Agile Methodology – A New Framework for Software Development

Rachna Soni 1 and Gurmeet Kaur2
1D. A.V. College Yamuna Nagar
2 DIET, Karnal
1rsoni63@gmail.com and 2grmtkaur02@gmail.com

ABSTRACT
Agile methodology is a conceptual framework where software is developed in iterations. There are various methodologies based on agile computing that can be used to minimize risk and maximize productivity by developing software in short iterations and de-emphasizing work on secondary or interim work artifacts. This paper deals with the use of agile computing in software development. The methodologies used in agile computing are discussed in detail with their pros and cons.

1. INTRODUCTION
Agile computing may be defined as opportunistically (or on user demand) discovering and taking advantage of available resources in order to improve capability, performance, efficiency, fault tolerance, and survivability. The term agile is used to highlight both the need to quickly react to changes in the environment as well as the need to exploit transient resources only available for short periods of time. Agile computing builds on current research in grid computing, ad-hoc networking, peer-to-peer resource sharing and software development methodologies. Agile computing is also useful in software development methodology. It is very effective where Client frequently changes his requirement. Since it has more iteration so you can assure a solution that meets clients requirement. Agile methodology is used for software development, including XP (Extreme programming), and it advocates iterator development. Agile methodology is more of people oriented. Agile methodology helps us to increase productivity and reduce risks. Agile methodology is a conceptual framework where software is developed in iterations. Each iterations has Requirement analysis, planning, design, coding, testing and documentation. [1]

2. PRINCIPLE OF AGILE COMPUTING
Four principles that constitute Agile methods are: 1. The reigning supreme of individuals and interactions over processes and tools. 2. As does, working software over comprehensive documentation. 3. Likewise, customer collaboration over contract negotiation. 4. And again, responding to change over plan follow-throughs.

3. AGILE METHODOLOGY- CHARACTERISTICS
- Many builds are delivered in the iteration process
- Accepts change of requirement at any stage
- Requires close communication between business, Development and Testing people
- Reduced risk and time to develop.
- Less documentation work compared to other methodologies
- Requires continuous testing
- Frequent Delivery
- More Iterations
- Test frequently
- Less defects

4. WATER FALL MODEL VS. AGILE COMPUTING
By focusing on the repetition of abbreviated work cycles as well as the functional product they yield, agile methodology could be described as “iterative” and “incremental.” In waterfall, development teams only have one chance to get each aspect of a project right. In an agile paradigm, every aspect of development — requirements, design, etc. — is continually revisited throughout the lifecycle. The difference between the two, one can say the classic waterfall method stands for predictability, while Agile methodology spells adaptability. Agile methods are good at reducing overheads, such as, rationale, justification, documentation and meetings, keeping them as low as is possible. Another Agile method advantage is one has a launchable product at the end of each tested stage. This ensures bugs are caught and eliminated in the development cycle, and the product is double tested again after the first bug elimination. This is not possible for the Waterfall method, since the product is tested only at the very end, which means any bugs found results in the entire programme having to be re-written. Agile methods allow for specification changes as per end-user’s requirements, spelling customer satisfaction. As already mentioned, this is not possible when the waterfall method is employed, since any changes to be made means the project has to be started all over again.

5. METHODOLOGIES FOR AGILE COMPUTING
There are different flavors of agile development by which this methodology can be implemented: - Agile Manifesto, XP
Scrum projects make progress in a series of sprints, which are timeboxed iterations no more than a month long. The first activity of each sprint is a sprint planning meeting. During this meeting the product owner and team talk about the highest-priority items on the product backlog. The product backlog is a complete list of the functionality that remains to be added to the product. The product backlog is prioritized by the product owner so that the team always works on the most valuable features first. The most popular and successful way to create a product backlog is to populate it with user stories, which are short descriptions of functionality described from the perspective of a user or customer.

Team members figure out how many items they can commit to and then create a sprint backlog, which is a list of the tasks to perform during the sprint. On each day of the sprint, a daily scrum meeting is attended by all team members, including the Scrum Master and the product owner. This meeting is timeboxed to no more than fifteen minutes. During that time, team members share what they worked on the prior day, will work on today, and identify any impediments to progress. Daily scrums serve to synchronize the work of team members as they discuss the work of the sprint.

At the end of a sprint, the team conducts a sprint review. During the sprint review, the team demonstrates the functionality added during the sprint. The goal of this meeting is to get feedback from the product owner or any users or other stakeholders who have been invited to the review. This feedback may result in changes to the freshly delivered functionality. But it may just as likely result in revising or adding items to the product backlog. Another activity performed at the end of each sprint is the sprint retrospective. The whole team participates in this meeting, including the ScrumMaster and product owner. The meeting is an opportunity to reflect on the sprint that is ending and identify opportunities to improve in the new sprint.

At the end of the sprint a “sprint review is conducted” during which the team demonstrates the new functionality to the product owner and other interested stakeholders who provide feedback that could influence the next sprint. At the end of the sprint, these features are done---they are coded, tested, and integrated into the evolving product or system.

