Volume 4, Issue 6, June 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

An Agile Methodology Based Model for Software development

Gurleen Singh
Senior System Administrator
HSBC Software Development Pvt. Ltd.
Pune, India

Tamanna
Assistant Professor
Chandigarh University
Gharuan, Mohali,India

Abstract- 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. Nimble practices to develop software projects like SCRUM, Extreme programming (XP), Feature driven Development (FDD), Adaptive software development (ASD) etc are increasingly being used to develop software using an adaptation approach rather than a predictive one [1]. This paper basically reviews different agile methodologies, their characteristics, principles, how they are divergent from the conventional process methods, pros and cons of using agile methodology.

Keywords- Agile Software Development, Extreme Programming, SCRUM, Feature driven development.

I. INTRODUCTION

With the further development of computer technology, the software development process has some new goals and requirements. In order to adapt to these changes, people has optimized and improved the previous method. At the same time, some of the traditional software development methods have been unable to adapt to the requirements of people. Therefore, in recent years there have been some new lightweight software process development methods, that is agile software development. The concept of agile development was proposed in 2001 by the agile team, and then many software development teams and companies recognized and accepted it, and gradually been widely used in many projects [2]. Agile Software Development published the Agile Manifesto shown in figure I at the same time, on behalf of software development has entered a new era.

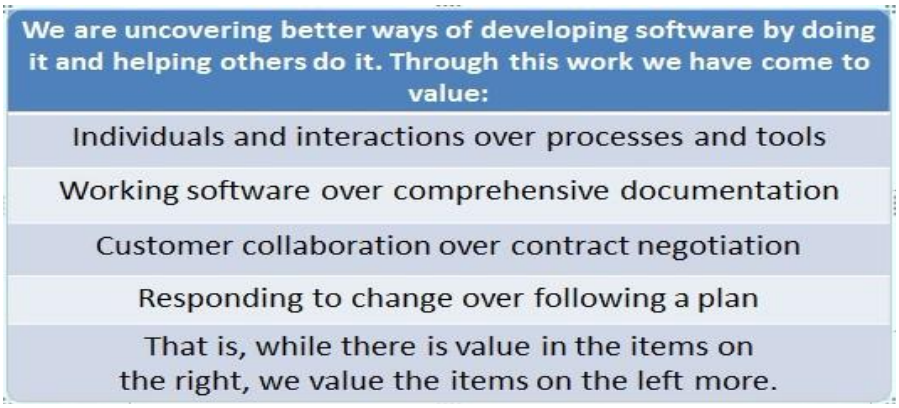


Fig. 1 Manifesto for Agile Software Development

II. AGILE DEVELOPMENT AND PRINCIPLES

The Agile Manifesto introduced the term in 2001. Since then, the Agile Movement, with all its values, principles, methods, practices, tools, champions and practitioners, philosophies and cultures, has significantly changed the landscape of the modern software engineering and commercial software development [3]. The methodologies originally embraced by the Agile Alliance were Adaptive Software Development (ASD) [4], Crystal [5], Dynamic Systems Development Method (DSDM) [6], Extreme Programming (XP) [7], Feature Driven Development (FDD) [8] and Scrum [9].The Agile Alliance also documented the principles they follow that underlie their manifesto:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.
4. Business people and developers must work together daily through the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development.
9. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
10. Continuous attention to technical excellence and good design enhances agility.
11. Simplicity – the art of maximizing the amount of work not done – is essential.
12. The best architectures, requirements, and designs emerge from self-organizing teams.
13. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly [10].

III. CHARACTERISTICS OF AGILE SOFTWARE PROCESSES

A. *Modularity*: Modularity is a key element of any good process. Modularity allows a process to be broken into components called activities. A software development process prescribes a set of activities capable of transforming the vision of the software system into reality.

