# Achieving Agility in Software Maintenance

Ms. Vishakha
M.Tech.(IT), 4th semester
University School of Information and Communication Technology
Guru Gobind Singh Indraprastha University
Sector-16 C, Dwarka, New Delhi-110075

***Abstract:*** **The purpose of this paper is to present the understanding of the maintenance of the project. Many studies shows that building a software is not as much time and money consuming as maintaining it. Approximately ninty percent of the software effort is maintenance. A light methodology is taken to be considering so that we can make our effort use, less in maintenance. One methodology is agile. In agile there are many framework like,SCURM, XP, RATIONAL UNIFIED PROCESS (RUP), KANBAN.**

## I. INTRODUCTION

Agility, for a software development organization, is the power of software to choose and react expeditiously and fittingly to various changes in its surround and to the demands imposed by this surround. An agile process is one that readily embraces and supports this degree of flexibility. So, it is not simply about the size of the process or the speed of delivery; it is mainly about flexibility. This term was agreed during a big gathering when seventeen of the developers of the "lightweight" approaches to software development came together in a workshop in early 2001. Previously, circumscribe of assorted groups have independently developed methods and practices to act to the changes they were experiencing in software processing and development.

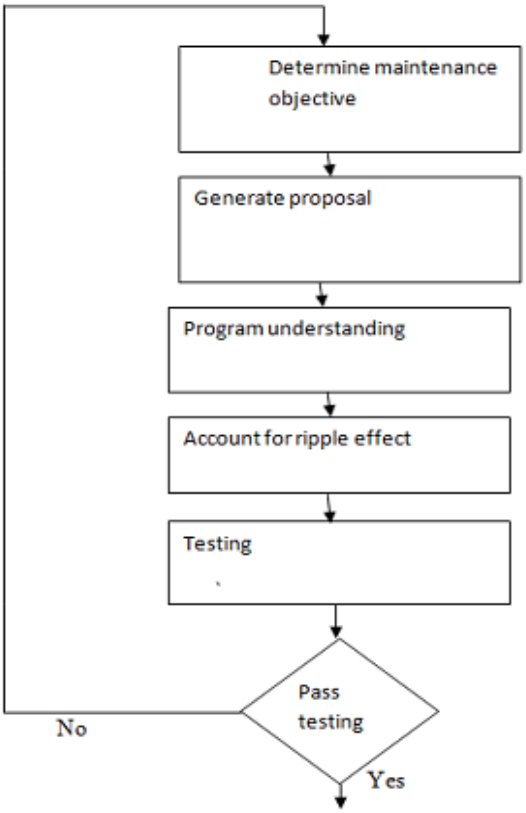
Maintenance:

ISO/EIC 14764(1999) defines software maintenance as "A software product that undergoes modification to code and associated documentation due to a problem or need for improvement " and also IEEE standard 1219(1998) defines software maintenance as " the modification of software product after delivery ,to correct faults ,to improve performance or other attributes , or to adapt the product to a modified environment"[1]

Factors effecting maintenance:
The amount of analysis/design in a maintenance effort is measured by the amount of Function Points, a metric which is proposed by Albrecht [3] as a measure for the size of a system's functionality. Coding/testing is measured by counting the amount of source lines of code (SLOC) added. The goal of the research was to identify environmental factors influencing the amount of labor hours required for a maintenance job .The most significant environmental factor negatively influencing the amount of labor hours required turned out to be the use of a structured design and analysis methodology. Using a structured design may lead to benefits in the long term. Other factors are high ability staff and absence of application experience.
Three factors that affect amount and type of maintenance are [2]
Functionality (strategic/important or not),
Development practice (automatically generated code or not), and
Software complexity
Age
Complexity
Old Module does not contain more error than the newer one, but require more effort to repair them. Shen et al. add additional support for a relation between age and maintenance by noting re pairing errors gets more expensive the later they are found [4].
Yau et al. [5] explicitly mention „Understand program" as a big part of the maintenance process, claiming the understanding comprises three parts: complexity, documentation, and self-descriptiveness. The latter refers to how well the code is understandable on its own, i.e. without of
supporting documents.
Problems during maintenance [6]:
1. Often the program is written by another person or group of persons working in isolation from each other.
2. Often the change is done by other person who did not have understood it clearly, resulting in deterioration of the program's original organization.
3. There is high staff turnover within the information technology industry. Due to this many systems are maintained by persons who are not the original authors. These persons may not have proper knowledge about the system. This may mean that these person often introduces changes without aware of the impact of the changes- the ripple effect. This problem may be worsened by the absence of the documentation. Even where it exists, it may be out of date or inadequate.
4. Some problems often get noticed when the system is in use.
5. Systems are not designed of change.
Thus cost of maintenance if often become close to the 40-70% of the total cost of the software system.

