Volume 4, Issue 5, May 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Study on Agile Methodology: A New Line Towards Business Applications

Deepanshu Thakral,
M. Tech Scholar, ACTM, Palwal, Faridabad, India

Mahesh Singh
Asst. Prof (CSE Department), ACTM, Palwal, India

Abstract: ***The paper presents a general understanding of how agile methodology is applied in making projects works. This approach includes the concepts applied, the tools used and the activities conducted for successfully delivering web business applications. By purchasing the Agile Platform and Agile Network, organizations are able to adopt 'Agile' development methods and scale their agile projects across their IT organization. Agile methodology is an approach to project management typically used in software development. It helps teams respond to the unpredictability of building software through incremental, iterative work cadences, known as sprints. At OutSystems, we believe that being Agile requires not only a set of adaptive processes but also tools that make everyone involved with the project, agile. This includes the complete Agile delivery team—management, development team members and business users. Agile methods for software development have brought attention in recent years. Scrum and XP are most popular among these and have been a topic of charm during this. Though agile methods have shown encouraging results in organizations which adopted them, yet a very little is known about the effectiveness of agile.***

Keywords: Agile, Scrum, Extreme Programming

I. INTRODUCTION

This methodology is an approach to project management, typically used in software development. It helps teams respond to the unpredictability of building software through incremental, iterative work cadences, known as sprints. It is a framework that promotes development iterations throughout the life-cycle of a project. It minimizes risk by developing software in short amounts of time. First of all, it assumes that every requirement of the project can be identified before any design or coding occurs. A development accomplished in one unit of time is called iteration. Each iteration is a project with analysis, design, coding, testing and also documentation. All the required functionality may not be covered in one iteration for releasing the project. But it will be covered in multiple iterations. The idea is to have a defect free release available at the end of each iteration. Agile development methodology attempts to provide many opportunities to assess the direction of a project throughout the development lifecycle. This is achieved through regular cadences of work, known as sprints or iterations, at the end of which teams must present a shippable increment of work. Thus by focusing on the repetition of abbreviated work cycles as well as the functional product they yield, agile methodology could be described as Â"iterativeÂ" and Â"incremental.Â" In waterfall, development teams only have one chance to get each aspect of a project right. In an agile paradigm, every aspect of development Â— requirements, design, etc. Â— is continually revisited throughout the lifecycle. When a team stops and re-evaluates the direction of a project every two weeks, there Â's always time to steer it in another direction.

A. Waterfall vs. Agile methodology

There is no IT meeting that does not talk and debate endlessly about Waterfall vs. Agile development methodologies of Waterfall. A classically linear and sequential approach to software design and systems development, each waterfall stage is assigned to a separate team to ensure greater project and deadline control, important for on-time project delivery.

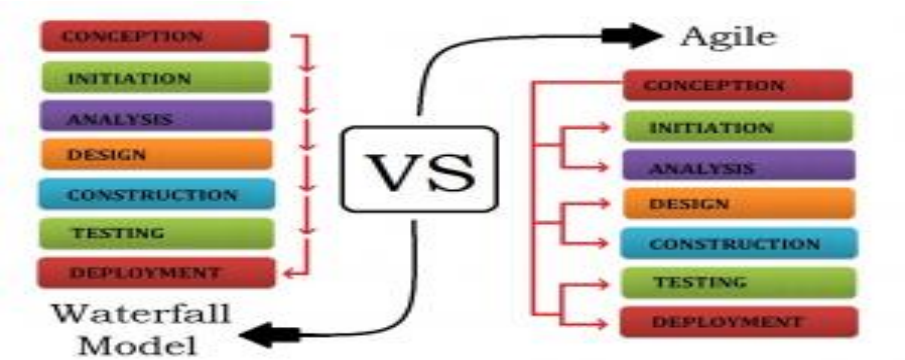

Figure 1.1 Waterfall Model Vs Agile Methodology

© 2014, IJARCSSE All Rights Reserved

A linear approach means a stage by stage approach for product building. The various points in this approach are considered as follows:

1. The project team first analyses, then determining and prioritizing business requirements / needs.
2. Next, in the design phase business requirements are translated into IT solutions, and a decision taken about which underlying technology i.e. COBOL, Java or Visual Basic, etc. etc. is to be used.
3. Once processes are defined and online layouts built, code implementation takes place.
4. The next stage of data conversion evolves into a fully tested solution for implementation and testing for evaluation by the end-user.
5. The last and final stage involves evaluation and maintenance, with the latter ensuring everything runs smoothly.