B. *Iterative*: Agile software processes acknowledge that we get things wrong before we get them right. Therefore, they focus on short cycles. Within each cycle, a certain set of activities is completed. These cycles will be started and completed in a matter of weeks. However, a single cycle (called an iteration) will probably not be enough to get the element 100% correct. Therefore, the short cycle is repeated many times to refine the deliverables.

C. *Time-Bound*: Iterations become the perfect unit for planning the software development project. We can set time limits (between one and six weeks is normal) on each iteration and schedule them accordingly. Chances are, we will not (unless the process contains very few activities) schedule all of the activities of our process in a single iteration. Instead, we will only attempt those activities necessary to achieve the goals set out at the beginning of the iteration. Functionality may be reduced or activities may be rescheduled if they cannot be completed within the allotted time period.

D. *Parsimony*: Agile processes are more than just a traditional software development process with some time constraints. Attempting to create impossible deadlines under a process not suited for rapid delivery puts the onus on the software developers. This leads to burnout and poor quality. Instead, agile software processes focus on parsimony. That is, they require a minimal number of activities necessary to mitigate risks and achieve their goals.

E. Adaptive: During an iteration, new risks may be exposed which require some activities that were not planned. The agile process adapts the process to attack these new found risks. If the goal cannot be achieved using the activities planned during the iteration, new activities can be added to allow the goal to be reached.

F. *Incremental:* An agile process does not try to build the entire system at once. Instead, it partitions the nontrivial system into increments which may be developed in parallel, at different times, and at different rates. We unit test each increment independently. When an increment is completed and tested, it is integrated into the system.

G. *Convergent*: Convergence states that we are actively attacking all of the risks worth attacking. As a result, the system becomes closer to the reality that we seek with each iteration. As risks are being proactively attacked, the system is being delivered in increments.

H. *People-Oriented:* Agile processes favor people over process and technology. They evolve through adaptation in an organic manner. Developers that are empowered raise their productivity, quality, and performance. After all, they are the best individuals in the organization to know how to make these changes.

I. *Collaborative*: Agile processes foster communication among team members. Communication is a vital part of any software development project. When a project is developed in pieces, understanding how the pieces fit together is vital to creating the finished product. There is more to integration than simple communication. Quickly integrating a large project while increments are being developed in parallel, requires collaboration [11].

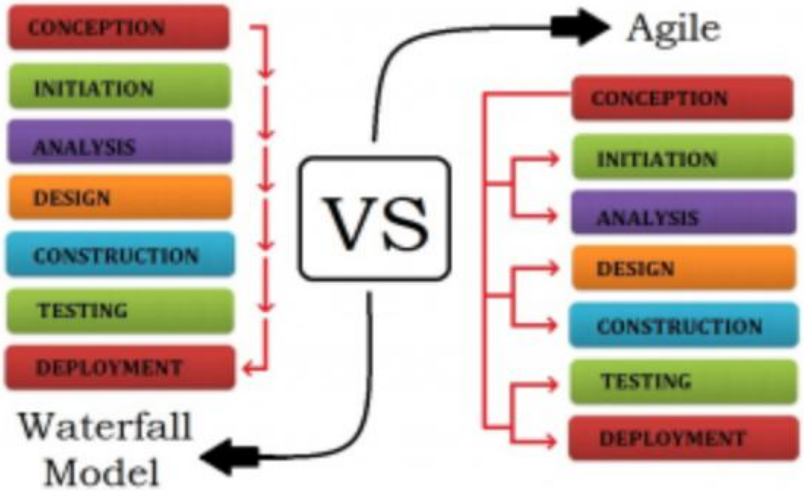
IV. AGILE Vs WATERFALL

A. The main advantage is the backward scalability in Agile. Under waterfall approach we cannot change the decisions and implementations that we had made under the previous stages. If we want to make changes under waterfall we will have to build the entire project from the scratch once again.

B. The flexibility to error check under any part of the development stage makes Agile more bug free and less erroneous as compared to Waterfall which can only test bugs at the end of the development module.

