Analysis on Software Development Approaches

Vijayalakshmi N.S.

Abstract - Software process models often represent a networked sequence of activities, objects, transformations, and events that embody strategies for accomplishing software evolution. Software systems come and go through a series of passages that account for their inception, initial development, productive operation, upkeep, and retirement from one generation to another.The objective of this paper is to analyze and categorize software development approaches and reveals where they can be beneficial in software industry with the type of project.

Index Terms- Software development, Process model, Code and fixed, Stepwise Refinement, Iterative and Incremental Development, Agile Development, Component Based Development, Component Driven Approach.

1. Introduction

In a software development effort the goal is to produce high quality software. Software development approaches are important, as it imposes consistency and structure on a set of activities. Its structure guides team actions by allowing them to examine, understand, control and improve the activities that comprise the process. The need of following and selecting a formal process for software development is to offer desired discipline to deliver a better quality product for success of business and to avoid wastage of money, time and demoralization in developers. The development process is, therefore, the series of activities that will produce such software. A software development process model is broken down into different activities. Basic activities of software engineering processes are as follows:

- Requirements Analysis

Extracting the requirements of a desired software product is the first task in creating it. While customers probably believe they know what the software is to do, it may require skill and experience in software engineering to recognize incomplete, ambiguous or contradictory requirements.

- Software architecture

The architecture of a software system refers to an abstract representation of that system. Architecture is concerned with making sure the software system will meet the requirements of the product, as well as ensuring that future requirements can be addressed.

- Coding and Testing

Reducing a design to code may be the most obvious part of the software engineering job, but it is not necessarily the largest portion. Testing software, especially where coded by two different engineers must work together falls to the software engineer.

- Implementation

Deployment of software in an organization is performed, once completed and its very important to have training classes for the most enthusiastic software users As large percentage of software projects fail because the developers fail to realize that it doesn't matter how much time and planning a development team puts into creating software if nobody in an organization ends up using it. People are occasionally resistant to change and avoid venturing into an unfamiliar area.

- Maintenance

Maintaining and enhancing software to cope with newly discovered problems or new requirements can take far more time than the initial development of the software. Not only may it be necessary to add code that does not fit the original design but just determining how software works at some point after it is completed may require significant effort by a software engineer.

This paper presents the analysis of software development approaches. The remainder of this paper is structured as follow: Section 2 explains the various software development approaches, with their strength and weakness. Section 3 highlight differences between various software process models and finally in section 4 represent the conclusions.

2. Software development approaches

The importance of software development process has direct various researchers to propose diverse approaches. Following are few software development approaches:

2.1 Code and fixed development

This approach is the most usual form of development for small companies. As the name depicts, it is a process in which the team perform coding for a product to be developed and at the same time fix the problems in the code. It is pretty flexible and often works well when there are three or fewer stakeholders. Code-and-fix development leads to costly communication problems.

2.2 The Stepwise Refinement

In this approach, software systems are developed through the progressive refinement and enhancement of high-level system specifications into source code components [1, 5].This model has been most effective and widely applied in helping to teach individual programmers how to organize their software development work. This approach is simple to understand and is highly scalable, as large systems can be developed in a structured and predictable fashion from it. The advantage of this approach is that it allows for incremental development but on a much finer level of granularity. It also uses unit tests as an integral feature of the development process. Unfortunately, it often leads to a solution where each module represents one part of the task in chronological terms, which can lead to multiple modules knowing the details of some data structures.

2.3 Iterative and Incremental Development

In iterative and incremental development process, the development process goes in cyclic base, which is called iteration. A new release includes new functionality, but existing functionality from the current release may have been enhanced. Frequently releases allow developer to fix unanticipated problems and team can focus on different area of expertise with different releases. It reduces risk and improves user satisfaction, as it involves user feedback. These process models are often used by the Spiral Model, with project risk reduced as each new increment or iteration is reviewed and enhanced. Iterative and Incremental development approaches have their advantages, software architects are still faced with the challenge of creating a reliable foundation. This type of development can create long term challenges that are significant in terms of cost and quality [2].

2.4 Agile Development

Agile software development processes are built on the foundation of iterative development. To that foundation they add a lighter, more people-centric viewpoint than traditional approaches. Agile processes use feedback, rather than planning, as their primary control mechanism. The feedback is driven by regular tests and releases of the evolving software. Agile processes seem to be more efficient than older methodologies, using less programmer time to produce more functional, higher quality software, but have the drawback from a business perspective that they do not provide long-term planning capability.