As for minimal risk Agile, it is a low over-head method that emphasizes values and principles rather than processes. Working in cycles i.e. a week, a month, etc., project priorities are re-evaluated and at the end of each cycle. 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.

Agile methodology means cutting down the big picture into puzzle size bits, fitting them together when the time is right e.g. design, coding and testing bits. So, while there are reasons to support both the waterfall and agile methods, however, a closer look clarifies why many software and web design firms make the more appropriate choice of employing Agile methodology.

Once a stage is completed in the Waterfall method, there is no going back, since most software designed and implemented under the waterfall method is hard to change according to time and user needs. The problem can only be fixed by going back and designing an entirely new system, a very costly and inefficient method. Whereas, Agile methods adapt to change, as at the end of each stage, the logical programme, designed to cope and adapt to new ideas from the outset, allows changes to be made easily. With Agile, changes can be made if necessary without getting the entire programme rewritten. This approach not only reduces overheads, it also helps in the upgrading of programmes.

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's modular nature means employing better suited object-oriented designs and programmes, which means one always has a working model for timely release even when it does not always entirely match customer specifications. Whereas, there is only one main release in the waterfall method and any problems or delays mean highly dissatisfied customers.

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. However, both methods do allow for a sort of departmentalization e.g. in waterfall departmentalization is done at each stage. As for Agile, each coding module can be delegated to separate groups. This allows for several parts of the project to be done at the same time, though departmentalization is more effectively used in Agile methodologies.

In conclusion, though on the plus side, waterfall's defined stages allow for thorough planning, especially for logical design, implementation and deployment, Agile methodology is a sound choice for software development and web design projects. More and more firms are becoming Agile.

B. Agile Principle

Agile methods are a family of development processes, not a single approach to software development. In 2001, 17 prominent figures in the field of agile development to discuss ways of creating software in a lighter, faster, more people-centric way. They created the Agile Manifesto, widely regarded as the canonical definition of agile development, and accompanying agile principles.

Some of the principles behind the Agile Manifesto are as follows.

1. **Active user involvement is imperative.**

According to this principle, it is not always possible to have users directly involved in development projects, particularly if the Agile Development project is to build a product where the real requirements are clearly communicated and understood at the outset. Requirements are prioritized appropriately based on the needs of the user and market. Requirements can be clarified on a daily basis with the entire project team, rather than resorting to lengthy documents that aren't read or are misunderstood. Emerging requirements can be factored into the development schedule as appropriate with the impact and trade-off decisions clearly understood. The right product is delivered. As iterations of the product are delivered, the product meets user expectations. The product is more intuitive and easy to use. The user/business is seen to be interested in the development on a daily basis. The user/business sees the commitment of the team. Developers are accountable, sharing progress openly with the user/business every day. There is complete transparency as there is nothing to hide. The user/business shares responsibility for issues arising in development; it's not a customer-supplier relationship but a joint team effort. Timely decisions can be made, about features, priorities, issues,

© 2014, IJARCSSE All Rights Reserved

and when the product is read. Responsibility is shared; the team is responsible together for delivery of the product. Individuals are accountable, reporting for themselves in daily updates that involve the user/business.

2. **Agile Development teams must be empowered.**

An Agile Development project team must include all the necessary team members to make decisions, and make them on a timely basis. Active user involvement is one of the key principles to enable this, so the user or user representative from the business must be closely involved on a daily basis. The project team must be empowered to make decisions in order to ensure that it is their responsibility to deliver the product and that they have complete ownership. Any interference with the project team is disruptive and reduces their motivation to deliver.

3. **Time waits for no man.**

In Agile Development, requirements evolve, but timescales are fixed. This is in stark contrast to a traditional development project, where one of the earliest goals is to capture all known requirements and baseline the scope so that any other changes are subject to change control.

4. **Agile requirements are barely sufficient**.