C. Since Agile provides flexibility to make changes as per customer requirements it is more inclined towards better client satisfaction. This is a real setback for the Waterfall model which doesn't allow any modifications once the module has been completed.

D. Under Agile development modular partitioning of the software can be effectively carried out as compared to its counterpart. Though both of them allows option for segregation the later lacks the modifications in the implementation stage. The rules are set down before the commencement of the project hence it hinders further break down of the logical module. Whereas Agile can be of great help under such situations and can allow simultaneous development of different modules at the same time as per time bound. If we want the project to be more segregated Agile comes as a pain relief for developers [12].



Fig, 2 Agile Vs Waterfall

V. DIFFERENT AGILE DEVELOPMENT METHODS

A. Extreme Programming: Extreme Programming (XP) is a methodology for creating software within a very unstable environment. It allows flexibility within the modeling process. The main goal of XP is to lower the cost of change in software requirements. With traditional system development methodologies, like the Waterfall Methodology, the requirements for the system are determined and often "frozen" at the beginning of the development project. This means that the cost of changing the requirements at a later stage in the project - something that is very common in the real-world can be very high. It is based on 12 principles:

- *The Planning Process*: The desired features of the software, which are communicated by the customer, are combined with cost estimates provided by the programmers to determine what the most important factors of the software are. This stage is sometimes called the *Planning Game*.
- *Small Releases*: The software is developed in small stages that are updated frequently, typically every two weeks.
- *Metaphor*: All members on an XP team use common names and descriptions to guide development and communicate on common terms.
- *Simple Design*: The software should include only the code that is necessary to achieve the desired results communicated by the customer at each stage in the process. The emphasis is not on building for future versions of the product.
- *Testing*: Testing is done consistently throughout the process. Programmers design the tests first and then write the software to fulfill the requirements of the test. The customer also provides acceptance tests at each stage to ensure the desired results are achieved.
- *Refactoring:* XP programmers improve the design of the software through every stage of development instead of waiting until the end of the development and going back to correct flaws.
- *Pair Programming*: All code is written by a pair of programmers working at the same machine.
- *Collective Ownership:* Every line of code belongs to every programmer working on the project, so there are no issues of proprietary authorship to slow the project down. Code is changed when it needs to be changed without delay.
- *Continuous Integration*: The XP team integrates and builds the software system multiple times per day to keep all the programmers at the same stage of the development process at once.

- *40-Hour Week*: The XP team does not work excessive overtime to ensure that the team remains well-rested, alert and effective.
- *On-Site Customer*: The XP project is directed by the customer who is available all the time to answer questions, set priorities and determine requirements of the project.
- *Coding Standard*: The programmers all write code in the same way. This allows them to work in pairs and to share ownership of the code [13].

