International Association of Scientific Innovation and Research (IASIR)
(An Association Unifying the Sciences, Engineering, and Applied Research)

ISSN (Print): 2279-0020
ISSN (Online): 2279-0039

International Journal of Engineering, Business and Enterprise Applications (IJEBEA)

www.iasir.net

AN AI AND AGILE BASED SOFTWARE DEVELOPMENT WITH SCRUM METHDOLOGY

Ms. Khushboo Karodiya [1], Mr. Manoj pawaiya[2]
Computer Science Department
Patel Group of Institute Indore (India) [1]
Prestige Institute of Engineering and science, Indore, (India) [2]
Address: Patel Engineering College near ralamandal, Indore (M.P) [1]
Address: "Prestige Vihar", Scheme no 54, Vijay nagar, Indore (M.P) [2]
INDIA

Abstract: *a software development process is a structure imposed on the development of a software product. In software development the agile methods have been used. In traditional method of agile have some control factors these factors are cost, schedule, requirement and quality. All these factors are interconnected to each other and affected by each other. In this paper we propose an approach by which the management of control factors are easily drive and prediction in each steps involve to provide better decision making capability in software development*

Keywords: *software development, control factors, prediction decision making, Scrum, Agile*

I. Introduction

Agile management or agile project management is an iterative method of determining requirements for engineering development projects in a highly flexible and interactive manner, due to this agile development has attracted huge interest from the software industry. A survey in the USA and Europe reveals that 14% of companies are using agile methods, and that 49% of the companies that are aware of agile methods are interested in adopting them in just six years.

Agile project management approaches balance the four variables in software development while keeping in mind the limits associated with new product development. In software development there are four broad control factors. These factors are interconnected when changes in one factor made changes automatically reflected on other factors.

1. Cost: or efforts available money impacts the amount of efforts put into the system.
2. Schedule: a software project is impacts as the time line is changed.
3. Requirements: the scope of the work that needs to be done can be increased to the effect the project.
4. Quality: cut corners by reducing quality.

Because software development is often considered a sequential, linear process, middle and upper management often assumes that all four of these factors could be dictated to the development team under the waterfall approach. However software development cannot be described by a simple linear process because it cannot be predicted accurately in advance. It is therefore unreasonable to assume that management can control all four of these factors. In reality, management can pick values for three of the four factors at most, and the development process dictates the fourth. The highly complex and uncertain nature of software development makes this expectation of full control unrealistic.

II. MANAGEMENT BACKGROUND

Success of project requires recognizing the following principles:
The control estimate and control schedule are used to establish baselines to measure progress and variation. An earned value system should be established to estimate actual physical progress and to establish a basis for predicting future effort and schedule time required for completion. It is important to note that cost and schedule control systems provide more that reactive accounting—they also provide timely information for management intervention. Cost overruns and schedule delays are predicted, not just reported. Timely information allows the opportunity to influence scope, crew size, interferences, late deliveries, and other project characteristics to mitigate the overall impact of negative occurrences.

a. *Cost Management:* Project cost is typically reviewed monthly. The cumulative effect of change is tracked and predicted for all firm-price contracts. Reimbursable cost is predicted by analyzing work progress, amount spent to date, committed cost, and amount predicted to complete. The cost management accounting system summarizes the cost of individual pieces of scope and predicts total

construction cost. Short-duration projects, like shutdowns and turnarounds, may require more frequent analysis and reporting. These projects often predict cost on a weekly, or even daily, basis.

b. *Schedule Management:* Typically, the schedule is reviewed each week. This is best done at a meeting led by the owner's Construction Manager, where all contractors are represented. Each representative reports schedule progress, delays, and potential interferences. The owner's Construction Manager enables the coordination of multiple schedules and analyzes schedule progress. Problem areas are addressed and remedies defined to bring schedule progress back toward the plan committed to the business owner. Again, short-duration projects like shutdowns and turnarounds may require more frequent analysis and reporting. These projects often predict schedule each day, or sometimes each shift.

c. *Change Management:* Change is inevitable and often desirable. The project change management system should identify and document all variation from the contract drawings and specifications and provide a process for technical approval and project authorization. The change order documentation should identify not just the cost impact of the change, but also any schedule, quality, and safety considerations. These are key issues to be addressed in the decision-making process. Proposed change should be dealt with in a timely manner. Once authorized, the change should be incorporated in the project scope of work. Excessive change can add complexity and cost beyond that identified in individual change orders.