Agile Development teams capture requirements at a high level and on a piecemeal basis, just-in-time for each feature to be developed. Agile requirements are ideally visual and should be barely sufficient, i.e. the absolute minimum required to enable development and testing to proceed with reasonable efficiency. The reason for this is to minimize the time spent on anything that doesn't actually form part of the end product.

5. **Fast but not so furious.**

Agile software development is all about frequent delivery of products. In a truly agile world, gone are the days of the 12 month project. In an agile world, a 3-6 month project is strategic. Nowhere is this is more true than on the web. The web is a fast moving place. And with the luxury of centrally hosted solutions, there's every opportunity to break what would have traditionally been a project into a list of features, and deliver incrementally on a very regular basis - ideally even feature by feature.

6. Done" means "DONE".

In agile development, "done" should really mean "DONE. Features developed within iteration (Sprint in Scrum), should be 100% complete by the end of the Sprint.

II. AGILE SOFTWARE DEVELOPMENT

This is based on fundamental changes to what we considered essential to software development ten years ago. The most important thing to know about Agile methods or processes is that there is no such thing. There are only Agile teams. The processes we describe as Agile are environments for a team to learn how to be Agile.

We realize the way a team works together is far more important than any process. While a new process can easily improve team productivity by a fraction, enabling your team to work effectively as a cohesive unit can improve productivity by several times. Of course to be eligible for such a big improvement you must be working at a fraction of your potential now. Unfortunately, it isn't that uncommon.

The biggest problem with software development is changing requirements. Agile processes accept the reality of change versus the hunt for complete, rigid specifications. There are domains where requirements can't change, but most projects have changing requirements. For most projects readily accepting changes can actually cost less than ensuring requirements will never change.

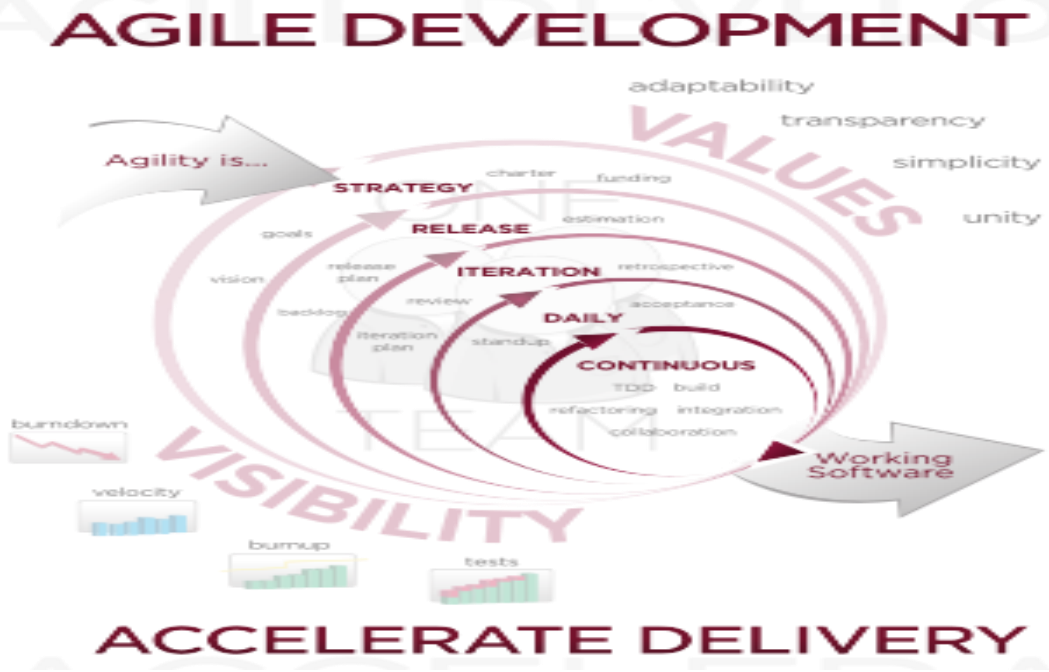

Figure 2.1 Agile Software Development

© 2014, IJARCSSE All Rights Reserved

We can produce working software starting with the first week of development so why not show it to the customer? We can learn so much more about the project requirements in the context of a working system. The changes we get this way are usually the most important to implement.

