International Journal of New Technology and Research (IJNTR)
ISSN:2454-4116, Volume-2, Issue-4, April 2016 Pages 153-155

A Study of Software Development Models for Small Organisations

Deepshikha Jamwal, Devanand

***Abstract*— Software Development is a complex and often difficult process requiring the synthesis of many disciplines, like modelling and design to code generation, project management, testing, deployment, change management and beyond. Software development organizations follow some process while developing a software product. A key component of any software development process is the lifecycle model on which the process is based.**

***Index Terms*— Software Development Process, Software Development Models.**

I. INTRODUCTION

The main objective of software process research is to improve software development practice by proposing:

a) Better ways of designing the developer organization processes.

b) Better ways of improving the organization at the level of individual processes and the organization as a whole. To this end, there are two lines of software process studies, software process modelling and software process evaluation and improvement.

In theory, the two kinds of models should be similar or the same, but in practice, they are not. Building a process model and discussing its sub processes help the team understand this gap between what should be and what it is [1].

A. REASONS FOR MODELLING A PROCESS.

There are several other reasons for modelling a process:

- When a group writes down a description of its development process, it forms a common understanding of the activities, resources, and constraints involved in software development.

Creating a process model helps the development team find inconsistencies, redundancies, and omissions in the process and in the constituent parts.

The models should reflect the goals of development, such as building high-quality software, finding faults early in development, and meeting required budget and schedule constraints. As the models are built, the development team evaluates candidate activities for their appropriateness with these goals.

Every process should be tailored for the special situation in which it will be used. Building a process model helps the development team understand where that tailoring is to occur [3].

B. SOFTWARE DEVELOPMENT PROCESS MODELS:

Every software development process model includes system requirement as input and a delivered product as output. Many such models have been proposed over the years some of these models are discussed:

Code and Fix Model: "This basic model was used in the earliest days of software development" and is not formally documented due to its simplicity. The code and Fix model is often used by default. To use the code and Fix model, start with a general idea of what is to be built. The coding and fixing continue until the product is released or project is cancelled.

Waterfall Model: One of the early models proposed was the waterfall model, where the stages are depicted as cascading from one to another. As the figure implies, one development stage should be completed before the next begins. Thus, when all of the requirements are elicited from the customer, analyzed for completeness and consistency and documented, then the development team can go on to system design and development.

V-Process Model: The V model is a variation of the waterfall model that demonstrates how the testing activities are related to analysis and design. , Coding forms the pointed edge of the V, with analysis and design on the left arm of V & testing and maintenance on the right arm of the V. Unit and integration testing addresses the correctness of programs. The V model suggests that unit and integration testing also be used to verify the program design.

Spiral Model: Boehm viewed the software development process in the light of the risks involved, suggesting that a spiral model could combine development activities with risks management to minimize and control risks. The spiral model is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model. It provides the potential for rapid development of incremental versions of the software. In the spiral model, software is developed in a series of incremental releases.

Deepshikha Jamwal, Assistant Professor, Govt Degree College Udhampur, University of Jammu.
Prof Devanand, Central University Jammu

Table 1. Strengths and Weaknesses of Models.

STRENGTHS	Waterfall	Incremental	Spiral
Allows for work force specialization	Y	Y	Y
Orderliness appeals to management	Y	Y	Y
Can be reported about	Y	Y	Y
Facilitates allocation of resources	Y	X	Y
Early functionality		Y	Y
Does not require a complete set of requirements at the onset		Y(*)	Y
Resources can be held constant		Y	
Control costs and risk through prototyping			Y
WEAKNESSES			
Requires a complete set of requirements at the onset	Y		
Enforcement of non-implementation attitude hampers analyst/designer communications	Y		
Beginning with less defined general objectives may be uncomfortable for management		Y	Y
Requires clean interfaces between modules		Y	
Incompatibility with a formal review and audit procedure		Y	Y
Tendency for difficult problems to be pushed to the future so that the initial promise of the first increment is not met by subsequent products		Y	Y
(*) The incremental model may be used with a complete set of requirements or with less defined general objectives.			