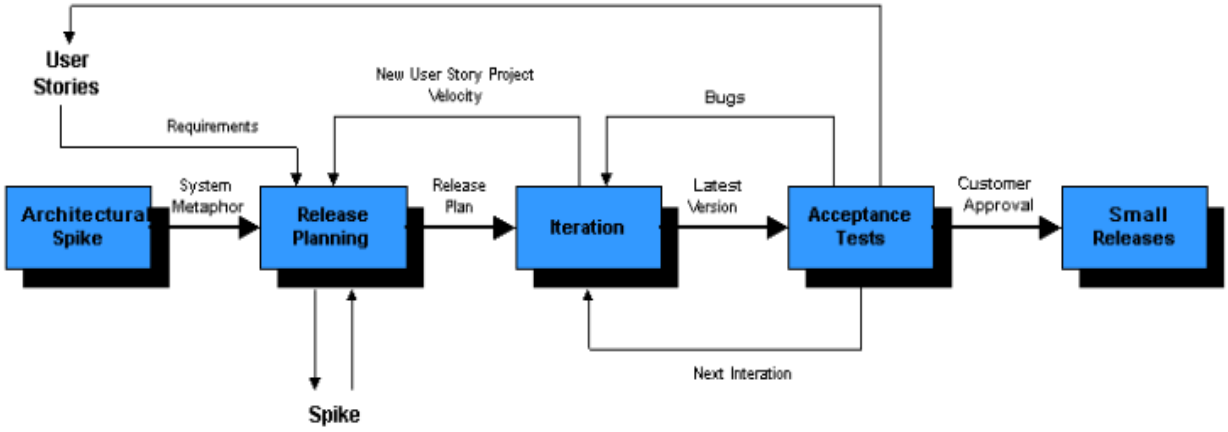


Fig. 3 Extreme programming [14]

B. *Scrum:* This is another light weight method victimize for the development of software. Its principle lies in the fact that small teams working cross functionally produce good results. Scrum is more revenue centric with attention on improving revenue and quality of the software. Since being lightweight it can adapt to changing requirements and releases the software in small release cycles called sprints. Scrum has three roles:

- *Product Owner:* In Scrum, the Product Owner is responsible for communicating the vision of the product to the development team. He or she must also represent the customer's interests through requirements and prioritization. Because the Product Owner has the most authority of the three roles, it's also the role with the most responsibility.
- *Scrum Master:* The Scrum Master acts as a facilitator for the Product Owner and the team. The Scrum Master does not manage the team. Instead, he or she works to remove any impediments that are obstructing the team from achieving its sprint goals. In short, this role helps the team remain creative and productive.
- *Team Member:* In the Scrum methodology, the team is responsible for completing work. Ideally, teams consist of seven cross-functional members, plus or minus two individuals. For software projects, a typical team includes a mix of software engineers, architects, programmers, analysts, QA experts, testers, and UI designers. Each sprint, the team is responsible for determining how it will accomplish the work to be completed. This grants teams a great deal of autonomy, but, similar to the Product Owner's situation, that freedom is accompanied by a responsibility to meet the goals of the sprint.[15]

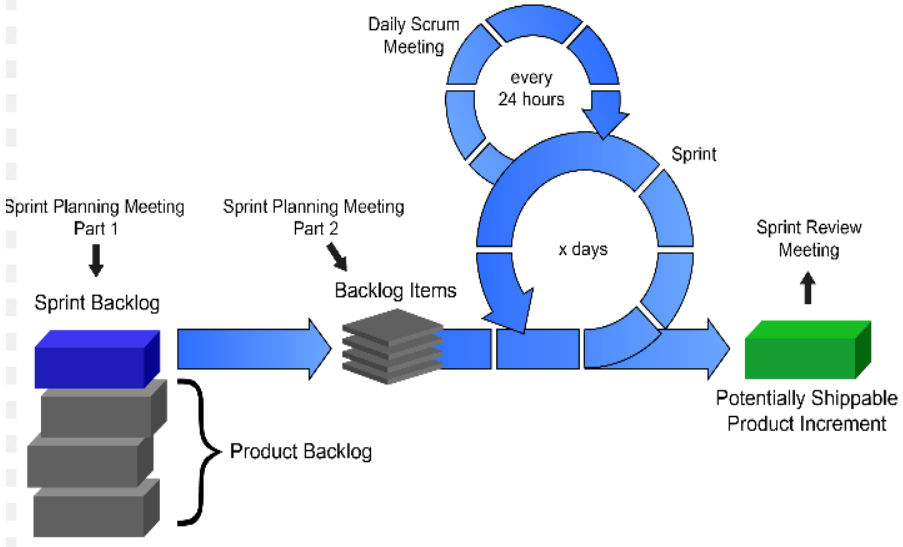


Fig. 4 Scrum [16]

C. Crystal

The Crystal methodology is one of the most lightweight, adaptable approaches to software development.