III. PRACTICAL DIFFERENCES IN METHODOLOGY

The first software development methodologies were hardly methodologies at all, but a free-for-all as organizations struggled to profit from new computer-related technologies. As the industry learned more about developing software, certain techniques for managing and predicting the cost of software development projects came into use. The methodology that has dominated software development projects for decades is called "waterfall." Winston Royce coined the term in 1970 to describe a serial method for managing software projects through the five stages shown in Figure

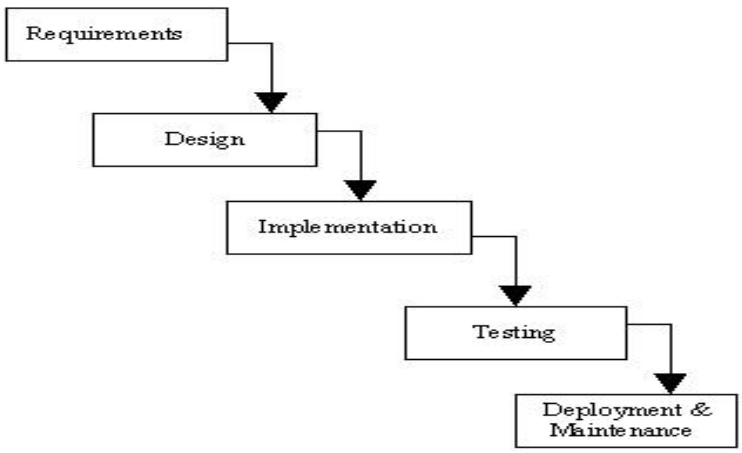


Fig shows waterfall model

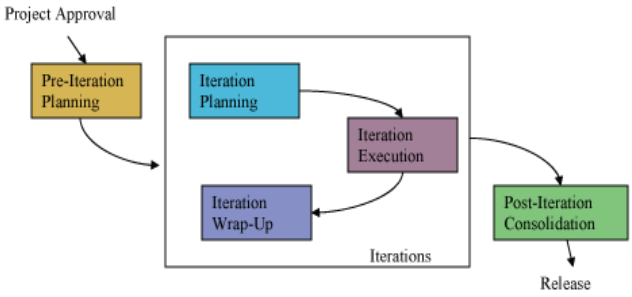
Adoption of waterfall has helped drive down the failure rate of software development projects, but even with rigorous project management and processes, a full 70 percent of software projects using this methodology fail to meet their objectives. To put this in perspective, waterfall software projects have less than half the success rate (66 %).

One of the most important differences between the agile and waterfall approaches is that waterfall features distinct phases with checkpoints and deliverables at each phase, while agile methods have iterations rather than phases. The output of each iteration is working code that can be used to evaluate and respond to changing and evolving user requirements.

Waterfall assumes that it is possible to have perfect understanding of the requirements from the start. But in software development, stakeholders often don't know what they want and can't articulate their requirements. With waterfall, development rarely delivers what the customer wants even if it is what the customer asked for.

Agile methodologies embrace iterations. Small teams work together with stakeholders to define quick prototypes, proof of concepts, or other visual means to describe the problem to be solved. The team defines the requirements for the iteration, develops the code, and defines and runs integrated test scripts, and the users

verify the results. (See Figure) Verification occurs much earlier in the development process than it would with waterfall, allowing stakeholders to fine-tune requirements while they're still relatively easy to change.



1. DEVELOPMENT METHODOLOGY

The most widely used methodologies based on the agile philosophy are XP and Scrum. These differ in particulars but share the iterative approach described above.

XP

XP stands for extreme programming. It concentrates on the development rather than managerial aspects of software projects. XP was designed so that organizations would be free to adopt all or part of the methodology.

XP development

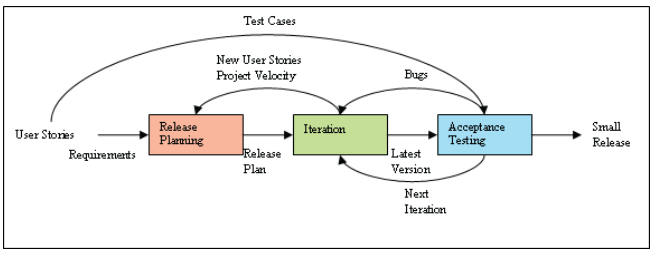
XP projects start with a release planning phase, followed by several iterations, each of which concludes with user acceptance testing. When the product has enough features to satisfy users, the team terminates iteration and releases the software.

Users write "user stories" to describe the need the software should fulfill. User stories help the team to estimate the time and resources necessary to build the release and to define user acceptance tests. A user or a representative is part of the XP team, so he or she can add detail to requirements as the software is being built. This allows requirements to evolve as both users and developers define what the product will look like.

To create a release plan, the team breaks up the development tasks into iterations. The release plan defines each iteration plan, which drives the development for that iteration. At the end of iteration, users perform acceptance tests against the user stories. If they find bugs, fixing the bugs becomes a step in the next iteration.

