INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 6.078
(Volume 6, Issue 1)
Available online at: www.ijariit.com

Adaptation and implementation of agile project management methodology in software development

Paramita Dey
paramita.dey2014@gmail.com

Agile methodologies are widely used in the software industry and every day more and more companies are joining the agile board. Present-day software projects are very complex and often are being met with vulnerabilities and unconventionality, at the same time the IT industry is always striving to attain excellence in terms of delivery speed and quality. Agile, the light-footed method attempts to respond to such challenges through gradual, iterative rhythms of job and experimental inputs. The Agile methodology is a critical system developed in the early 1990s to address the vulnerability problems identified with evolving customer needs and innovation development. Agile methodology has changed the paradigm of traditional project delivery through its business-value based frequent delivery of working software in more frequent releases through continuous collaboration with customers. It has not only reduced the probability of project failure for customers but also boosted the motivation and productivity of the development team by creating an environment of transparency, cooperation, and technical excellence.

The development team is the backbone of any project and agile principles give much value in this regard. Agile respects a constant working pace which has been agreed by stakeholders including the development team. During sprint development, the pace of progress at the early stage may be slow, which necessitates the sprint-0 before the development sprint. The sprint-0 should take care of the basic setup necessary to initiate the first sprint. Every sprint follows agile connotations like Sprint Planning, Daily Scrum, Sprint Review and Retrospection and has a releasable working software as an outcome. Some of the activities in sprint-0 could be story grooming and setting priority in terms of functionality, infrastructure setup, architectural envisioning and last but not the least necessary pieces of training.

Awareness about agile methodology and acceptance of the ground rules by all team members is crucial to avoid friction during sprint development which would be detrimental in attaining sprint goals. Out of five stages of team formation defined in PMP® viz forming, storming, norming, performing and adjourning, the first two stages must be completed before the first sprint for smooth execution.

Agile advocates a self-organized team, meaning a team that has the maturity to self-organize thus limiting frequent management intervention. Richard Hackman, a Harvard University professor defined a self-managed team as a team that will not only perform the tasks but will also manage its own process. Such teams are given more authority than a traditionally managed team, whose authority remains within the allocated tasks only. Such self-organized teams should have the liberty of selecting a process that suits their purpose and maximize efficiency. Thus, it is the team's discretion whether to select Agile methodology or adopt other software development processes. But once decided, the team agrees on the process and sprint goals and adheres to it. The team also reviews the pitfalls of the process in retrospection meetings and constantly tailors the process to attain maximum efficiency. In a way, Agile methodology promotes more authority to team members thereby increasing overall team-ownership for both failure or success of the project.

In 1959, Frederick Herzberg, a behavioral analyst proposed the Motivator-Hygiene theory. He stressed a lot about factors that motivate an individual for a high standard of performance. According to him, empowerment, meaningful work which would challenge individuals and recognition are some of the factors which add to motivation. The agile methodology encourages these motivational factors and brings forward the best out of an individual which is a crucial element to the success of any project.

In 1967, Melvin Conway in his publication titled "How do committees invent?" stated "Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations". This sociological statement advocates the benefits of leveraging small and independent teams against teams aligned to an organization's rigid structure. For the successful implementation of Agile, the overall organization structure should go through a realignment process. Apart from the development team, there are other supporting teams, customers and management who to need to be aware of Agile methodology

so that they know their clearly defined roles and responsibilities and boundaries so that the development team can work with full potential.

Early-stage testing is a vital differential factor for Agile from traditional process models like Waterfall. Sprints being time-boxed along with the involvement of customers in early testing, it is imperative to adopt automated testing. Manual testing for whole software will not operate in a coordinated fashion rather it would lead to surrey programming in dexterous testing.

One crucial KPI is sprint velocity which is a pointer to a team's productivity for a particular sprint. A good velocity would be one that shows improvement or higher values over a period of time in subsequent sprints. In the first few sprints, the velocity may not be as high as anticipated, as the team might take some time to get familiar with project architecture and the environment. Agile, in its essence is designed to fail fast (as failing later in the project is expensive). Hence any shortcomings in terms of design, architecture and business value would surface in the initial sprints for which velocity may below. However, over a period of time, the velocity should show an upward trend. If it does not happen, then such a situation might qualify for an RCA. There could be many reasons for low velocity. Firstly, a stringent focus on quality like code reviews, unit testing, code quality check, etc may be effort-intensive and impact velocity. Secondly, if the build software is complex for any reason, one being highly coupled to many other modules, then testing might take more time affecting the velocity. Thirdly, if interdependencies and impediments were not resolved on time then associated waiting time will have a negative hit on velocity. Fourthly, team members not having the necessary skills or attrition among team members will adversely impact velocity.

In today's fast-paced software industry operating in an ambiguous and dynamic environment, a light-footed process like Agile embedded in a group of highly motivated and competitive team has a very high probability of success.

