out, rather than having to take to mean and estimate the design based on metaphors. [3]

3.4. Agile Development

Agile term refers to a group of software development methodologies based on iterative development, where requirements and solutions progress via partnership between self-organizing cross-functional teams [5]. It uses iterative approach as a basis but refers a lighter and more people-centric perspective than traditional approaches.

3.5. Rapid Application Development (RAD) model

RAD is a software development style, which positive discrimination iterative development and the rapid construction of prototypes instead of large amounts of up-front planning. The setting up of software developed using RAD is interleaved with writing the software itself. The shortage of extensive pre-planning generally allows software to be written more rapidly, and makes it easier to modify requirements. [6]

3.6. Extreme Programming (XP)

XP is planned to develop software quality and openness in varying customer requirements. Rest essentials of XP includes of programming in pairs or doing general code assessment, unit testing of whole code, ignoring programming of features until they are really required, a flat supervision structure, plainness and clarity in code, expecting changes in the customer's requirements with time and the trouble is better understood, and regular communication with the customer and among programmers [2][3][5].

3.7. Spiral model

A formal software system development spiral model, which combines some key aspect of the waterfall model and rapid prototyping methodologies, in an effort to combine advantages of top-down and bottom-up concepts [1][2][3][5]. It provided emphasis in a key area many felt had been neglected by other methodologies: deliberate iterative risk analysis and it is particularly suited to large-scale complex systems.

3.8. Win-Win spiral

The Win-Win spiral approach is an extension of the spiral approach. The only difference is that at the time of the identifying the requirements, the development team and the customer hold discussion and negotiate on the requirements that need to be included in the current iteration of the software [3][4][6][9].

IV. COMPARISON OF VARIOUS MODELS BASED ON STANDARD PARAMETERS

In Previous section we discussed about various approaches to develop software with many options and parameters. Now in this section we need to compare those papers with respect to almost all parameters to come to know that which model is good to use in all cases or what improvement should be there to work with the models as a part of research. We are going to combine various approaches to make an efficient model that is suitable for all kinds of requirements of software development. Here comparison is based on many parameters but the highlighted parameters are risk analysis, man power, resource consumption etc [1][2][4][7][8].

Features	WF	IT	Pro	Agile	EP	RAD	SP	WSP
Requirement Specification	B	B	FC	B	FC	FC	B	B

Understanding Requirements	WU	NU	NU	WU	WU	NU	WU	WU
Cost	L	L	A	A	M	H	A	L
Time	H	H	H	M	L	L	A	A
Risk	H	M	A	A	L	L	L	L
Errors	L	H	M	A	L	H	L	L
Man power	M	M	A	M	M	L	L	L
Resources	H	H	H	M	A	A	L	L
Reusability	N	N	Y	Y	Y	Y	Y	Y
Complexity	L	L	L	M	A	H	H	H
Success Ratio	L	L	M	M	M	H	H	H
Merging Phases	N	N	N	Y	Y	Y	Y	Y
Flexibility	N	Y	N	Y	Y	Y	Y	Y
User Involvement	L	A	M	H	A	H	H	H
Changes	N	N	Y	Y	Y	Y	Y	Y
Size of Project	S	S	W	S	W	S	W	W
Expertise	N	N	Y	N	Y	Y	Y	Y
Cost Control	N	N	Y	Y	Y	Y	Y	Y
Resource Control	N	N	Y	Y	Y	Y	Y	Y
Manpower Control	N	N	N	N	Y	Y	Y	Y

Table 1: Comparison of various s/w models with various parameters

Terms for Comparison table:

Y-Yes N-No **L**-Less **H**-High **WU**-Well Understood **FC**- Frequently Changed
B-At Beginning **A**- Average **M**-Moderate **NU**- not well understood **S**-Small **W**-Big Project
WF – Waterfall **IT**-Iterative **SP**-Spiral **Pro**-Prototyping **WSP**-Win-win Spiral
RAD-Rapid Application Development **EP**-Extreme Programming

V. PROPOSED WORK

In previous sections, we compared various software development models with respect to various 20 parameters. After completing that much resources and comparing based on that the spiral, win-win spiral, RAD and Agile models are very good to develop software products and projects. All the approaches have some merits and drawbacks.