Potential solution to the maintenance problem [7]
Budget and effort reallocation
In now- a- days suggested that more time and resources should be invested in development and specification and

design of more maintainable system but not in unmanageable system. The use of ISOP 9000, CMM and maintenance standard are aimed at this issue.
If maintaining an existing system is more costly than the original system than its better to replace it completely.
Complete replacement of the system is not good option. Existing system in itself can be an asset to an organization in terms of investment in technical knowledge and working culture engendered.
Maintenance process involves the steps:[8]




In determining objectives we consider what to do:
1. Correct program errors.
2. Add new capabilities.
3. Optimization.
In program understanding we consider:
1. How much it is Complex.
2. Documentation is it properly understandable or not.
3. Is itself descriptive or not.
In generate proposal:
1. We consider extensibility, whether the program support or not.
Account of ripple effect:
1. Program modification not only changes that part, but also affects the other part too. Ripple effect finds out the areas.
Finally we do testing of that part by doing Regression testing.
Many studies shows that building a software is not as much time and money consuming as maintaining it. Approximately 90% of the software effort is maintenance. A light methodology is taken to be considering so that we can make our effort use in maintenance. One methodology is agile. In agile there are many framework like Scrum, XP, RATIONAL UNIFIED PROCESS (RUP), KANBAN.

Maintenance type [9]:
There are 3 type of maintenance:
Corrective: elimination of error and poor performance
Adaptive: changes to comply with business law
Perfective: new requirement
Adaptive maintenance includes all efforts aimed at changing the software to respond to a changing or changed environment. Corrective maintenance comprises tasks to correct discovered bugs, while preventive maintenance is composed of modifications aiming to prevent any faults in the future, i.e. find and fix bugs that have not yet been discovered in the practical environment. Changing the software in order to optimize its performance or maintainability is referred to as perfective maintenance.

It is hard to describe size of software maintenance cycle, buts lets imagine a typical maintenance work in these lines OR use of agile practices: [16]

- Request from one or more order called demands
- These demand piles up and be put in a batch
- Analyst evaluate the demands, classify it and express it in terms of impact on the application and the daily work
- Team is small mainly two or three people
- Development is done internally , there is no contract with others
- Implementation is conducting with little formality

In this type of maintenance the main process drivers is the client or business team priority. There is no consideration like build from scratch. Over the time there is new need, and the challenge for the planner is that put together that is cost effective for implementation and at the same time meets both content and time priorities
The first to behave smartly in maintenance is form a backlog of the requirements, intentional backlog that enables you to work the batch planning elements. The smaller the work for the requirements is more the work for maintenance higher the work for process set up and works and rework. In worst case scenario where there is no package or batch. We work on individual basis means each demand is implemented on individual basis, the planner is not able to combine request and boost resources in the programming. . If the planning manager able to assemble a package of the request, assigning priorities to them and put them in a queue for implementation than he would be able to retain a factor called cadence. For a repetitive task, if the work demand is messy, the team reaction will vary, and performance is unpredictable. For making work cadence, first set a time period (fix a time period). Therefore cadence and work walk hand in hand.
Thus in short we can say that making a backlog and then prioritizing the requirements and then fixing the period that we will be able to implement these requirement in this much of time period called a sprint period will make the maintenance work to work well and easier and smooth.
In the system that have been live for a long time also have in maintenance from a long time. The team already knows how to evaluate the impact of change. And best to release the

implementation of new release depends on factors like team efficiency, user capacity to absorb new release business demands and so forth. Team determines this time empirically and adjusts resources and effort to meet demand in batch that fits that timing. This learning process comes with commitment and agreement made by planning manager with the user area and organization administration.
Daily stand up meeting are done. On occasion, a representative will be sent to the development team's stand-up meeting to compare notes and find out what is going on in the other project room. Developers have good idea of what is going on so usually they are not convinced with this daily meeting. Refactoring is applied to the codes generally there is goal of fixing the bug without having much change to the code. Generally maintenance team identify that what section of code is poorly designed. If certain section of code is find to have lots of defects than it should be prioritized first and should be fixed first.
All these steps make the maintenance effort less.