Incremental Model: This model is similar to the waterfall model, but without the heavy documentation requirement (although this can be specified if required). Multiple functional product releases are made, with each release incrementally adding functionality or increasing performance. This model is also known as the 'Incremental Development' model or 'Staged Delivery' model. A slight variation of this model is to allocate specific module delivery to each stage, rather than a complete system. The architectural design phase identifies which modules are required for a formal release. The stages, in which these modules are to be delivered, are developed in parallel [4].

II. ASSUMPTIONS FOR THE MODEL

In order to develop a simple model that focuses on the more important aspects of the methodologies and the objective of this study, some assumptions were employed. These assumptions are as follows:

1. Constant number of tasks is not a frequent scenario. This assumption was made because this is a management issue and is independent of the methodology approach used and therefore, out of the scope of this study.

2. No delay or other factors affecting the motivation are considered. Unlike the previous assumption, changes in the motivation can dramatically impact the development speed and quality of a project. They can also be different in an iterative or sequential approach. However, since the model represents small projects it is reasonable to assume that the impact of changes on the motivation is not significant.

3. Tasks that need rework are only reworked in the current phase. In the theory, both iterative and sequential approaches contemplate the possibility of sending a task back to a previous phase. Several authors have studied how the cost increases as the project moves forward to fix a mistake. This increasing cost is caused by the overhead time to fix tasks from previous phases and by the additional rework generated by the tasks associated with errors. Although capturing of this effect would be beneficial to increase the accuracy of the model, it would require the creation of a specific set of levels for each phase, which, in turn, would increase dramatically the number of elements of the model and their relationships. For that reason, with the exception of the testing, this model considers that rework is only done in the current phase. However, the model does not keep track of the tasks mistakenly approved in the previous phase and use it as a variable to calculate the quality of the work done in the next phase [5].

III CONCEPTIONS FOR THE LIFECYCLE MODEL

The primary objective of the research work is to do the study of different software development models and to develop a framework to guide for identifying the most suitable lifecycle for the projects especially in small organizations.

The objective has been achieved by determining a set of factors like ***size of software***, ***software complexity***, ***required quality***, ***requirements volatility***, ***amount of documentation***, ***experience of personnel, personnel availability*** and ***project duration***, which mostly influence a software project. The commonly used lifecycle models have been identified and research work was carried out to identify how the strengths and weaknesses (attributes) of selected lifecycles influence these factors.

The data was obtained from experienced software professionals, primarily working either as software development professionals or having an advisory role within the commercial or public sectors. This suggests that the *lifecycle influencing factors data values* can be accepted with confidence and are applicable to a range of environment and projects [6, 7].

IV. CONCLUSION

In the existing work, the author has tried to purpose a conceptual model of information system for small organization. The model recreates these development considering two different development methodologies: The first is a ***Sequential waterfall-based*** approach & the second is an ***Iterative-based*** approach that gathers elements from agile and extreme programming methodologies. For these different

model parameters, ***model stocks***, ***main variables***, ***main flows*** are described, but the model is still not used for any software project in a small organization. So this work will be included in the future work.

ACKNOWLWDGEMENT

This work is being partially been supported by university of Jammu & various software companies. The authors would like to extend their thanks to IT companies and professionals for providing data, with the help of which the paper is framed.

REFERENCES

[1] 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.

[2] Jennifer Stapleton, DSDM – Dynamic Systems Development Method, ISBN 0-201-17889-3, Pearson Education, Harlow 1997.

[3] Molokken-Ostvold et.al, "A comparison of software project overruns - flexible versus sequential development models", Volume 31, Issue 9, Page(s): 754 – 766, IEEE CNF, Sept. 2005.

[4] Boehm, B. W. "A spiral model of software development and enhancement", ISSN: 0018-9162, Volume: 21, Issue: 5, on page(s): 61-72, May 1988.

[5] Deepshikha Jamwal, Vinod Sharma, Devanand, Mansotra V., " A critical analysis of software Usage vs. cost", presented and published in INDIACOM - ISSN no. 0973-7529 & ISBN 978-81-904526-25, 2009.

[6] Deepshikha, Pawanesh Abrol, Devanand, "Study & Analysis of Software Development and user satisfaction level", ICSTM, Springer link, 2009.

[7] Deepshikha jamwal, "Analysis of Software Development Models", IJCST Vol1, Issue2, ISSN 097-8491, December 2010.