Advances in Computational Research
ISSN: 0975-3273 & E-ISSN: 0975-9085, Volume 7, Issue 1, 2015, pp.-187-190.
Available online at http://www.bioinfopublication.org/jouarchive.php?opt=&jouid=BPJ0000187

TRADITIONAL Vs. MODERN SOFTWARE ENGINEERING - AN OVERVIEW OF SIMILARITIES AND DIFFERENCES

SHINDE L.K.*, TANGDE Y.S. AND KULKARNI R.P.

MCA Department, Marathwada Institute of Technology, Aurangabad- 431 028, MS, India.
*Corresponding Author: Email- shinde.laxmikant88@gmail.com

Received: December 18, 2014; Revised: January 05, 2015; Accepted: January 15, 2015

Abstract- Software Development Life Cycle (SDLC) is important part of every complete software project management. Most of the times success or failure of software project management is crucial and depends on chosen software life cycle. In Software Engineering there are many software development approaches. Most of them follows the systematic approach of development such as Traditional Software Development and others other categories don't follow the structure but their emphasis is on rapid and adaptive approach such as Agile Software Development and many more. This paper reviews traditional software development and agile software development and explains advantages and disadvantages of both the methodologies and finally the conclusion is presented.

Keywords- Software Development Life Cycle (SDLC), Traditional Software Development, Agile Software Development

Citation: Shinde L.K., Tangde Y.S. and Kulkarni R.P. (2015) Traditional vs. Modern Software Engineering - An Overview of Similarities and Differences. Advances in Computational Research, ISSN: 0975-3273 & E-ISSN: 0975-9085, Volume 7, Issue 1, pp.-187-190.

Introduction

Software Development Life Cycle is systematic process of developing software application. Typically it includes various framework activities from early project initiation activity to post implementation activity and we can see this approach of development is common in various types of software development activity. Operational, Transitional and Revision are the characteristics of the software. These attributes motivates the technical team to develop good quality software that will fulfill requirement of user in any circumstances and even with least maintenance effort. Here type of development approach plays important role for deciding high degree of integrity and robustness. Currently there are two SDLC methodologies which are utilized by most system developers, namely traditional development and agile development. In section 4 we compare and contrast these two methodologies in detail in following sections. Finally the conclusion is presented.

Traditional Software Development

This software development is also called as "Generic Software Models" or "Prescriptive Software Model" or "Plan-driven Software Model" or "Heavy-Weight Software development". These methodologies follow the predefined order and sequential stages of software development where each succeeding step or stage depends on proceeding steps. Traditional software models generally have the following activities. They are:

a) Communication
b) Planning
c) Modeling
d) Construction
e) Deployment

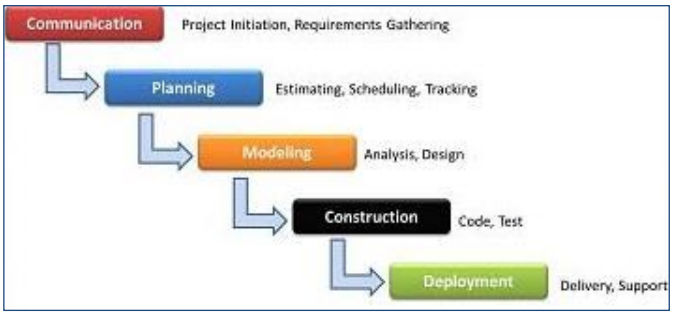

Fig. 1- Traditional Software Development

Step 1: Communication and Requirement Gathering

This activity starts with the intention that all detail requirements for the software are well understood on the basis of this, system and software requirements are understood and analyzed, after that feasibility report and software requirement specification (SRS) is created. This feasibility report and software requirement specification (SRS) will help the system analyst and development team to decide whether they can practically implement the solution or not.