What is agile methodology

Agile methodology is an alternative to the traditional methodology .These are also alternative to waterfall technique use for software development
English meaning of agile is:

- (Mentally quick ) able to think rapidly an clearly
- Physically quick able to move your body quickly and fluently

According to Barry Boehm" an outgrowth of rapid prototyping and rapid development experience as well as the resurgence of a philosophy that programming is a craft rather than a industrial process."
Alistair Cockburn is one of the initiators of the agile movement in software development, he defines agile as "*agile implies being effective and manoeuvrable. An agile process is both light and sufficient. The lightness is a mean of staying manoeuvrable. The sufficiency is a matter of staying in the game*" [10]
Barry Boehm described agile methods as "*an outgrowth of rapid prototyping and rapid development experience as well as the resurgence of a philosophy that programming is a craft rather than an industrial process*" [11]

Difference between traditional and the agile approach: [12]

| | |
|---|---|
| 1. Change in the later stages in requirements cannot be possible | Changes can be done , respond to customer request and changes easier |
| 2. No communication with the team | High level of communication and interaction, meetings. |
| 3. Document and review meeting are needed to solve an issue | 5minutes discussion can solve the problems. Like done in sprint meeting |
| 4. Normal release takes approximately 18 months to 20 months | After 10 months the first release is ready |
| 5. Too slow to provides fixes to user | Respond quickly to customer feedback |
| 6. More time is spent on design so that the product will be more maintainable. The "what ifs" arise earlier | No time for "what ifs" |
| 7. It is work centric | It is people centric |
| 8. Documentation is substantial | Documentation is minimal |
| 9. Project life cycle is guided by task | Project life cycle is guided by product features |

Characteristics of agile: [14]
• One of the most characteristics of agile is that it is incremental in nature. Working software is developed and available very quickly. In XP process it is 2 to 3 weeks and in scrum in a moth but in traditional model it is available at the last of whole model.
• Agile methods are people oriented rather than process oriented and code oriented rather than documentation oriented. This makes best way to communicate face to face rather than though the documentation so makes the understanding of the team with each other more good.
• Agile is good for small project rather than the big project. Its design uncovers what you are developing right now. As the change is inevitable so planning for future work is waste.

How it can be helpful in maintenance: [15]
Defects: A development method called TDD (Test-Driven Development) is a method which says that do not implement any functionality until a test for that functionality exists. This development method is an integral part of the XP process. This improves code quality and thus helps in maintainability thus decrease code complexity and also decreases the defects too. Fewer bugs lead to less maintenance effort. This translates the maintenance work to corrective category.
Understanding: In Scrum, daily meetings are done. It is good for the maintenance team if they are part of the Scrum team otherwise it will be very difficult to points out what is exactly need to do in maintenance. As there is less focus on the documentation. Lack of documentation becomes more difficult to the team member who is not intra part of team.
Changing design: agile quickly react to the change in environment. Simple design and constant refactoring helps in it, regularly change to the environment translates the maintenance work to the adaptive category.

Age, Experience: agile development methods have no influence on age of the system. Also does not influence how much experienced people working.

**XP**:

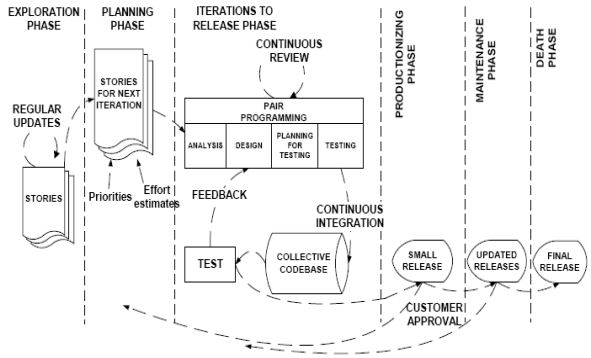



Figure Extreme Programming (taken from (13))

XP breaks the project into the simple, self contained, one to three week mini projects. XP emphasizes planning, analysis and design throughout the life cycle. It includes a practice called "continuous integration "where systems are integrated and build every time a task is completed. XP view peoples, rather than formal documentation, as a project most potent element.