A. Agile methods

There are well-known agile software development methods. These methods include:

- Agile Modeling
- Agile Unified Process (AUP)
- Dynamic Systems Development Method (DSDM)
- Essential Unified Process (EssUP)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Open Unified Process (OpenUP)
- Scrum

Two methods are explained as follows:

1. Extreme programming

Extreme Programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development,[1][2][3] it advocates frequent "releases" in short development cycles which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted. Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, simplicity and clarity in code, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers.[2][3][4] The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels, on the theory that if some is good, more is better. Extreme programming also introduces a number of basic values, principles and practices on top of the agile programming framework. XP describes four basic activities that are performed within the software development process: coding, testing, listening, and designing. Each of those activities is described below.

(a) Coding

As we know that without code, there is no working product. Coding can also be used to figure out the most suitable solution. Coding can also help to communicate thoughts about programming problems. A programmer dealing with a complex programming problem and finding it hard to explain the solution to fellow programmers might code it and use the code to demonstrate what he or she means. Code, say the proponents of this position, is always clear and concise and cannot be interpreted in more than one way. Other programmers can give feedback on this code by also coding their thoughts.

(b) Testing

Extreme programming's approach is that if a little testing can eliminate a few flaws, a lot of testing can eliminate many more flaws. Unit tests determine whether a given feature works as intended. A programmer writes as many automated tests as they can think of that might "break" the code; if all tests run successfully, then the coding is complete. Every piece of code that is written is tested before moving on to the next feature.

Acceptance tests verify that the requirements as understood by the programmers satisfy the customer's actual requirements. These occur in the exploration phase of release planning.

(c) Listening

Programmers must listen to what the customers need the system to do, what "business logic" is needed. They must understand these needs well enough to give the customer feedback about the technical aspects of how the problem might be solved, or cannot be solved. Communication between the customer and programmer is further addressed in the Planning Game.

(d) Designing

From the point of view of simplicity, of course one could say that system development doesn't need more than coding, testing and listening. If those activities are performed well, the result should always be a system that works. In practice, this will not work. One can come a long way without designing but at a given time one will get stuck. The system becomes too complex and the dependencies within the system cease to be clear. One can avoid this by creating a design structure that organizes the logic in the system. Good design will avoid lots of dependencies within a system; this means that changing one part of the system will not affect other parts of the system.

2. Scrum

Scrum is an iterative, incremental framework for project management often seen in agile software development, a type of software engineering. Scrum is a process skeleton that contains sets of practices and predefined roles. The main roles in Scrum are as follows:

- The Scrum Master, who maintains the processes (typically in role of a project manager)
- The Product Owner, who represents the stakeholders and the business
- The Team a cross-functional group of about 7 people who do the actual analysis, design, implementation, testing, etc.

© 2014, IJARCSSE All Rights Reserved

Scrum has not only reinforced the interest in software project management, but also challenged the conventional ideas about such management. Scrum focuses on project management institutions where it is difficult to plan ahead with mechanisms for empirical process control, such as where feedback loops constitute the core element of product development compared to traditional command-and-control-oriented management. It represents a radically new approach for planning and managing software projects, bringing decision-making authority to the level of operation properties and certainties. Scrum reduces defects and makes the development process more efficient, as well as reducing long-term maintenance costs.

3. Dynamic development system method

Dynamic Systems Development Method (DSDM) is primarily a software development methodology originally based upon the Rapid Application Development methodology. In 2007 DSDM became a generic approach to project management and solution delivery. DSDM is an iterative and incremental approach that emphasizes continuous user/customer involvement. There are nine underlying principles consisting of four foundations and five starting-points.