2.5 Component Based Development

Component based software developments (CBSD) are built by making considerable use of existing software components, instead of building software system from scratch; they are assembled from reusable software components. During component based software development, reusable artifacts have to be explicitly considered and the development process has to be tailored to appropriately echo the reusable approach. Intermediate stages in a reuse-oriented process: component analysis, requirement modification, system design with reuse and development and integration [4]. The main objective of CBSD is to reduce the overall cost of a software development. The problem with this approach is, there is no specific place during the development where the developer can consider reusing existing component and the lack of means to clearly identify the elements that constitute a CBSD process model that describe requirement, architecture, analysis, design, test and implementation along the development stream also makes the current CBSD models complicated to used [3].

2.6 Component Driven Approach

The software process model based on component driven development approach divide the development process into various phases like traditional process models, but the component driven approach differs in the way it implements the phases in development process. In component driven development approach each phase implements a model called phased model (3C-Model). The 3C-Model is a general model for generalizing the problem solving process in each phase of software development. This approach defines a generalized process, in which a problem specification is transformed to a solution by decomposing the problem into sub-problems that are independently solved and integrated into an overall solution. This consists of multiple steps or cycles. Each cycle in process corresponds to a transformation from one state to another, consisting of a problem specification state and a design state. After each state transformation, a sub-problem is solved. In addition a new sub-problem may be added to the problem specification state. Each transformation process involves an evaluation step whereby it is evaluated whether the design solutions so far (design state) are consistent with the initial requirements and if there are any additional requirements identified during the evaluation. In particular, this process includes an explicit phase for searching design alternatives in the corresponding solution space and selecting these alternatives based on explicit quality criteria [6]. 3C-Model assists Fusion Process Model in generalizing the process of solving the problems in each phase [7]. This makes the software development scope wider and provides firmer control over software development process. This is a new approach and not much practical implementation of this is known.

3. Comparison of Software Process Approaches:

Approaches	*Strength*	*Weakness*	*Type of projects*
Code and Fix Development	Software is rapidly built.	Particular is focus need on developing clear line of communication among all the parties involved. Transitioning out of this approach, can be difficult.	Mainly for small project.
Stepwise Refinement	Helps to impose structure on the development and highly scalable.	Often leads to a solution where each module represents one part of the task in chronological terms, which can lead to multiple modules knowing the details of some data structure.	For complex projects.
Iterative and Incremental Development	Developer can fix problems with frequent released and client feedback.	Long term challenges in terms of cost and quality.	For business where time is important, when risk of a long project can't be taken, when requirement are not known earlier.
Agile Development	Quick development approach and client is involved in development.	Less stress on analysis and design. No emphasis on designing and documentation.	Good for small projects, it become difficult to judge the effort and time required for the projects in the software development life cycle.

Component Based Software Development	Prebuilt, standardize software components are used, which reduce overall cost. Increased reliability.	There is no specific place during the development where the developer can consider reusing existing component and complicated to use.	Only for those projects which are based on reusable component development.
Component Driven Approach	Client is involved. Requirement changes can be handled until final delivery. Developer can track the time, cost and quality.	Not much practically applied.	For all kind of projects-small, medium and large.

4. Conclusion:

In this study, a review has been made on a number of software development approaches. Each approach has its strengths and weaknesses, derived out of the review. Most popular approach these days is Agile Process Approaches, but only skilled developers are involved as quick decisions are necessary for this process. These days component driven development approach is also in practice which is implemented by various process models like Fusion Process Model. Fusion Process Model use 3C-Model in its each phase, it can manage cost, time and quality of software project very efficiently. All the approaches has its own pros & cons, and the selection of software process model based on various other factors like type of project, time, cost , complexity etc.

References:

[1] A Mili, J. Desharnais and J.R.Gagne, "Formal Models of Stepwise Refinement of Programs", ACM Computing Surveys,18(3), pp. 231-276, 1986.

[2] D.Graham, "Increment development and delivery for large software system", IEEE Computer, 25(11), pp. 1-9, 1992.

[3] I.Jacobson, M.Griss and P.Jonsson, "Software Reuse Architecture, Process and Organization for Business Success", Addision-Wesley,1997.

[4] I.Sommerville, "Software Engineering", Tata McGraw-Hill, New York, 2005.

[5] N.Wirth, "Program Development by Stepwise Refinement", Communication of the ACM, 14(4), pp. 221-227, 1971.

[6] Rupinder Kaur and Jyotsna Sengupta, "Development and Analysis of 3C-Model for Software Development Lifecycle", IEEE 2nd International Conference on Computer Engineering and Technology (ICCET 2010),Volume 6, pp. 688-691, April 16 -18, Chengdu, China, 2010.

[7] Rupinder Kaur and Jyotsna Sengupta, "New Approach in Software Development - Fusion Process Model", Journal of Software Engineering and Applications, USA, 3(10), pp. 998-1004, October 2010.