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

Smarter Agile Deliveries

Swati Chawla*
M.TECH, CSE Student(Amity University)
Noida, India

Sanjeev Thakur
Professor, CSE, Amity University
Noida, India

Abstract- For the people related to software development industry, it was a continuous challenge to select the appropriate software development methodology. Since the early 1990s, the general trend in software development technology was to change from the plan-driven approach to more iterative incremental development approaches. This change has led to the birth of a group of methodologies called "agile methodologies". The values and principles are becoming more prevalent in the software development industry. At the later stage, since 2004, the focus of selecting specific agile methodology shifted to selection of the most appropriate practices from the agile family. The latest agile methodologies have undoubtedly fastened the pace of development of software applications. But the current deliveries and deployments are still quite old fashioned and slow paced. There is a critical need of understanding the business impact and market value of a deliverable and to be able to make a smarter and agile delivery. This paper, contribute towards greater understanding of software development issues related to making smarter deliveries and would be useful to Agile projects who want to keep pace with the fast pace Agile development and make their software deliveries and deployments also smarter and Agile.

Keywords: Agile development, Continuous deliveries, Continuous integrations, Deliveries, Deployments

I. SDLC METHODOLOGIES

Software development is term referring to the development of a software product. Software development is the activity of computer programming, which is the process of writing and maintaining the source code, but in a broader sense of the term it includes all that is involved between the conception of the desired software through to the final manifestation of the software, ideally in a planned and structured process. Therefore, it may include research, new development, prototyping, modification, reuse, re-engineering, maintenance, or any other activities that result in software products and solutions.

A computer software application can be developed for a variety of reasons, the three most common being to meet specific needs of a specific business or a client, to meet a perceived need of some set of potential users or open market base, or for personal use. The need for better quality control of the development process has given rise to may researches in the discipline of software engineering, which aims to apply the approaches exemplified in the engineering world to the process of software application development.

There are several different ways of software development. Some are a more structured, engineering-facts based approach of developing robust business solutions for the electronic world, whereas others may take a more incremental and iterative approach, where software evolves as it is developed in small parts. Most software development methodologies share some combination of the following stages of software application development:

- Analysing the problem
- Market research
- Gathering requirements for the proposed business solution
- Devising a plan or design for the software-based solution
- Coding of the software
- Testing the software
- Deployment
- Maintenance and bug fixing

These stages are often referred to collectively as the software development lifecycle, or SDLC. Different approaches to software application development may have these stages in different combinations, or devote more or less time to individual stages as compared to any other methodology of development[4]. The level of detail of the documentation produced at each stage may also vary. These stages may also be carried out in specific position, or they may be repeated over various cycles of development or frequently termed as software delivery iterations. The more extreme approach usually involves less time spent on planning and documentation, and more time spent on coding and development of automated tests. More "extreme" approaches also promote continuous testing throughout the development lifecycle, as well as having a working product at all times. More structured development approaches attempt to assess the majority of risks and develop a detailed plan for the software before implementation begins, and avoid significant design changes and re-coding in later stages of the software development life cycle planning[3].

There are pros and cons to the various methodologies, and the best approach to solving a problem using software will often depend on the type of problem. If the problem is well understood and a solution can be effectively planned out ahead of time, the more "waterfall" based approach may work the best. If, on the other hand, the problem is unique and the structure of the software solution cannot be easily envisioned, then a more "extreme" incremental approach may work best[1]. A software development process is a structure imposed on the development of a software product. Synonyms include software life cycle and software process. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process.

Agile development methodologies uses iterative development as a basis but advocates a lighter and more people-centric viewpoint than traditional approaches. Agile processes fundamentally incorporate iteration and the continuous feedback that it provides to successively refine and deliver a software system. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. Agile methods grew out of the real-life project experiences of leading software experts 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. 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[2]. 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. By delivering working, tested, deployable software on an incremental basis, agile development delivers increased value, visibility, and adaptability much earlier in the life cycle, significantly reducing project risk.

Software deployment is all of the activities that make a software system available for use. The general deployment process consists of several interrelated activities with possible transitions between them. These activities can occur at the producer side or at the consumer side or both. Because every software system is unique, the precise processes or procedures within each activity can hardly be defined. Therefore, "deployment" should be interpreted as a general process that has to be customized according to specific requirements or characteristics.

There are many ways to deliver a software to the business. With evolution to the development tasks into agile the delivery mode and deployments have also gone automated and quicker. The latest in the trend being Continuous Integration and Continuous Delivery.

Continuous integration (CI) is the practice, in software engineering, of merging all developer working copies with a shared mainline several times a day. CI was originally intended to be used in combination with automated unit tests written through the practices of test-driven development. Initially this was conceived of as running all unit tests and verifying they all passed before committing to the mainline. This helps avoid one developer's work in progress breaking another developer's copy. If necessary, partially complete features can be disabled before committing using feature toggles[1].