Iterative user acceptance testing, in theory, can result in release of the software. If users decide that enough user stories have been delivered, the team can choose to terminate the project before all of the originally planned user stories have been implemented.

Figure shows a simplified version of XP. Full XP includes many steps in release planning, iteration, and acceptance testing that is not shown here. Visit www.extremeprogramming.org for a full description of XP.



XP rules and concepts

Here are the most important concepts:

Integrate often. Development teams must integrate changes into the development baseline at least once a day. This concept is also called **continuous integration**.

Project velocity. Velocity is a measure of how much work is getting done on the project. This important metric drives release planning and schedule updates.

Pair programming. All code for a production release is created by two people working together at a single computer. XP proposes that two coders working together will satisfy user stories at the same rate as two coders working alone, but with much higher quality.

User story. A user story describes problems to be solved by the system being built. These stories must be written by the user and should be about three sentences long. User stories do not describe a solution, use technical language, or contain traditional requirements-speak, such as "shall" statements. Instead, a sample user story might go like this: Search for customers. The user tells the application to search for customers. The application asks the user to specify which customers. After the user specifies the search criteria, the application returns a list of customers meeting those criteria.

Because user stories are short and somewhat vague, XP will only work if the customer representative is on hand to review and approve user story implementations. This is one of the main objections to the XP methodology, but also one of its greatest strengths.

Scrum

In rugby, 'scrum' (related to "scrimmage") is the term for a huddled mass of players engaged with each other to get a job done. In software development, the job is to put out a release. Scrum for software development came out of the rapid prototyping community because prototypers wanted a methodology that would support an environment in which the requirements were not only incomplete at the start, but also could change rapidly during development.5 Unlike XP, Scrum methodology includes both managerial and development processes

Scrum management

At the center of each Scrum project is a backlog of work to be done. This backlog is populated during the planning phase of a release and defines the scope of the release (see Figure). After the team completes the project scope and high-level designs, it divides the development process into a series of short iterations called sprints. Each sprint aims to implement a fixed number of backlog items (see Figure). Before each sprint, the team members identify the backlog items for the sprint. At the end of a sprint, the team reviews the sprint to articulate lessons learned and check progress.

During a sprint, the team has a daily meeting called a scrum. Each team member describes the work to be done that day, progress from the day before, and any blocks that must be cleared. To keep the meetings short, the scrum is supposed to be conducted with everyone in the same room—standing up for the whole meeting.

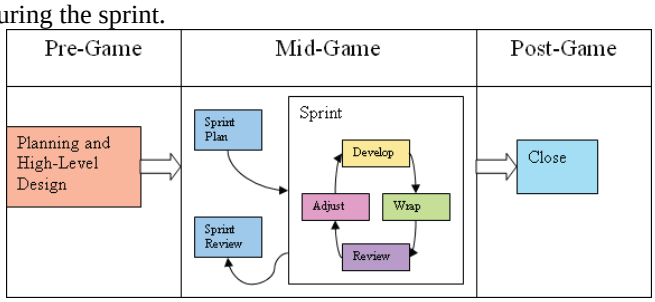
When enough of the backlog has been implemented so that the end users believe the release is worth putting into production, management closes development. The team then performs integration testing, training, and documentation as necessary for product release.

Scrum development

The Scrum development process concentrates on managing sprints. Before each sprint begins, the team plans the sprint, identifying the backlog items and assigning teams to these items. Teams develop, wrap, review, and adjust each of the backlog items (See Figure 6).

During development, the team determines the changes necessary to implement a backlog item. The team then writes the code, tests it, and documents the changes. During wrap, the team creates the executable necessary to demonstrate the changes. In review, the team demonstrates the new features, adds new backlog items, and assesses risk. Finally, the team consolidates data from the review to update the changes as necessary.

Following each sprint, the entire team—including management, users, and other interested parties—demonstrates progress from the sprint and reviews the backlog progress. The team then reviews the remaining backlog and adds, removes, or reprioritizes items as necessary to account for new information and understanding gathered during the sprint.



Scrum concepts

Here are a few of the most important concepts:

Burndown chart. This chart, updated every day, shows the work remaining within the sprint. The burndown chart is used both to track sprint progress and to decide when items must be removed from the sprint backlog and deferred to the next sprint.

Product backlog. Product backlog is the complete list of requirements—including bugs, enhancement requests, and usability and performance improvements—that are not currently in the product release.

ScrumMaster. The ScrumMaster is the person responsible for managing the Scrum project. Sometimes it refers to a person who has become certified as a ScrumMaster by taking ScrumMaster training.

Sprint backlog. Sprint backlog is the list of backlog items assigned to a sprint, but not yet completed. In common practice, no sprint backlog item should take more than two days to complete. The sprint backlog helps the team predict the level of effort required to complete a sprint.

