A Comparative Analysis of Agile methodology

Parteek
Assistant Programmer, National Institute of Electronics and Information Technology (NIELIT)

Received: November 20, 2018 Accepted: December 29, 2018

ABSTRACT *In this paper we have discussed the Agile Methodology and Experience with the Agile Methodology in Model-driven Approach. Agile is an important area in the software management. Main aim of Agile is to satisfy the Customer with low defects but it become crucial that code development should not become bottleneck for the project. So, we have reviewed two papers in this review paper.*

Keywords:

Introduction

In software development life cycle, we mainly consider to emphasize on process and the software quality. Agile model is an incremental and iterative based development, where we can change the requirements as per customer needs. It helps us in time boxing, planning, and iterative development. Agile framework promotes foreseen interactions in whole development cycle. Every SDLC has their own advantages. Software development life cycle is framework that shows the activities performed at every stage of a software development process.

Model-driven approach can help us for separate technology platform from functional concerns and enable the developers to focus on business specification functionality in a better way that is closer to the domain problem. This leads us to look into the process of model-driven development. Agile methodology is gaining acceptance from industry. Here in 2nd paper author argue that this method can not be used with model-driven approach. They have suggested modifications for delivering a better specific system.

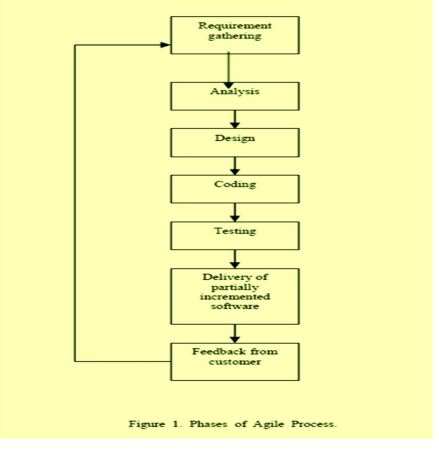

Figure 1. Phases of Agile Process.

Here we have reviewed two papers

Paper 1: Agile Processes and Methodologies: A Conceptual Study

Paper 2: Early Experience with Agile Methodology in a Model-driven Approach

Review of Paper 1 :

In this paper author discussed the various software development life cycle models, the characteristics of agile process, and spiral model, methodologies of agile process, advantages and disadvantages. In the comparative study of agile software development with other software development models we conclude that agile project is much better than other software development process in terms of productivity, performance, faster time cycles, risk analysis. Agile processes are implemented in important applications such as web based, testing tools, etc.

Firtstly,we will discuss its characteristics which are mentioned by the Author in the paper. These are as:

Iterative: Each day, the agile development team is planning, working on, and completing tasks while the software is being designed, coded, tested, and integrated for customer acceptance.

CHARACTERISTICS OF AGILE PROJECTS

1. Time Boxing

As agile process is iterative in nature, it requires the time limits on each module with respective cycle.

2. Iterative

The main objective of agile software processes is satisfaction of customers, so it focuses on single requirement with multiple iterations.

3. People Oriented

In the agile processes customer satisfaction is the first priority over the technology and process. A good software development team increases the performance and productivity of the software.

4. Parsimony
In agile processes parsimony is required to mitigate risks and achieve the goals by minimal number of modules.

5. Adaptive
Due to the iterative nature of agile process new risks may occurs. The adaptive characteristic of agile process allows adapting the processes to attack the new risks and allows changes in the real time requirements.

6. Incremental
As the agile process is iterative in nature, it requires the system to be developed in increments, each increment is independent of others, and at last all increments are integrated into complete system.

7. Convergent
All the risks associated with each increment are convergent in agile process by using iterative and incremental approach.

8. Collaborative
As agile process is modular in nature, it needs a good communication among software development team. Different modules need to be integrated at the end of the software development process.

9. Modularity
Agile process decomposes the complete system into manageable pieces called modules. Modularity plays a major role in software development processes.

METHODOLOGIES

A. Extreme Programming (XP)
XP is the most successful method of developing agile software because of its focus on customer satisfaction. XP requires maximum customer interaction to develop the software. It divides the entire software development life cycle into several number of short development cycles. It welcomes and incorporates changes or requirements from the customers at any phase of the development life cycle.

B. Scrum
Scrum is another popular method of agile development through which productivity becomes very high. It is basically based on incremental software development process. In scrum method the entire development cycle is divided into a series of iteration where each iteration is called as a sprint. Maximum duration of a sprint is 30 days.

Paper 2. We are in the business of delivering software intensive business systems using model-driven techniques. Since developing suitable code generators is an important step in model-based development of purpose-specific business applications, it becomes critical to ensure that code generator development doesn't become a bottleneck for the project delivery. After having put in place sophisticated technology infrastructure in place to facilitate quick and easy adaptation of model-based code generators, we experimented with agile methodology. We discussed why pure Agile methodology does not work for model-driven software development. We proposed modification to the Agile method in the form of meta-sprints as a golden mean between Agile method and traditional plan-driven method. Unlike Agile method, meta-sprint deliverables are not working software but could be a design document, a proof-of-concept implementation, evaluation of a set of design strategies etc. Unlike traditional method, meta-sprint puts an upper bound in terms of time (and hence effort) on exploratory activities. In addition, meta-sprint deliverables facilitate subsequent normal sprints. Thus development proceeds on two parallel threads namely meta-sprint and sprint with periodic synchronization between the two threads.