XP uses refactoring to simply software source code by approximately by 40 % by removing dead code unused code from the code

Whole team: team may consist of developer, tester, quality assurance and analyst and customer representative (who provide feedback). [18]

- Planning: for calculation of effort and cost

Release planning: The customer presents features that are expected by him in the software. Developer reviews these features and find out difficulty in achieving them. Based on these constrains initial release is decided.

Iteration planning: The customer presents the features that needed to develop till the next iteration. Based on feature team estimate the time and cost that may be involved in it.

Also each iteration helps in learning about the product.

- Small releases: the customer give some features ,it is like a story , each story represents small iteration that helps in putting new features, that takes only few week to developed. The team releases the running and tested software after each iteration
- Simple design: it shows functionality of the system, XP emphasis on refactoring technique. XP process requires all phases of software life cycle to be carried by a pair of programmer, sharing one pc. This helps in spending more time in finding solution to the problems .less time on routine debugging.
- Pair programming: code is written by two programmers on single machine. Thus a program is reviewed by one other programmer at least. This helps in better design, implementation and testing. Switching of the work in programmer helps in raising their skill and understanding of the system.
- Collective code ownership: Here codes owned by entire team. In XP any developer can work on code base any time thus code can be seen by many people. This increase quality and decrease defects
- Sustainable pace: the team works in a constant pace, the pace that can be sustained indefinitely. We work in a way that productivity lasts.
- Customer tests: acceptance testing is done b the customers.

The advantage of this XP method is that it speeds up the development process. It gives the programmer Wright to fix the problem as it comes. Coding standard is fixed globally, so development team must sure that it uses that standard in coding and design

- XP have some positive points :
- Reduces code complexity by stripping unused code.
- Implements pattern that make it easier to maintain test and understand the code
- Reduces code size by 40%
- Enables most of defects reduction.
- Eliminates code complexity

Promotes improvement of the test coverage quality

What is scrum[20]?

Agile mythology is really quite simple. Let's concentrate on scrum first which is most popular

Backlog:

Process starts is product owner creating prioritizing backlog. Creating backlog is most understandable part of this one. Too many company trying to put lots of effort to build lots of feature in their product but the fact is that too many of them never will get used. Never work correctly or just they get waste, don't use ever. Backlog is actually a type of story that tells who will b conducting the operation, what they will be doing and why. This creates a help to the programming team making a backlog means u are putting yours what you want actually. Than product owner prioritize the list .the most priority requirements are put at the top of stack a least one put are at the bottom. We start working from top of the stack. This list can change with each sprint or each iteration.

Sprint:

This backlog formed is actually helps in sprint planning. Actually sprint is basic unit of time in this agile software development in most cases it varies from 1 month to one week. Project can have one too many sprint .we do sprint planning meeting. The purpose of the meeting is that they decide what amount of work to be done in the sprint period. This is not a guessing game .velocity a term defined is actually work done per sprint. A good team will never commit more than its velocity. Good team will also recognize the importance of the priority backlog. Good team will

always work in higher priority backlog to towards lower one. Once sprint planning is done team actually start working.

**Scrum meeting:**

After the sprint planning scrum meeting is done. It is done on daily basis and last for 15 to 20 minutes. In this the team members tells each other what they have done and what will have to complete before next sprint period. Actually, it is about knowing of progress. What are the risks? What are the impediments? This helps the team to share information so that they can work according to their sprint commitment. This short time meeting exposes risk .profanity of failure. This helps the team to work more effectively

Scrum master:

Person plays special role in software management. He insures the success of the project. He actually removes the impediment coming. This lets the team to work smoothly. This helps the team anyway. This helps in decision making.

During the last two days two meeting are to hold more one is sprint review, another sprint retrogressive.

Client see here first working product, they give there feedback so that further process can be improved by including the feedback result. Thus aims to improve the quality of the product

Sprint retrogressive:

This aims to improve team, not the product. This makes the scrum completely different from the other framework. What went well, what need to and identity area of improvement?