- Crystal is actually comprised of a family of agile methodologies such as Crystal Clear, Crystal Yellow, Crystal Orange and others, whose unique characteristics are driven by several factors such as team size, system criticality, and project priorities.
- This addresses the realization that each project may require a slightly tailored set of policies, practices, and processes in order to meet the project's unique characteristics.
- Several of the key tenets of Crystal include teamwork, communication, and simplicity, as well as reflection to frequently adjust and improve the process.
- Like other agile methodologies, Crystal promotes early, frequent delivery of working software, high user involvement, adaptability, and the removal of bureaucracy or distractions.

D. *Feature-Driven Development (FDD)*

The Feature-Driven Development (FDD) approach focuses on the software features of the system. They are the main driver of the entire development process. It differs significantly from the other agile processes because they put a strong emphasis on planning and upfront design. As shown in Figure 4,

- The first step of the FDD process is to build a detailed model of the system, which captures all the stakeholders' assumptions and requirements.
- Once the domain model is built, the team members print a list of the features of the system. Each feature has to be developed in a few hours or days, but no longer than 2 weeks.
- The development work is performed in parallel on all the features. Each team is headed by a feature owner, who is responsible for the code segment that implements those features.
- The FDD process enforces rigorous guidelines in order to find defects in the system. It also enforces coding standards and encourages regular builds on a daily or weekly basis in order to add freshly designed features to the base system.
- All the features are developed in parallel, it is mandatory to have a configuration management system that allows calmly integrating of the changes that are made to the system [1].

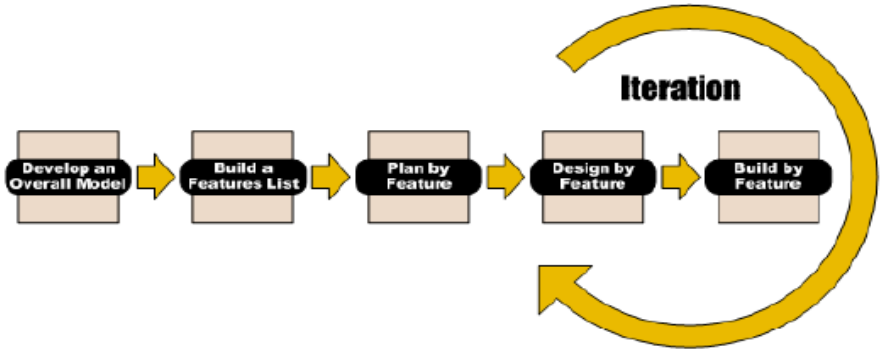


Fig.4 Feature-Driven Development

VI. BENEFITS OF USING AGILE SOFTWARE DEVELOPMENT

A. Improved communication and coordination among team members. Specifically, the daily scrums were seen as instrumental, and were especially useful to bring testers and developers together.

B. Quick Releases is consequence of Continuous Integration, Developers create demo-able releases every few weeks instead of every few months or years.

C. Developers noted that short sprints combined with more emphasis on customer feedback led to better agility and efficiency in responding to changing requirements, internal processes, reorganizations or politics, and flushed out bad designs more quickly.

D. Most popular benefit of ASD is a More Reasonable Process. Many developers complained about rigid development processes that were relaxed in an Agile environment.

E. The process supports real-time tracking of progress and ability to adjust future forecasts based on real data. Agile methodologies are more dynamic and incur less overhead [17].

VII. PROBLEMS WITH AGILE SOFTWARE DEVELOPMENT

A. Works for small co-located teams, but not for complex large projects. It can be difficult for larger teams to be as flexible as smaller teams with respect to design and architectural changes. Scrum meetings were sometimes considered inefficient, especially when the team was inexperienced with Agile.

B. Scrum meetings involve all members of a team and often occur daily. These scrums can be often inefficient of especially when they were poorly run by a Scrum Master who was not disciplined and focused enough to run the meeting quickly.