Advantages of win-win spiral:
- Good in organizing manpower
- It's a risk driven approach
- Good in organizing resources[3]

Disadvantages of win-win spiral:
- Takes time
- Requirement must specified at beginning
- Cost and effort parameters not considered[3]

Advantages of RAD:
- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.[5]

Disadvantages of RAD:
- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills[5]

Advantages of EP:
- Perfect project making without errors
- High quality projects
- Resource requirement is less[11]

Disadvantages of EP:
- It cost high as experts are involved in development
- It makes project unfeasible all the time[11]

Here by forgetting the disadvantages and concerning advantages by mixing up two or more models to work efficiently and errorless is our concern. Now our focus is to suggest a new approach that is hybrid from many approaches. Our focus is based on some parameters that are resource allocation, man power allocation, risk reduction and bug reduction. Win-

win spiral model can efficiently organize working staff with less number of members to a less number of resources for the same project compared to any other software development model with good quality as well. RAD develops projects with a speed [5]. It is used when we need to complete project in urgency and application is started developing based on requirements. Optional stages are excluded from the process as experts are working so no need to work with designing and all. Of course the cost of development due to expert staff is high undoubtedly. But it is good for short term and urgent projects.
In general, win-win spiral model is good for allocation of manpower and resources [2] [3] and RAD is good for fast execution of stages and to remove unnecessary stages. So by combining both the approach, we will definitely get the third approach that is better in performance from both previous approaches as we are considering only the advantages of the both system model by forgetting the disadvantages. So, the new proposed software model works with almost all types of requirements and able to reduce cost and manpower and resource allocation with less errors/ bug in project i.e. much efficient in concern with time. So the improvement is justified and software development experience will be improved.

5.1 New approach for software development:
Our goal is to make such a software engineering model that makes the software with less effort and less resources with same quality and less time. So, here we are going to combine various software models as we know that for small modules extreme programming is useful as it takes less time and less efforts. And RAD model skips the unnecessary steps in making of software. Win-win spiral model is risk driven approach when you feel some risk at any stage or module, one can directly prevent it by discarding that stage or redesign and regenerate that module again [1].
So, in this research we are proposing new approach that takes only advantages of above discussed models.
That means, Extreme programming for less time and less effort and high quality and RAD approach for completing project in less time and scheduling and Spiral model for risk estimation at every stage of module development.

Now we are going to present new model steps as below:

5.2 Proposed Steps:

1. Requirement Specification and analysis
2. Dividing into small modules
3. Discard module designing phase (Advantage of RAD)
4. Assign a module to Expert programmers (Advantage of EP)
5. Implement and test module (Advantage of Code and Fix model)
6. Check risk on completing the every stage. (Advantage of win-win spiral model)
7. Deployment and maintenance

Note: it is not necessary that one need to follow all steps one by one. One can work in parallel. So at the end by following the above proposed steps the future software will have less time, high quality and less effort that is what the goal of this research to reduce errors and resource allocation and efforts. And the practical implementing of the approach will be shortly.

5.3 Way of Implementation
The above mentioned approach has been implemented based on live research and implementation approach. The research dataset has been gathered from any standard software organization and firstly implemented on the base of existing models such as spiral model or agile development model. And after implementing the software based on existing models, the various parameters like effort estimation, resource allocation, time estimation and quality management specially. The graph of the above mentioned parameters has been drawn. And the same approach will be taken for the implementation of new proposed model and the same graph will be generated for effort estimation, resource allocation, time estimation and quality management. And finally by comparing both the software based on standard parameters, the improvement has been stated.

