AGILE PROGRAMMING-A NEW APPROACH

Prema.S.Thomas
Assistant Professor
Department Of M.C.A,Rajagiri College Of Social Sciences
Kalamasserry, Kochi
E-Mail: prema.s.thomas@gmail.com

ABSTRACT

The paper “Agile Programming- A New Approach “intends to present the need for a new method in agile programming. A brief introduction about the traditional methods used in agile programming is discussed .The reasons for moving to agile programming is explained. A comparative study of the various technologies like Scrum, Extreme Programming and Lean Development in agile programming is presented. The study reveals quite a few drawbacks such as inability to cater to large sized projects, does not support multi site projects and so on. The research is on “what methodology could be designed” such that it takes advantage of all the existing methods while the deficiencies of the current agile techniques are overcome.

The key points addressed in this paper on Software Project Management are:

- Issues With The Traditional SDLC methods
- Reasons For Moving To Agile Programming
- Strategies In Agile Programming
- Various Techniques In Agile Programming
- Comparison Between The Different Methods
- Drawbacks Of The Existing Methods
- Need For A New Approach
- Proposed Features Of The New Approach To Agile Programming
The paper focuses on

- An overview of the existing methodologies in agile programming
- It also analyzes the differences among the various methods.
- Proposing a new methodology.

**Keywords:** Agile programming, Scrum, Extreme Programming, Lean Development, Sprint, Burn Down chart.

**WHY AGILE PROGRAMMING?**

Agile Programming is an outcome of the difficulties encountered in the traditional methods like waterfall model, spiral model etc. where you need to wait till the end of the complete cycle to know the flaws. Agile Programming, a Software Project Management technique aims at developing the product with a clearly defined series of steps that are recursively iterated to get the ultimate product. The methodology involves defining clear cut tasks that are to be executed in a short time period which is reviewed at the end of the term. In agile programming the tasks are precise and the time period short, that the results become visible at the end of this time period. Hence changes can be done easily since the flaws are identified in the early stages itself. The effective communication among the peer members of the team helps in reducing the issues that might exist in the development process.

**HISTORY**

Agile Programming had its start way back in 1970, when Dr. Winston Royce analyzed his thoughts on the traditional waterfall model. In his paper “Managing the Development of Large Software Systems” he analyzed the steps in the waterfall model and found that the entire development cycle lacked effective communication. Though the precise steps in the model helped identifying the process, lack of communication among the team members created flaws in the ultimate product. In fact it was an eye opener to him and so he decided to have a more effective development method - Agile Programming.

In the traditional models, the developer assumes the requirements based on the requirements specifications and proceeds developing the product. But when the final product is released the client realizes that he had a different requirement. This issue could
have been sorted out through a simple and effective communication before the release of the product. Hence it revealed the need for frequent and timely meetings that could make the development process more precise. When the requirement is documented, it becomes a good proof but misinterpreted most of the time. This gives enough reasons why agile programming has been introduced.

**STRATEGIES**

A few strategies used in agile programming which could be the key rules in agile programming are:

1. Simple Design – Design in such a way that the steps are self-explanatory
2. Progress as you proceed – Steps that are upgraded once you complete a particular task
3. Iterative Steps – Steps are automatically incremented after each task
4. Optimal Principle – Use methods that leads to a solution to the problem
5. Self reliant steps – Design in such a way that the steps are independent
6. Tools with a purpose – Use tools which serve a definite task

**AGILE DEVELOPMENT TECHNIQUES**

**Scrum**

The most popular agile programming technique is the scrum. The scrum method of agile programming helps in attaining a smooth work flow on a day to day basis guided by sprints. It reduces the unnecessary documents which are being generated and which is often misinterpreted by the developers. A very efficient method of developing a product which follows a divide and conquer strategy for achieving the requirements specified by the customer. Sprint with its time box of around two weeks to one month helps in defining clear cut deadlines and specifications to be met on that day. Though the no. of days cannot be increased, the sprint backlog can be modified if needed.
Figure 1.1

Important
- Processes and tools
- Detailed documentation
- Contract negotiations
- Following a plan customer

More important
- Individuals and interaction
- Functioning software
- Collaboration with the
- Adapting to changes

(Source: Manifesto for Agile Software Development http://agilemanifesto.org/)

Figure 1.2
EXTREME PROGRAMMING

The next most popular agile development technique is Extreme Programming. It is a way of programming that focuses on how to work in a project.