6.4 THE SCRUM CONSTRUCTION LIFE CYCLE

Figure uses the terminology of the Scrum methodology. Figure shows how agileists treat requirements like a prioritized stack, pulling just enough work off the stack for the current iteration (in Scrum iterations/sprints are often 30-days long, although this can vary). At the end of the iteration the system is demoed to the stakeholders to verify that the work that the team promised to do at the beginning of the iteration was in fact accomplished.
Agile Methodology – A New Framework for Software Development

The Scrum construction life cycle of Figure, although attractive proves to be a bit naive in practice. Where does the product backlog come from? Does it get beamed down from the Starship Enterprise? Of course not, it's actually the result of initial requirements envisioning early in the project. You don't only implement requirements during an iteration, you also fix defects (disciplined agile teams have a parallel testing effort during construction iterations where these defects are found), go on holiday, support other teams (perhaps as reviewers of their work), and so on. So you really need to expand the product backlog into a full work items list. You also release your system into production, often a complex endeavor.[2]

7. EXTREME PROGRAMMING (XP)

Extreme Programming (XP) is one of the more well known Agile methodologies. As its name suggests it is a programmer-centric methodology that emphasizes technical practices to promote skilful development through frequent delivery of working software. In fact, full adoption of all of XP’s practices requires a high level of discipline, teamwork, and skill. Agile processes, exemplified by eXtreme Programming (XP) have a great deal to offer Web services and performance programmers. These processes emphasize small teams, intensive collaboration, very short development cycles, and an approach to coding that includes writing tests first.

Extreme Programming (XP) has iteration period would be less then in scrum, which is being 2-4 weeks. Here developer prioritise what to do first on the basis of client requirement. This duration which was being fixed for a iteration, can be increase if the some development part is still pending. The build would deployed with having all the client needs. Thus iteration period is not fixed here it can be increase. but iteration should meets all the client's requirement in this build. More attension is required for testing in XP.[3] But XP is most commonly known for its technical practices. At the heart of XP are four core values: communication, simplicity, feedback, and courage. From these values 13 practices are derived. These practices are:

1. Planning Game: Plan work incrementally.
2. Small Releases: Release as quickly as possible to increase time to market, and get feedback as soon as possible.
3. Metaphor: If possible, define a metaphor for the system being developed. For example, the shopping cart metaphor is widely used to describe an online ordering system.
4. Simple Design: Use the simplest design that will work for the functionality (user story) being implemented. Do not design for things that may never actually be used.
5. Testing: Test everything, and try to automate the testing if possible.
6. Refactoring: Instead of designing the entire system up front, design as you go, making improvements as needed. Change the implementation without changing the interface to the functionality, and use automated testing to determine the impact of the refactoring.
7. Pair Programming: Programming in teams of two (or three) allow for a discussion to occur in real-time that addresses requirement, design, testing, and programming concerns.
8. Collective Code Ownership: Anyone on the team can make a change to any code at any time. Collective Code Ownership: Anyone on the team can make a change to any code at any time.
9. Continuous Integration: The entire code base is constantly being rebuilt, and retested in an automated fashion.
10. Sustainable Pace: Ideally, team members do not need to work more than 40 hours per week to meet project deadlines. Burning the midnight oil is chunked by management in favor of consistent, predictable, repeatable delivery.
11. Whole Team: The team functions as a whole. Members are encouraged to be more generalized than specialized. Learning about all technologies and requirements is encouraged.
12. Coding Standards: In order to maximize communication, coding standards are defined by the team, and used to ensure consistent coding practices.
13. Onsite Customer: Having constant and direct access to the customer allows the team to work at the fastest possible speed.[6]

7.2 CHARACTERSTICS OF XP PROGRAMMING

- The primary requirement artifact in XP is the user story. Visibly, a user story is nothing more than a note card with a short description on it. But user stories actually are comprised of the card (reminder of
promised functionality), conversations between developers and requirements donors, and tests (all types: unit, integration, acceptance, etc.).

- One of the characteristic of XP is its cycle time and level of ceremony. XP recommends very short iterations between one and four weeks.

8. PROS AND CONS OF AGILE METHODOLOGIES
Agile methodologies are faster time to market and increased productivity. Other benefits of switching to agile include fewer software defects and reduced software development costs. Agile processes can also help improve security. Applications developed using an agile process end up being more secure. Agile methods can also benefit software outsourcing providers. Agile isn't perfect. While organizations are reaping benefits with agile, practitioners acknowledge there are some challenges to this style of development, the top Agile challenge, followed by documentation. Resistance to change and tool integration were also cited as challenges. Agile can be particularly challenging when you're dealing with distributed teams. There are a handful of major agile testing perils. As an agile tester, you are expected to test without having formal requirement documents, to test in real time, to test changing code, to test on changing requirements, to automate most of your tests and to be a part of a close-knit team. Issues to watch for include waiting for a specific build (in agile, you need to test constantly); trying to test everything manually (using automation is key).[5]

CONCLUSION & FUTURE SCOPE
Finally we can say that software based on agile methodologies are completed in time and increases productivity in market. The advantages of Agile Management & development methods are in design issues, budgets, scope, documentation planning and scheduling. Higher flexibility to change of Management & development plans, higher quality by earlier feedback from the customers. With agile methods the user is deeply involved in deciding which stories have priority for implementation. Hence the trade-off of one piece function against another are much clearer. More companies would benefit from embracing agile project management

REFERENCES
[2]. Roman Pichler “Agile Product Management with Scrum”: Creating Products that Customers Love
[3]. Martin Fowler “The New Methodology” ,13 December 2005
[4]. Kevin Aguanno, PMP “Managing Agile Projects” , MAPM [Agile Management / Project Management]
[5]. By Mike Cohn “Succeeding with Agile”