Continuous Delivery (CD) is a design practice used in software development to automate and improve the process of software delivery. Techniques such as automated testing, continuous integration and continuous deployment allow software to be developed to a high standard and easily packaged and deployed to test environments, resulting in the ability to rapidly, reliably and repeatedly push out enhancements and bug fixes to customers at low risk and with minimal manual overhead. Continuous ways of delivering are interring and can be simulated to make deliveries every sprint(2-4 weeks), every Friday, each day, or even every other minute.

II. CONCLUSIONS

We want things to be automated to support the fast pace of Agile development and changing business needs to save our development facing a delivery bottleneck. We want them to be customisable. But we do not want them to be continuous. Because continuous is not Agile.

Agile has a crisp meaning which is changing as per the need. The frequency with which the business changes is much faster than the development processes. A feedback received changes the face of an application in development completely. There is no certainty that the market stays the same till the point the product hits the business.

To summarise, Agile is a creative and innovative solution to most of our development needs to keep things upfront and in sync with the ever changing business. But the delivery model or deployments are not.

Through this paper, I would like to present a new model for "Agile and Smart Deliveries and Deployments" which will be smart enough to cater the needs of the current scenario and be agile enough to solve support the fast pace of Agile development.

The above stated model is as follows:

1. Each new user story is added to the product backlog with the status as "To Be Prioritised".
2. Once the Business associates a story with its business values or estimated impact factor, we would mark that story as "Business Prioritised"
3. Starting from the stories which have the highest business value or impact factors, the business analyst starts analysing the stories for their exact functional details, this phase is termed as "Functional Analysis".
4. Post Functional analysis of any story it is submitted for the technical architects for feasibility analysis and design, this phase is called "Technical Design".
5. Based upon the detailed functional and technical instructions furnished by the previous steps, one or more developers starts working on the story, this phase being termed as "Development Phase". After completion of which the stories are submitted for "To Be Tested" phase.
6. Following the development phase the testing competency starts picking up the prioritised stories from the backlog, which are in "To Be Tested" state. The stories in this stage of development would be marked as "Under Evaluation".
7. Once the story is Tested, the state of the story is marked as "Ready for Delivery", and poked back to the business stakes for review of business priorities and impact values.
8. The business then evaluates the new developed feature by several new mechanisms like A/B random tests, automation and regression evaluations, and market value and impact factor of the feature. Under this phase the business realises the actual business value a new developed feature has, its estimated impact factor, whether or not they what this development in their shippable product, and how soon the business wants this new feature to hit the market.
9. Having answered the questions, the business takes the decision to mark the feature as "To Be Delivered" or "Parked For Delivery".
10. The system then automatically finds all the "To Be Delivered" items and make a quick release and delivery for the old system plus the new feature set. The new features added in this delivery are now termed as "Delivered".
11. The featured which are "Parked For Delivery" may be discarded or may be added later in the future deliveries, based upon the business requirements.

III. ACKNOWLEDGMENT

I take this opportunity to express my profound gratitude and deep regards to my guide Prof. Sanjeev Thakur for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life.

I would also like to thank Prof. (Dr.) Ajay Rana, Program Director - Amity School of Engineering and Technology, and all the faculty members for their cooperation during the period of my assignment and their selfless help for the successful completion of project.

I also take this opportunity to express a deep sense of gratitude to my Mentor and Colleague Mr. Anup Sharma, Technical Lead, Sopra Group, for his/her cordial support, valuable information, which helped me in completing this task through various stages of project development and design.

Lastly, I thank almighty, my parents, and friends for their constant encouragement without which this assignment would not be possible.

REFERENCES

[1] Beck, Kent; et al. (2001). "Manifesto for Agile Software Development". Agile Alliance. Retrieved 14 June 2010.
[2] Ambler, S.W. (2002a), Introduction to Agile Model Driven Development, Agile Modeling Essays, *Ronin International Inc.* [online]. Available from: http://www.agilemodeling.com/essays.
[3] Mikael Lindvall, Vic Basili, Barry Boehm, Patricia Costa, Kathleen Dangle, Forrest Shull1, Roseanne Tesoriero, Laurie Williams, and Marvin Zelkowitz "Empirical Findings in Agile Methods"
[4] Cockburn, 2000; Ambler, 2004; Ambler, Nalbone, and Vizdos, 2005 and Glass and Vessey, 1995).
[5] Bradley, P. (2003), *Agile as Evolution and Exploration*, FDD Website, [online]. Available from: http://www.featuredrivendevelopment.com/node/view/592, [last accessed 10 June 2006].
[6] Beck, K. (1999), Extreme Programming Explained: Embrace Change, Addison–Wesley, Reading, Mass, pp. 10–70.