Proposed Methodology
With the principal objectives of delivering functionality that brings value to the customers, establishing better mechanisms for feedback from all stakeholders, and enabling quick transformation of an idea/requirement into a set of MasterCraft features, we found Agile Manifesto [18] as the best bet for many reasons. Existing approach heavily depended on documentation for communication between the phases that, we felt, could be eliminated to some extent by having more and closer interactions of customer with tool and solution builders – it was always a demand to show some working software than say, a usecase diagram or a design document. In existing approach, one could get to see a working version after a significant time has elapsed after the requirements were communicated. Thus, it was hard to establish quick-wins with the existing approach. Moreover, most certainly the requirements would have undergone a change thus necessitating rework. Agile method puts greater stress on close collaboration with the customer as opposed to a contract. As advised in Agile methodology, we felt that responding to a change requisitioned by the customer should take precedence over following a plan. However, we found some limitations of using Agile methodology for all kinds of MasterCraft development activities. Agile methods haven't been as useful for large development teams comprising of members having wide variance in

expertise levels and operating in geographically distributed manner [10]. In addition, some characteristics of MasterCraft created hindrances for applying Agile methodology uniformly. We observed that mutation and exploration kind of changes are not suitable for Agile method. Catering to mutative changes demands in-depth analysis, experimentation and detailed documentation of the results, observations and conclusions. These activities are difficult to achieve in the short sprint cycles advocated by Agile method. Exploratory work demands in-depth study and analysis which is hard to synchronize with sprint timelines. Agile method advocates to keep customer aware of all decisions, however, that may create unnecessary pressure during exploratory stage and also entail significant product testing and quality assurance effort at the exploratory stage itself. Agile method puts greater stress on working software as opposed to documentation, however, low / no documentation of design rationale and far reaching changes may create problems for hassle-free maintenance and smooth induction. Customer visibility into sprint-backlogs and planning means there is little opportunity for long term research activities. Moreover, defining sprint-backlogs based purely on customer requirements is not always possible – as MasterCraft is a set of interrelated tools, sometimes the order of scoping a feature in a sprint-backlog depends on internal factors rather than purely customer needs.

Conclusion

With the help of proposed development method, we observed slow but steady improvement in features on time. In first iteration we delivered only 50% result. But results improved in following iterations, and we achieve our target on time within sprints. Early results of using Agile methodology are encouraging with a note that it is not applicable for all kinds of development activities and needs considerable preparedness for deployment in practice. We adapted true agile methodology by introducing meta-sprint concept for mutative and exploratory work; and used this methodology only after plug-in architecture and suitable tools were in place. Though our objective is to channelize more development activity through sprint stream than meta-sprint stream and establish an agile development environment, we would like to continue with a relatively relaxed environment (meta-sprint) for exploratory and knowledge intensive activities. Our early results show that this kind of hybrid development environment is better suited for model-driven software development.

References

1. Ambler, S.W.: Agile Model Driven Development (AMDD). http://www.agilemodeling.com/essays/ mdd.htm, 2007
2. Barat, S., Kulkarni, V.: Developing configurable extensible code generators for model-driven development approach. SEKE 2010: 577-582
3. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley, MA (2000).
4. Boehm, B., Turner, R.. Observations on Balancing Discipline and Agility, Proceedings of the Agile Development Conference ADC'03, IEEE Computer Society. 2003.
5. Cao, L., Mohan, K., Xu, P., Ramesh, B.: A framework for adapting agile development methodologies. EJIS 18(4): 332-343 (2009)
6. Cawley, O., Wang, X., Richardson, I.: Lean/Agile Software Development Methodologies in Regulated Environments – State of the Art. In Proceedings of Lean Enterprise Software and Systems, p. 31-36. Helsinki, Finland, October 2010: Springer Berlin Heidelberg.
7. Cockburn, A.: Agile Software Development. Addison-Wesley Professional December 2001, ISBN 0-201-69969-9.
8. Tobin J Lehman, Akhilesh Sharma , "Software Development as a service: Agile Experiences", in annual SRII Global
1. Conference (2011).
9. A. Ahmed, S. Ahmad, Dr. N Ehsan, E. Mirza, S.Z. Sarwar,"Agile Software Development: Impact on Productivity and Qulaity", in the Proceedings of IEEE ICMIT.(2010).
10. B.Boehm and R.Turner, "Balancing Agility and Discipline: A Guide for the Perplexed, Addison, Wesley, 2003.
11. Jeffery A. Livermore, "Factors that Impact Implementing an Agile Software Development Methodology" in the Proceedings of IEEE (2007)
12. .http://en.wikipedia.org/wiki/Scrum_(development)
13. B.Boehm, " Anchoring the Software Process," IEEE Software,