Step 2: Planning

This phase includes following important activities

a) Estimation of the resources (people, tools, time, and technology) required for software development and software project management.
b) Identification, forecasting and management of all the types of software development risk regarding project, process and prod-

uct that are then categorized into proactive risks and reactive risks. Experienced development team usually refers the documentation of previous completed project for identifying known and unknown patterns of risk factors.

c) Software project management and scheduling this activity is considered as most important activity from the perspective of monitoring the software development project. Because the project manager has to manage the project according to:

- Total number of people (effort) required for project development.
- Duration of software project development (Minimum, average and maximum time)
- Partition of Software project development (Functionality) depending on total no. of modules or components to be developed.
- Definition of project milestones and outcomes in each phase of software project development.
- Defining Team structure of development teams in case project is critical to implement or development team includes inexperienced people in the team. And many more activities are carried in this phase. The scope and management of this phase depends on complexity and size of the software project.

Step 3: Modeling

In third phase of software project development the complete software that is to be developed is viewed and studied from operational and technical point of view and only those important aspects and dimension of proposed software product are abstracted, analyzed and presented in graphical notation so that system analyst will validate the requirements and verify it from stakeholder or user. Finally these verified requirements are given to technical team as milestone and on the basis of that pre-defined outcomes are set associated to the requirements. This approach is generally treated as "Modeling". In software engineering modeling is called as bridge between developing software and development software. Modeling basically focuses on two important activities i.e. "Analysis" and "Design".

Software analysis is nothing but understanding the software product for the needed requirements after that feasible requirements are elicited, observed or analyzed and then recorded. On the basis of this analysis it becomes practically clear to system analyst whether they can proceed practically to implement the solution or not.

Software design is representation of the finalized requirement though different graphical notations such as UML Diagrams, E-R Diagram, Component Design, Architecture Design, and few Runable Interfaces. This software design is very important and useful for development team to communicate and understand the integrity, dependency and architecture of the proposed software product.

Step 4: Construction

In traditional software development the fourth phase is called as construction phase, this generally includes "Coding and Testing". First manual or automated code generation techniques are applied to all the forms, components and module. Then simultaneously Software Quality Assurance Activity is applied to ensure that all the errors have been uncovered during code generation. SQA largely involves different types and levels of testing from white-box testing to black box testing and from unit testing to user acceptance testing vice versa.

Step 5: Deployment

Software Deployment phase is the last phase in software development as if all the requirements are stable and acceptable from user point of view. Otherwise an iteration of software developments is carryout out. During each iteration of software development software as a complete entity or partial entity is delivered to the user or customer. Customer has to evaluate it and depending on the changes feedback is given in return. In such way the iteration of software development generally take place that generally carried out to develop the small increments of the proposed software (partially completed software) or complete version of the software. In traditional software development all these steps or phases are implemented or carried out in sequence commencing from communication activity and culminating at Deployment activity. The degree of complexity, size and type of proposed software may decide the iteration required to complete the software project.

There are different types of traditional software development that are follows the sequential and planned driver approach, which are Waterfall Model, V-Model, Iterative development Model, Evolutionary Software models such as Prototyping and Spiral.

Agile Software Development

In modern software engineering agile software development is also called as "Value Driven Software Model". In agile there is no proper "Structure" or "Model" is followed to develop the software but some of the important facts has influenced socially and have found very effective for human factor like development team and customer they are:

- Flexible communication
- Promoting teamwork among software development activity
- Endorsing value based collaboration of human factor and technical factors (customer, individuals in development team, project manager, tools)
- Adaptability of process throughout the life cycle of the project
- Effective Change management of user requirements and project management and development.