- User involvement is the main key in running an efficient and effective project, where both users and developers share a workplace, so that the decisions can be made accurately.
- The project team must be empowered to make decisions that are important to the progress of the project without waiting for higher-level approval.
- A focus on frequent delivery of products, with assumption that to deliver something "good enough" earlier is always better than to deliver everything "perfectly" in the end. By delivering product frequently from an early stage of the project, the product can be tested and reviewed where the test record and review document can be taken into account at the next iteration or phase.
- The main criterion for acceptance of a “deliverable” is delivering a system that addresses the current business needs. Delivering a perfect system which addresses all possible business needs is less important than focusing on critical functionalities.
- Development is iterative and incremental and driven by users’ feedback to converge on an effective business solution.
- All changes during the development are reversible.
- The high level scope and requirements should be base-lined before the project start.
- Testing is carried out throughout the project life-cycle.
- Communication and cooperation among all project stakeholders is required to be efficient and effective

Like XP, there are many other development methods that show similarities to DSDM, but DSDM does distinguish itself from these methods in a number of ways. First there is the fact that it provides a tool and technique independent framework. This allows users to fill in the specific steps of the process with their own techniques and software aids of choice. Another unique feature is the fact that the variables in the development are not time/resources, but the requirements. This approach ensures the main goals of DSDM, namely to stay within the deadline and the budget. And last there is the strong focus on communication between and the involvement of all the stakeholders in the system. Although this is addressed in other methods, DSDM strongly believes in commitment to the project to ensure a successful outcome.

B. Agile testing

Agile testing is a software testing practice that follows the principles of agile software development. Agile testing is executed in a very different way from the traditional norm. It basically consists of a continuous process throughout the software development life cycle as the testing provides information about the new software’s performance as well as the functionality. As a result there is no time for detailed testing of the software at a later stage let alone doing the necessary corrections, since it is an iterative process. Therefore the first sprint is developed and tested. Then the second is developed and tested in isolation, before being integration tested with the first sprint. This goes on and on till the software has been completely developed [4].

Agile testing does not emphasize testing procedures and focuses on ongoing testing against newly developed code until quality software from an end customer's perspective results. Agile testing is built upon the philosophy that testers need to adapt to rapid deployment cycles and changes in testing patterns. Test case would have detail steps of what the application is supposed to do Functionality of application and In addition you can refer to Backend, is mean look into the Database.

Agile testing involves testing from the customer perspective as early as possible, testing early and often as code becomes available and stable enough, since working increments of the software are released often in agile software development. This is commonly done by using automated acceptance testing to minimize the amount of manual labor involved. The nine principles of agile testing are as follows [5]:

- Testing Moves the Project Forward
- Continuous testing is the only way to ensure continuous progress.
- The independent testers are responsible for all test activities.
- Agile projects the software is ready to test almost from the beginning.
- Test represents expectations.
- Keep the Code Clean.
- Lightweight Documentation
- Instead of writing verbose, comprehensive test documentation.

© 2014, IJARCSSE All Rights Reserved

C. ***Phases of agile testing:*** The Phases of Agile Testing are as follows:

1) Unit Testing: The developers do the unit testing of their code. Think of this as being an initial test to see whether the code works at a high level. These tests should be automated as far as possible to reduce the testing time.

2) Integration Testing: Integration testing and system testing start together. As soon as the second sprint is over, the system and integration testing of that sprint, together with the other ones start.

3) Regression Testing: After each period of integration testing there is a short cycle of regression testing. As the development is automated test driven, the system should be working well at this stage and hence there would be no need for a long period of regression testing.

Agile testers use reusable checklists to suggest tests. While script writing must be done in advance this can often be difficult to do, particularly in cases where the level of system documentation is poor. It is thus essential to understand what is appropriate to go into the script. The challenge is defining scripts that meet the following objectives.

Keep agile: Often testing out of proportion of what needs to be tested

- **Ensure complete coverage**: It is important that there are no significant holes in the testing
- **Develop in cooperation of IT**: Develop scripts in close cooperation with the developers – testers often feel the preparation activity must be done in a vacuum as they fear that involvement of other parties may contaminate the impartiality – this is a mistake – typically time is very constrained at this point of the project so any ways of right sizing are of benefit.
- **Ensure separation of testers to IT**: The area where you want to keep clear separation in terms of responsibility is the execution of the tests (rather than the formulating of test scripts). If the scripts are clear and concise enough to allow a 3rd party without any domain or functional experience to run, then you have succeeded in pitching the scripts at the right level.
- **Define the evidence basis**: How is success verified? SOX mandates that testing evidence is to be recorded in certain cases.
- **Ensure the tests are right for what is being tested**: Unless the application is very large, or has very complex STP/ algorithms the following areas should are really all that should be considered for the typical web-based application:
- **Quality assurance**: Given that the major part the development is user-facing, quality assurance suite of scripts are to confirm that the user experience is operating in a consistent way (for example ensuring that basic standards in UI design are adhered to such as formatting, accessibility, error-handling, alignment and spelling). Support for different browsers and operating systems is also to be tested. The functionality of security (availability to functions for those with/without a valid account/password) is also covered. The format of these scripts is a checklist against all screens.
- **End-to-end**: The majority of screens form part of an end-to-end enterprise, so it is important to demonstrate that the users can navigate through the screens to achieve their end-goal, without any logical breaks or unexpected behaviour. It is also important to demonstrate that data input in an earlier stage is carried forward in a consistent manner. User-experience suite of tests therefore briefly describes scenarios involving a number of screens to ensure all stages and the end result occur as expected.