IV. THE NEW APPROCH TO SOFTWARE DEVELOPMENT

A. Introduction

To improve the performance of the agile software development using scrum technology, we proposed a more effective and advanced approach over the existing approach. In proposed approach we introduce a tool. Using tool we describe the methodology of software project management using scrum methodology.

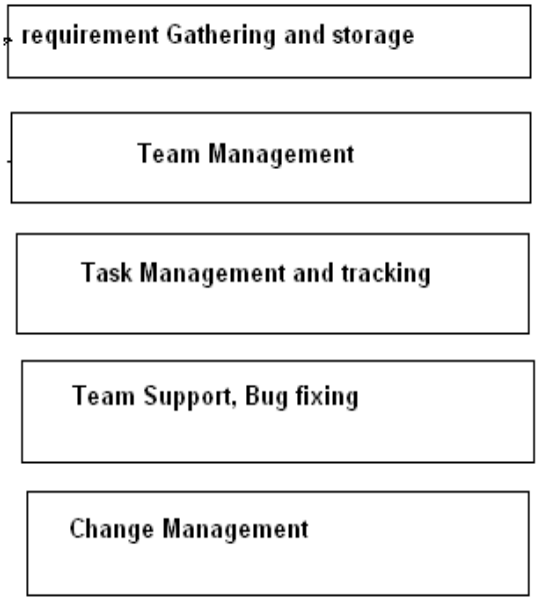
B. working

Our tool works in some steps these steps are:

1. Requirement gathering: Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. This step of application takes input about requirement analysis. Requirement analysis organization has its own typical format user required to follow these formats.
2. Planning: this step of software development, project analysis is driven (or requirement refinement is done). Here goal and aim of software decided. For more effective and efficient planning, our tool manages a database of all members of organization. Tool analyzes members profile and suggests the team members. More over these suggestions of cost, time and quality of project predicted. Here a provision to adjust these control factor by add or remove any resources in team. After adjustment of control factor proceed to next step.
3. Implementation: A product software implementation method is a systematically structured approach to effectively integrate software based service or component into the workflow of an organizational structure or an individual end-user. After deciding the team members, different works assigned to the members in daily basis. Task assigned, task remaining and task progress is defined by graphical manner that is called Burndown chart. For improvement purpose we define burndown chart in two ways. First for complete project and second for individual team members. These charts are updated every day and used to track the project work and team member work.
4. Back Log: backlog is the complete list of requirements—including bugs, enhancement requests, and usability and performance improvements. here used back log some thing different form the traditional back log management. It contains bugs, enhancement requests and usability and performance improvements for both (overall project and individual team member).
5. Team Support: Moreover that, there is a new concept based on burndown chart analyzer. If any team member's task not completed in time and/or made delay to submit report. Then tool understand that factor as problem in particular task and make automatic request for help.
6. **Redesign and Change Management:** there are different kind of projects in software industry sometimes they are small and some times they are log enough. Some other factors also make project difficult and complex. Like any requirement is changed, and/or any team member leave the project. Thus problems arise, for that purpose to handle theses change management calls. Like if any member leaves the project then system suggest a compatible member who can replace the need of that member. And if any change in requirement then system predicts what effect are going on in project deciding factors.

C. ARCHITECTURE:

The given fig shows the different management schemes involved in project management.



V. CONCULSION AND FUTURE WORK

Our tool is based on the agile software development using scrum methodology. Tool manages the development process involved in software project industry. By Changes over some traditional method of agile software development can improve the previous solution for project resource planning and management. These changes produce a revolutionary effect on planning and management.

In future we improve more features of our tool to make things more transparent and flexible.

VI. LIMITATIONS

Each and every system has its own limitations our tools also have some limitations.

1. Requirement analysis step required large manual efforts.
2. Prediction over different control factors is normal, that is not much accurate.
3. Team support is just a request for help provided help is depended over the scrum master.

VII. REFRENCES

[1.] Winston Royce, Managing the Development of Large Software Systems, Proc. Westcon, IEEE CS Press, 1970, pp. 328-339.

[2.] Standish Group International, Inc., "Chaos Chronicles", 1994, http://www1.standishgroup.com//sample_research/ chaos_1994_1.php

[3.] Kent Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley, 2000, pp. 18-19.

[4.] Ken Schwaber, Mike Beedle, Agile Software Development with Scrum, Prentice Hall, 2001, pp. 89-94

[5.] Standish Group International, Inc., "Chaos Chronicles",1994, http://www1.standishgroup.com//sample_research/chaos_1994_1.php

[6.] [Weinberg97] G. Weinberg, Quality Software Management – Vol. 4., Anticipating Change, Dorset House, New York, 1997.