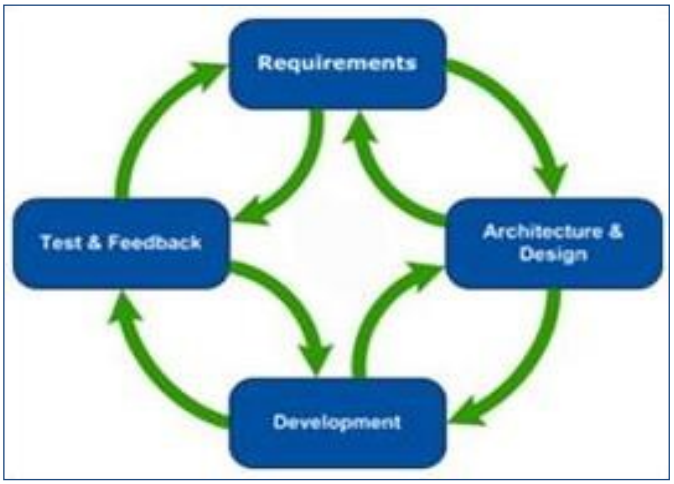

Fig. 2- Agile Software Development

All these factors of development gives flexibility in the implementing the solution so that during software development and management

there is possibility to increase efficiency of work and to maintain the balance between demanding stakeholder and fragile nature of technical team. Generally the activities that are take place in traditional software development are similar but the approach of implementation is different in any other software development same is the fact with agile software development. Here presenting precisely following are the various steps or phases or activities comes in agile software development. They are:

- Requirement Gathering
- Architecture and Design
- Development
- Test and Feedback

In modern software engineering such as agile software development, there is no proper flow of process has been defined or used. But the approach is commonly focused to satisfy the customer and technical team through intensive collaboration and flexible communication as earliest as possible with minimal planning done and to increase the scope of project development. So that more creative solution can be possible to implement practically. In agile software development following are the main activities take place rapidly. They are:

The iteration cycles in agile software development are kept very short that includes two to three weeks.

Step 1: Requirement Gathering

During each iteration cycle more refined requirements are gathered then they are analyzed focusing on core elements or components of proposed software then requirement are finalized.

Step 2: Architecture and Design

Moving ahead software architecture is designed and developed; some graphical design notation is used to understand relationship and inter-dependencies of software component.

Step 3: Development

Coding is done for each of the form, module and component using manual or automated code generation.

Step 4: Test and Feedback

Each of the partially completed entity or complete version is tested to see no bugs reside into it. And that tested component will be delivered to customer for feedback for further adaption in it if any.

In each iteration all these activities take place in such short time frame so frequently that any kind of change may be possible to implement at any time at any phase of the development. The applicability and degree of efficiency of this flexible and rapid approach depends on following manifesto of agile software development. They are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contact negotiation
- Responding to change over following a plan [1]

There are seven software development approaches that come in agile software development. They are XP Programming (XP), Adaptive Software development (ASD), Dynamic System Development Method (DSDM), Scrum, Crystal, Feature Driven Development (FDD), and Lean Software Development [1].

Discussion

Both traditional software engineering and agile software engineering offers numerous advantages to the software development. The selection of these approaches and any other software development process depends on two important factor complexity and functionality of proposed software and other factors includes nature of project to be developed, size of software, type of software, availability of the resources such as people, process, tools and importantly time required, solution accepted in terms of duration of time and so on. Following are the noticeable similarities and differences have been found in both the approaches. They are:

Documentation vs. Working Software

One major difference between traditional software development and agile development is Documentation which is not found in agile software development but working software is used as a reference and source of information in agile software development.

Contract Negotiation vs. Collaboration

Second major difference is that, in traditional software development the focus of development is on contracts, plans, and processes and tools where as agile development emphasizes on extensive customer collaboration suggesting and taking participation in development for visualizing and responding to change.

Clear Requirements vs. Fuzzy Requirements

Third notable difference is that in order to choose traditional software development it is implicitly stated that all the requirements for developing software must be clear and complete but this explicit condition is not seen as important factor in selecting agile software development even fuzzy or unclear ideas and requirements can be granted for initial development.