4) Boundary testing: Testing of the behaviour of both inputs and outputs is shown in the boundary-test suite of scripts. With the inputs this works by examining that values of each control in isolation and stressing them with boundary conditions or inconsistent values to their type, for outputs the boundary values of what can be accepted by the receiving component are examined, and tests are contrived to see if these can be exceeded (i.e. it drives us to ensure that test coverage is complete, but we are not actually testing the interface directly).

- **Interface testing**: This is a technical test. Checks to ensure input/output to interface are as expected, and that unavailability of interface does not cause unexpected/unsupported application behaviour in both synchronous and asynchronous modes
- **Load/SLA testing**: For purposes of this agile testing these can be often be combined. Load testing will attempt to see the upper number of users the application can support.

III. AGILE TESTING LIFE CYCLE

All the software development life cycles either follow a sequential model (eg waterfall model) or an iterative model .But for processes which are a little more complex these models find it difficult to tackle the changes which are large in number and continuous.

Agile model was basically made in order to counter these changes effectively and smoothly. This method is a collection of values, principles and practices that test and provide the feedback to convert it into a new style of development. The steps followed in agile testing life cycle are as follows [7][9]:

- **Iteration -1** :Identify, prioritize the potential project, and consider its feasibility
- **Iteration 0 (Warm Up)** : Initiates the project, takes care of the initial inputs , and emphasis is on building a unit
- **Construction Iterations**: Client participates actively and advises about the expected output, requirements and stuff which is not required by the project
- **Release:** Deploys first release into production , Final system testing and trains end users
- **Production**: Operates, supports and identifies defects in the system
- **Retirement**: Removes the system completely from production and migrates the users.

© 2014, IJARCSSE All Rights Reserved

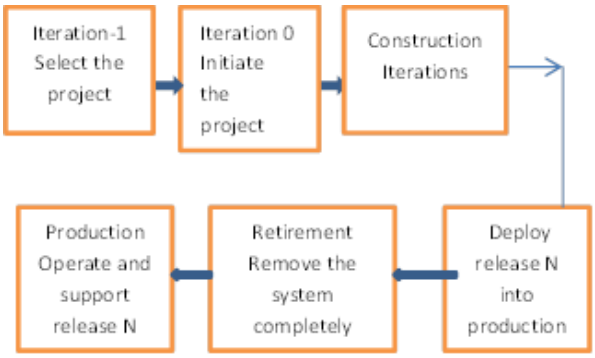

Figure 1.1 Agile Development Life Cycle

A. Challenges in agile testing

In Agile testing, there are most difficult challenges, which a tester can face:

- Due to the tight timelines, there is hardly any time for detailed test planning.
- There are too many changes in the software requirements and development which conflict with the test planning
- The testing phase is not specified. That is, it is not clear when tests start and end.
- Since testing starts when the first iteration is complete, judging at that stage whether it will lead to working software is difficult.

In agile testing, a tester is a utility person, whose skills are used throughout the process of software development.

B. Agile benefits

Agile methods grew out of the real-life project experiences of leading software professionals who had experienced the challenges and limitations of traditional waterfall development on project after project. The approach promoted by agile development is in direct response to the issue associated with traditional software development – both in terms of overall philosophy as well as specific processes.