C. Management buy-in is also a concern. Many program managers in agile development are worried that upper-level management would ask for progress reports and productivity metrics that would be hard to gather in an agile work environment. Upper management still tries to get specific dates for specific deliverables.

D. Another concern is coordinating with other teams. This is especially worrisome in larger projects where only a few groups are Agile and the rest are using a typical Waterfall model. Problems arise in the scheduling of deliverables between dependent projects [17].

VIII. CONCLUSION

Agile software development stresses rapid iterations, small and frequent releases, and evolving requirements facilitated by direct user involvement in the development process. Agile processes are not a new phenomenon. They are the evolution of the best practices which have been refined over the past thirty years. They are not a silver bullet which will cure your project of all of its ills or guarantee success. Successfully delivering a project requires hard work and the understanding of the potential pitfalls. No single process will work for every project. Yet, most projects would agree that they would like to be more agile. Developing a more agile process for your project requires an understanding of the dynamics of these projects.

REFERENCES

[1] Kaushal Pathak and Anju Saha, *"Review of Agile Software Development Methodologies,"* International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 2, February 2013.

[2] K. B. Marten Folwer, James A.Highsmith, Manifesto for Aigle Software Development, http://agilemanifesto.org/, 11 2011.

[3] http://en.wikipedia.org/wiki/Agile_software_development

[4] J. Highsmith, *Adaptive Software Development*. New York, NY: Dorset House, 1999.

[5] A. Cockburn, *Agile Software Development*. Reading, Massachusetts: Addison Wesley Longman, 2001.

[6] J. Stapleton, *DSDM: The Method in Practice*, Second ed: Addison Wesley Longman 2003.

[7] K. Beck, *Extreme Programming Explained: Embrace Change*. Reading, Mass.: Addison-Wesley, 2000.

[8] S. R. Palmer and J. M. Felsing, *A Practical Guide to Feature-Driven Development*. Upper Saddle River, NJ: Prentice Hall PTR, 2002.

[9] K. Schwaber and M. Beedle, *Agile Software Development with SCRUM*. Upper Saddle River, NJ: Prentice-Hall, 2002.

[10] Laurie Williams, "A Survey of Agile Development Methodologies," 2007.

[11] Granville G. Miller, "The Characteristics of Agile Software Processes," Proceedings of the 39th Int'l Conf. and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS'01).

[12] http://www.sdlc.ws/agile-vs-waterfall/

[13] http://www.webopedia.com/TERM/E/Extreme_Programming.html

[14] https://www.google.co.in/search?q=extreme+programming&source=lnms&tbm=isch&sa=X&ei=D9VlU77hEIaGuASO2oKIBQ&sqi=2&ved=0CAYQ_AUoAQ&biw=1138&bih=536#facrc=_&imgdii=_&imgrc=Bg3ep7MiIBdKnM%253A%3BJdRPmJ0yMUXHOM%3Bhttp%253A%252F%252Fwithfriendship.com%252Fimages%252Fd%252F18667%252FExtreme-Programming-image.jpg%3Bhttp%253A%252F%252Fwithfriendship.com%252Fuser%252Fcyborg%252Fextreme-programming.php%3B768%3B320

[15] http://scrummethodology.com/

[16] https://www.google.co.in/search?q=scrum&source=lnms&tbm=isch&sa=X&ei=svllU-mgL8idugTGz4DAAw&sqi=2&ved=0CAYQ_AUoAQ&biw=1138&bih=536#facrc=_&imgdii=_&imgrc=SVI-DN1QSKrjpM%253A%3Bi0c1hSjpVPrD0M%3Bhttp%253A%252F%252Fcomsysto.files.wordpress.com%252F2010%252F06%252Fscrum_cycle.png%3Bhttp%253A%252F%252Fquoteko.com%252Fscrum-cycle.html%3B3094%3B2012

[17] Andrew Begel and Nachiappan Nagappan, " Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study".