Plan Driven vs. Value Driven

Fourth difference in traditional software development and agile is that, traditional software development always follow pre defined plan and structure and accordingly all the activities, actions or task has to be carried out. In case any ambiguity or problem takes place then also pre determined solution is followed. Agile software basically follows short or minimal plan and evolutionary approach but the structure of implementation is very different and philosophy is that developing each increment as early as possible in short time frame and responding to the changes and generally focused on satisfying the user utmost, moving ahead with more clear needs in creative manner.

Customer Participation vs. Customer Collaboration

Fifth difference is in traditional software and agile software development is related with review meetings. In traditional software development when one iteration of development over, then technical team has to first conduct the meeting with their project manager. Then project manager reviews the entire work task completed by each one of the member and then depending of the quality of the milestone and completion of task, he approves it and then at the end that particular entity of software (partially or complete) is delivered to customer for evaluation and verification. Extensive communication among customer, technical team and project manager and involvement of customer throughout the development of software in agile software development makes it more effective and creative for management and thus technical team directly holds the meeting

with customer without involvement of project manager. There are many differences that have been found in agile and software development some of them have been precisely stated and others are presented in [Table-1].

Table 1- Difference between Traditional and Agile development

	Traditional Development	Agile Development
Fundamental Hypothesis	Systems are fully specifiable, predictable and are developed through extended and detailed planning	High quality adaptive software is developed by small teams that use the principle of continuous improvement of design and testing based on fast feedback and change.
Management Style	Command and control	Leadership and collaboration
Knowledge management	Explicit	Tacit
Communication	Formal	Informal
Development model	Life cycle model (Waterfall, spiral or modified models)	Evolutionary delivery model
Organizational structure	Mechanic(bureaucratic, high formalization), targeting large organization	Organic(flexible and participative, encourages social cooperation), targeting small and medium organization
Quality control	Difficult planning and strict control. difficult and late testing	Permanent control or requirements, design and solutions. Permanent testing
User requirements	Detailed and defined before coding	Interactive input
Cost of restart	High	Low
Development direction	Fixed	Easily changeable
Testing	After coding is implemented	Every iteration
Client involvement	Low	High
Additional abilities required from developers	Nothing in particular	Interpersonal abilities and basic knowledge of business
Appropriate scale of the project	Large scale	Low and medium scale
Developers	Oriented on plan, with adequate abilities, access to external knowledge	Agile, with advanced knowledge, co-located and cooperative
Clients	With access to knowledge, cooperative, representative and empowered	Dedicated, knowledgeable, cooperative, representative and empowered
Requirements	Very stable, known in advance	Emergent, with rapid change
Architecture	Design for current and predictable requirements	Design for current requirements
Remodeling	Expensive	Not expensive
Size	Large teams and projects	Small teams and projects
Primary objectives	High safety	Quick value

Conclusion

In Traditional and modern software engineering, the software development paradigms are technique of systematic management, development and maintenance of software product In spite of many fragile factors of traditional software development approach, it suggests the good practice of development especially for the new learner i.e. “Think and analyzes the problem before implementation and then prepare the final solution and always carry out the plan and follow the structure”. Traditional software development has been significantly recommended for large mission critical project where planning is critical. In the same manner agile software development also need to improvise in its scope so that team should be close to the process instead of providing alternative way of doing for the solution but that isn’t hampering practically in implementation, when solution is provided in open environment. Agile development proposes the philosophy of “social learning” i.e. “No matter whether you don’t follow the roadmap, no matter what you stick to implement it, but do it in efficient way with the help of each other on trial and error basis in creative manner”.

Conflicts of Interest: None declared.

References

[1] Pressman R.S. (2001) *Software engineering: a practitioner’s approach*, McGraw-Hill, 466-472.

[2] Stoica M., Mircea M. & Ghilic-Micu B. (2013) *Informatica Economica*, 17(4), 64-76.

[3] Leau Y.B., Loo W.K., Tham W.Y. & Tan S.F. (2012) *Software Development Life Cycle AGILE vs Traditional Approaches*, International Conference on Information and Network Technology, 37(1), 162-167.