Agile development, in its simplest form, offers a lightweight framework for helping teams, given a constantly evolving functional and technical landscape, maintain a focus on the rapid delivery of business value (i.e., "bang for the buck"). As a result of this focus and its associated benefits, organizations are capable of significantly reducing the overall risk associated with software development. .In particular, agile development accelerates the delivery of initial business value, and through a process of continuous planning and feedback, is able to ensure that value is continuing to be maximized throughout the development process. As a result of this iterative planning and feedback loop, teams are able to continuously align the delivered software with desired business needs, easily adapting to changing requirements throughout the process. By measuring and evaluating status based on the undeniable truth of working, testing software, much more accurate visibility into the actual progress of projects is available. Finally, as a result of following an agile process, at the conclusion of a project is a software system that much better addresses the business and customer needs.

IV. AGILE SURVEY

We present a survey usage and popularity of agile methods in current industry practices. Results have been drawn from recent surveys.

Forrester Survey: In a survey report, Forrester surveyed 229 technology industry professionals in various roles including quality assurance (QA), product management, sales, support, consulting and marketing to determine the effects of agile methods. They also surveyed 24 companies including ACS, IBM, Borland Software, Protegra, Sybase and others, to supplement the results. Results the various questions were analyzed as:

The first question answers the level of completeness of agile implementation in the companies, 35% of the companies have mature implementation of agile methods, and 33% are in middle of their implementation, and 17% have started implementing agile methods [3]. This shows that over 85% of companies have switched over to agile methods. The second question was to interrogate whether agile implementation has changed the frequency of product releases. Implementation and synchronization to bring new vision, [3] 67% of respondents responded increase in frequency of releases. Shorter cycles and frequent releases make the managers to analyze more accurately when the product will be finished. Third question was to know that whether agile implementation has improved companywide understanding of target customers, markets and competitors. 15% respondents declared significant improvement, 34% said that the above factors were moderately improved, 40% said no change, 2% said no change, and other 7% shown inability to report any changes. The improvements are significant. Fourth question was to analyze whether agile implementation improves companywide implementation and synchronization to bring new product and vision to market. Over 57% of respondents reported improvement, 27% reported no change, and 8% said things got worse, other 5% showed inability to report [8].

V. CONCLUSION

In this paper we described various agile methods and showed their increasing role and popularity in software industry. We also established the relevance of Agile methods in current industry practices. Results clearly show popularity of Scrum method for agile development. As resulted from surveys and Intel experience, we can state that Agile methods clearly provide improvements in software development process. Agile methods require change in mindset and thinking of

© 2014, IJARCSSE All Rights Reserved

stakeholders. Though these methods have shown encouraging results and software industry is at adopting these methods at fast pace, yet there are certain challenges for agile methods such as ensuring CMMI standards. More research needs to be done in this area. We have also described agile testing is a software testing practice that follows the principles of agile software development. Agile testing is executed in a very different way from the traditional norm. With the help of Agile testing life cycle, we have shown the various steps performed in making projects based on Agile.

REFERENCES

[1] Jonathan Foote, An Overview of Audio Information Retrieval, ACM Multimedia Systems, Vol.7, 1999, pp. 2-10.
[2] Rabiner and B.H. Juang, Fundamentals of speech recognition, Prentice Hall, Upper Saddle River, New Jersey 07458,1993.
[3] G.Tzanetakis, P.Cook , A framework for audio analysis, Organised sound,Vol.4(3), 2000.
[4] http://www.onestoptesting.com/agile-testing/
[5] http://www.onestoptesting.com/agile-testing/principle-nine.asp
[6] http://msdn.microsoft.com/en-us/library/ff649520.aspx
[7] http://www.softwaretestingnet.com/2011/01/
[8] http://www.versionone.com/Agile101
[9] http://www.ambysoft.com/essays/agileLifecycle.html#Development
[10] http://www.ambysoft.com/essays/agileLifecycle.html#Release
[11] http://agilemanifesto.org/
[12] http://agilemanifesto.org/principles.html
[13] http://www.agile-process.org/
[14] http://en.wikipedia.org/wiki/Agile_software_development
[15] http://www.agilealliance.org/the-alliance/what-is-agile/
[16] http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm
[17] http://agiledata.org/
[18] http://www.indicthreads.com
[19] www.globalknowledge.com

© 2014, IJARCSSE All Rights Reserved