Following are the 12 practices which is followed in the case of Extreme Programming:

1. Planning – The requirements are listed by the customer, the team checks for the feasibility based on several factors such as cost, time, resource etc and then both the parties negotiate for the most feasible plan that meets the business needs. This game is played at the beginning of each iteration in the development process.

2. Small Releases – The team then decides on the divisibility of the tasks and then makes small units of work that can be completed within a week or two.

3. Metaphor - Extreme Programming metaphor is a high level view of the system to be developed. Any detailed description is maintained as a separate document and not given in the high level vision of the system

4. Simple Design - Simplicity is one important feature of Extreme Programming. The design is kept as simple as possible in order to avoid the complexity of any rework in the future.

5. Test First – To make the task simpler, before they write a single line of code, they should have a clear idea about the test cases to be applied to check for the correctness of the code. Hence it is often advisable to code a little and then test a little till the code is bug free.

6. Refactoring – Refactoring is a clean up of the code in order to make the non functional software functional without changing the behavior of the code.

7. Pair Programming – The most healthy and visible feature of XP Programming is the pair programming. It help the pair (a team of developers) double check on the code as well as the testing phase of the code.

8. Collective Ownership – In XP programming, there is no single ownership for the code. The code is owned by the complete team.
9. Continuous Integration – The integration of the system is a continuous process and not just done once the complete system is build. The pair will have to integrate the code with the base line system once they have completed the code and then test it for errors.

10. 40-hour Week – The time frame for XP Programming is 40 hour week in order to motivate the team to see the results at the end of 40 hour week.

11. On-site Customer – The person who will have hands on experience with the system is the on-site customer. The on-site customer is responsible for trying out the needs of the customer and suggests any modifications needed.

12. Coding Standards –X Programming follows some standards such as pair programming, collective ownership etc which maintains the order of the system. When a collaborative work is done, some standard has to be maintained in order to avoid flaws in the system.

**COMPARISON OF SCRUM & EXTREME PROGRAMMING**

<table>
<thead>
<tr>
<th>Scrum</th>
<th>Extreme Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scrum is based on sprints that are normally two weeks to one month long</td>
<td>Extreme Programming is generally one to two weeks long</td>
</tr>
<tr>
<td>Scrum does not allow changes in its sprint</td>
<td>Extreme Programming is more flexible</td>
</tr>
<tr>
<td>Scrum prioritizes the product backlog but the team decides in which order it is to be executed</td>
<td>Extreme programming works in a strict priority basis set by the product owner</td>
</tr>
<tr>
<td>Scrum does not insist on any engineering practices</td>
<td>Extreme programming works on the engineering practices such as refactoring, automated testing etc</td>
</tr>
<tr>
<td>Scrum is a project management process</td>
<td>Extreme programming is a project management practice</td>
</tr>
<tr>
<td>The main aim of scrum is to get an estimate of how long will the development take</td>
<td>The main of extreme programming is to help developers to get the things done quickly</td>
</tr>
</tbody>
</table>

**LEAN DEVELOPMENT**

Lean development is an agile development technique that is developed by first understanding “value “ and then what are the resources needed to create the value. Lean development helps to develop a high quality, lowest cost product within the given time.
frame. The main focus of lean development is the development of software that is open to changes.