Why to use agile: [17]

Agile development methodology actually keeps the track of the direction of the project. This is done by using sprint at the end of which it a progress is shown or we can say a potentially shippable product is formed.

Advantages of the agile:

- It provides backward scalability means in the traditional waterfall model it is not easy to change decision that were made in the early phase. As any change in between will require, making the software from starting. Whole product will get damage if made any change in between. So we have to be stricter for the requirement what we need in the very beginning. But in agile requirement are stated than they are prioritizing and then implemented. Each sprint period a product shippable formed.
- The flexibility of error checking during any phases makes agile methodology more popular. But in the waterfall `model one check error very late. Means it can be done only after the development of module
- Agile also allows changes according to the customer's requirements. As in the phase of sprint review we take feedback from the customers. But waterfall does not allows modification in midway
- Actually Agile works on inspect and adapt phenomena. thus decreases both time and cost of the software

Kanban (development):

A methodology used for managing knowledge. Scrum has some negative points that's un-clear development task and task switching and partially don't work. Kaban overcomes these inabilities. Let's have a look on Kanban. The development steps are [19]:

- onboard
- in progress
- and done

Work flow divided into the on deck, specify, critical design review, execute, review, done. Kanban actually allows visualizing the work flow. It means that anyone can know what is going on. Thus allows high level of transparency to all people. In this we focus on adding value to the customer. They don't add value is removed. These value added service add quality into the software. In this one we put a limit on work in progress as it is difficult the team member to control many works at a time this means that in a sprint of two to three week we put only two to three tasks at a time. And we also put finite capacity to each level called manage flow. Scrum delivers features one bucket at a time Kanban delivers features in steady streams because of this scrum have several partially complete task and the Kanban have one 100% complete task

## CONCLUSION AND FUTURE WORK:

This report based on the study of the maintenance and the agile methodology and how it differs from the traditional methodology. The objective is to help software engineers to understand the key characteristics of these processes and therefore select the most suitable process with respect to the type of software projects they develop and to use agile methodology in maintaining software. As the maintaining software is more difficult than to build it, or even can cost much more than the original software.

Future work may be to consider other agile methodology like LSD etc.

## REFERENCES:

[1,2,6,7,8] K.K. Aggarwal, Yogesh singh ,"software engineering", 3rd edition.

[3] Albrecht, A.J. and Gaffney, J.E. Software Function, Source Lines of Code, and Development Effort Prediction - a Software Science Validation. Ieee Transactions on Software Engineering, 9 (6). 639-648.

[4] Shen, V.Y., Yu, T.J., Thebaut, S.M. and Paulsen, L.R. Identifying Error-Prone Software - an Empirical-Study. Ieee Transactions on Software Engineering, 11 (4). 317-324.

[9] http://www.vlegaci.com/can-agile-methods-work-for-software-maintenance-part-1/

[10] Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects.

[11] Agile Processes in Software Engineering and Extreme Programming: 9Th International Conference, XP 2008, Limerick, Ireland, June 10-1

[12] S. Nerur and V. Balijepally, "Theoretical Re -fl ections on Agile Development MethodologY," Comm. ACM, vol. 50, no. 3, 2007, pp. 79–83.

S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of Migrating to Agile Methodology," Comm. ACM, vol. 48, no. 5, 2005, pp. 72–78.

P. Schuh, Integrating Agile Development in the Real World, Charles River Media, 2004. 4, 2008 : Proceeding

Mikio Aoyama, "Agile Software Process Model", IEEE Software, pp 454-455s.

[13] Kieran conboy and Sharon coyle,"People over process: Key Challenges in agile development" IEEE Software.

[14,15]http://referaat.cs.utwente.nl/conference/15/paper/7269/agile-software-development-and-maintainability.pdf

[16]http://i-proving.com/wp content/uploads/2008/04/shaw_paper_agile20071.pdf

[17] Roger Valade , "The Big Projects Always Fail: Taking an Enterprise Agile", Agile 2008 Conference.

[18]What is XP, Ron Jeffries, http://www.xprogramming.com/xpmag/whatisxp.htm, extracted on 11/02/2006

[19] http://en.wikipedia.org/wiki/Kanban_(development)

[20] http://www.youtube.com/watch?v=OJflDE6OaSc