VI. SUMMARY

At the end, after comparing and studying various approaches of software development, we came to know certain facts and parameter effects in software generation. The research goal is to make such a software engineering model that makes the software with less effort and less resources with same quality and less time. So, here we are going to combine various software models as we know that for small modules extreme programming is useful as it takes less time and less efforts. And RAD model skips the unnecessary steps in making of software. Win-win spiral model is risk driven approach when you feel some risk at any stage or module, one can directly prevent it by discarding that stage or redesign and regenerate that module again. So, in this research, we are proposing new approach that takes only advantages of the Spiral, EP and RAD models. Extreme programming provides high quality by extensive code review whereas RAD approach for

completing project in less time and scheduling and win-win Spiral model is used for risk estimation at every stage of module development with the win conditions of customers and developers both. By following the new model the software generation will be in less time with high quality and less effort and less resources. So this new approach will be useful the IT industry a lot.

REFERENCES

[1] Mr. Preyash Dholakia, Mr. Dishek Mankad, "The Comparative Research on Various Software Development Process Model", International Journal of Scientific and Research Publications, Volume 3, Issue 3, March 2013, ISSN 2250-3153

[2] Ms. Shikha maheshwari, Prof.Dinesh Ch. Jain, "A Comparative Analysis of Different types of Models in Software Development Life Cycle", IJARCSSE, Volume 2, Issue 5, May 2012

[3] Barry Boehm, Dan Port, Ye Yang," Win-Win Spiral Approach to Developing COTS-Based Applications", EDSER-5 Position Paper, University of Southern California, Texas A&M University

[4] Walt Scacchi, "Process Models in Software Engineering", Institute for Software Research, University of California, Irvine, February 2001, Revised Version, May 2001, October 2001, Final Version to appear in, J.J. Marciniak (ed.), Encyclopedia of Software Engineering, 2nd Edition, John Wiley and Sons, Inc, New York, December 2001.

[5] Nabil Mohammed Ali Munassar and A. Govardhan, "A Comparison between Five Models Of Software Engineering", IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 5, September 2010, ISSN (Online): 1694-0814

[6] Barry Boehm, edited by Wilfred J. Hansen, "Spiral Development: Experience, Principles and Refinements, Spiral Development Workshop", February 9, 2000, July 2000

[7] C. Abts, B. Boehm, and E. Bailey Clark, "COCOTS: A Software COTS-Based System (CBS) Cost Model" Proceedings, ESCOM 2001, April 2001, pp. 1-8.

[8] C. Albert and L. Brownsword, "Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview" CMU-SEI-2002-TR-009, July 2002.

[9] R. Balzer, "Living with COTS" Proceedings, ICSE 24, May 2002, p. 5.

[10] Gupta, A., I.S. Mumick, "Maintenance of Materialized Views: Problems, Techniques, and Applications." Data Eng. Bulletin, Vol. 18, No. 2, June 1995. 9 Zhuge, Y., H. Garcia-Molina, J. Hammer, J. Widom, "View Maintenance in a Warehousing Environment, Proc. Of SIGMOD Conf., 1995.

[11] Roussopoulos, N., et al., "The Maryland ADMS Project: Views R Us." Data Eng. Bulletin, Vol. 18, No.2, June 1995. [11] O'Neil P., Quass D. "Improved Query Performance with Variant Indices", To appear in Proc. of SIGMOD Conf., 1997.

[12] Ian Sommerville, "Software Engineering", Addison Wesley, 7th edition, 2004.

[13] CTG. MFA – 003, "A Survey of System Development Process Models", Models for Action Project: Developing Practical Approaches to Electronic Records Management and Preservation, Center for Technology in Government University at Albany / Suny,1998 .

[14] Steve Easterbrook, "Software Lifecycles", University of Toronto Department of Computer Science, 2001.

[15] National Instruments Corporation, "Lifecycle Models", 2006, http://zone.ni.com.