The principles of Lean Development are:

- Eliminate waste by selecting only the truly valuable features for the system;
- Satisfy stakeholders;
- Empowerment;
- Deploying Comprehensive testing;
- Deliver as fast as possible;
- Refactoring;
- Learn by Experimentation;
- Measure Business impact;
- Optimize across organization;

## COMPARISON OF SCRUM & LEAN DEVELOPMENT

<table>
<thead>
<tr>
<th>Scrum</th>
<th>Lean Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scrum starts at the team</td>
<td>Lean development starts at the process</td>
</tr>
<tr>
<td>Scrum first creates a self organizing team with a clear set of objectives</td>
<td>Lean concentrates on value creation and elimination of waste</td>
</tr>
<tr>
<td>Scrum restricts itself to a basic framework of project management practices</td>
<td>Lean gives specific hints as to how to create a project</td>
</tr>
<tr>
<td>Scrum has a set of well defined roles, meetings etc..</td>
<td>Lean consists of more general advices with a value system</td>
</tr>
<tr>
<td>Scrum is more specific on time boxes</td>
<td>Lean is less specific on time boxes</td>
</tr>
<tr>
<td>Less emphasis on quality</td>
<td>More emphasis on quality</td>
</tr>
</tbody>
</table>

### Burn Down Chart of Agile Programming

The burn down chart of agile programming helps in realizing how much work remains to implement and the time left for delivering the product. It is a tool for comparing the percentage of work left in comparison with the no of days left for the release of the product.
PROS & CONS OF AGILE PROGRAMMING

Agile programming is best suited for small sized projects. It cannot be applied for large sized projects that run for years together and which involves a bigger team of more that 100 members in the team.

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>The team is fully focused</td>
<td>Suitable for small sized projects</td>
</tr>
<tr>
<td>The delivery of the product is fast</td>
<td>Easily leads to burnouts</td>
</tr>
<tr>
<td>There is a good feedback system</td>
<td>May lead to social issues if decisions are not mature</td>
</tr>
<tr>
<td>It discovers optimal solutions faster based on the feedback</td>
<td>It might lead to overheads such as testing after each iteration</td>
</tr>
<tr>
<td>There is less chance of deviating from the actual product</td>
<td>Might affect the product if the sprints are mutually dependant</td>
</tr>
<tr>
<td>It gives way to good learning</td>
<td>Delay due to waiting for resources outside the team</td>
</tr>
<tr>
<td>Identifies the flaws easily and can be corrected easily</td>
<td>May lead to complexities as the projects become larger</td>
</tr>
</tbody>
</table>

A NEW APPROACH TO AGILE PROGRAMMING

The major drawback of agile programming is that it cannot be applied to large projects. Another key issue is that it does not support multi site projects. Hence there is a need for a new methodology apart from the scrum, XP Programming and Lean
Development. A technique that takes advantage of the three techniques and gives a better method that best suits larger applications. The basic idea of this method is to review the existing agile programming techniques and extract the limitations of the same. Further based on various case studies generate a new technique that suits larger applications. The study basically aims at focusing on multi site project, projects of large size and so on. The key factor in this new technology is the deployment of the method to practical scenarios. Some of the features of the proposed methodology are

- Support projects of large size (more than 100 people)
- Support multi site projects
- To be able to do overall optimization on schedule using fast tracking

**CONCLUSION**

The paper titled “Agile Programming – A New Approach” presents the methodology used in agile programming. The basic intention of introducing this concept is to give a comparative study of the various methodologies that exist in agile programming. The existing methodology can be applied only to small projects and so aims to design a new methodology for agile programming that can be applied to larger projects. The new methodology should be applicable in all the larger projects while retaining the benefits of the existing methodologies. It gives way to research on a few questions such as, why is it not applicable to larger projects?, what are the drawbacks of the existing methodologies?, and so on. Hope with the comparisons and with a brief overview of the existing agile development methods, a newer and better method would evolve that suit any type of projects.

**REFERENCES**

1. Ian Sommerville, Software Engineering, Pearson Education, Eight Edition
4. David Berger, Bill Cloke, Lean products start with Lean design,


