AGILE APPROACH -DIGITAL TRANSFORMATION

Priyanka Konde and Prof. Rashmi Tundalwar

Dept of Computer Engineering, Dhole Patil College of Engineering, Pune, India

Abstract:

Agile software development (ASD) is major paradigm, in field of software engineering which has been widely adopted by the industry, and much research, publications have conducted on agile development methodologies over the past decade. The traditional way to develop software methodologies follow the generic engineering paradigm of requirements, design, build, and maintain. These methodologies are also called waterfall–based taking from the classical software development paradigm. They are also known by many other names like plan–driven, (Boehm and Turner, 2004), documentation driven, heavyweight methodologies, and big design upfront, (Boehm, 2002), [16]. Boehm and Phillip report that during their project development experience, requirements often changed by 25% or more. Due to constant changes in the technology and business environments, it is a challenge for TSDMs to create a complete set of requirements up front. Williams and Cockburn, [18] also mentioned that one of problems of TSDMs is the inability to respond to change that often determines the success or failure of a software product.

Keywords: Agile, Digitization, Jenkins, Software Development.

Introduction:

The agile approach to software development is based on the understanding that software requirements are dynamic, where they are driven by market forces .2002; Cockburn & Highsmith, 2001); [16]. Agile systems development methods emerged as a response to the inability of previous plan-driven approaches to handle rapidly changing environments (Highsmith 2002). Williams and Cockburn [18] state that agile development is about feedback and change‖, that agile methodologies are developed to embrace, rather than reject, higher rates of changel.

Agility is the ability to sense and response to business prospects in order to stay inventive and aggressive in an unstable and rapidly shifting business environment (Highsmith, 2002). The agile approach to development is about agility of the development process, development teams and their environment (Boehm & Turner, 2004). This approach incorporates shared ideals of various stakeholders, and a philosophy of regular providing the customers with product features in short time-frames (Southwell, 2002). This frequent and regular feature delivery is achieved by team based approach (Coram & Bohner, 2005). Agile teams consist of multi-skilled individuals.
The development teams also have on-site customers with substantial domain knowledge to help them better understand the requirements (Abrahams son, Solo, Ronkainen, & Warsta, 2002). Multiple short development cycles also enable teams to accommodate request for change and provide the opportunity to discover emerging requirements (Highsmith, 2002). The agile approach promotes micro-project plans to help determine more accurate scheduling delivery commitments (Smits, 2006).

M Lindvall, V Basili, B Boehm, P Costa, (2002), [17] summarize the working definition of agile methodologies as a group of software development processes that must be iterative (take several cycles to complete), incremental (not deliver the entire product at once), self-organizing (teams determine the best way to handle work), and emergent (processes, principles, and work structures are recognized during the project rather than predetermined). In the paper by (Abrahamsson, Warsta, Siponen & Ronkainen, 2003), in general, characterized agile software development by the following attributes: incremental, cooperative, straightforward, and adaptive. Boehm, B., & Turner, R. (2005), generalize agile methods are lightweight processes that employ short iterative cycles, actively involve users to establish, prioritize, and verify requirements, and rely on a team ‘s tacit knowledge as opposed to documentation.

AGILE methodology is a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. Both development and testing activities are concurrent unlike the Waterfall model. In this agile approach digital transformation its all about software development process in project. Digital transformation means far more than a customer focused technology transformation, companies can achieve agile transformation in three level, such as project level, portfolio level and organization level. From last 18 years agile work with companies for successful software development, and in development activities agile provides rapid development. Todays IT world is very challenging and hence to successful development and delivery of software product agile need to become digital or agile need to transform it into digital world. Basically previously in development process all the task of development done manually, like planning, development of code, script execution of code etc. but with agile it is possible rapid development and with quality software. Agile not only perform rapid development but it performs core part in rapid deployment of software. So we can say that agile work on continuous integration, development and deployment, this entire process does in company with the help of automation tool JIRA, and Jenkins. This is the concept of agile approach to digital transformation for continuous integration and deployment.

Literature Survey:

1. Tore Dyba and Torgeir Dingsoyr in 2008 have conducted review on Agile development in their paper “Empirical studies of agile software development: A systematic Review”. In that they discuss the benefits and limitations of agile development, they identifies factors that contribute to agile software development as human and social factors, etc.

2. Yu Beng Leau, Wooi Khong Loo, Wai Yip Tham, and Soo Fun Tan in 2012 have conducted review on agile methodology vs traditional methods in their “Software Development Life Cycle AGILE vs
Traditional Approaches,” paper. In this they provide way to choose appropriate method for software development according to project team, time, and other factors. They conclude review on traditional SDLC and agile SDLC, where agile SDLC excels traditional SDLC. However, agile SDLC has its own disadvantage that agile is best suitable for small and medium size project and for large scale project traditional SDLC are still suitable. SDLC selection is depend upon time, project requirement, risk, size, complexity of software, business strategy and many other factors.

3. Marian STOICA, Marinela MIRCEA, Bogdan CHILIC-MICU in 2013 have conducted review on agile vs traditional development methods in their “Software Development: Agile vs. Traditional” paper. In this they discuss some advantages and disadvantages of traditional methods such as waterfall and incremental method, and some techniques to use and improve agility in an organization to gain strategic advantages and market success.

4. Yasuyo KOFUNE and Takahiro KOITA in 2014 have conducted review on agile software development and writing programming skill in their “Empirical studies of Agile software development and learn programming skills” paper. In this they conclude education support system based on agile development encourages and build communication between students, and improve student understanding about programming skills.

5. Alfred Zimmermann, DierkJugel, Kurt Sandkuhl, Rainer Schmidt, Justus Bogner, Stefan Kehrer in 2016 have conducted review on business process improvement and identify need for an extended understanding and support of collaborative decision-making in the process of architectural adaptation and enterprise transformation. It also define some limitation that there is need to integrate more analytics based decisions and context data driven architecture decision making. In their future work they discuss about digital transformation and digital design model or architecture.

6. Bettina Horlach, Paul Drews, and Ingrid Scirmer in 2016 have conducted review on Digital transformation in IT Industries in their “BiomodalIT: Business-IT Alignment in the Age of Digital transformation” paper. In this they discuss about importance of digital transformation in today’s IT world. Biomodel basically deals with business IT alignment with organizations. They also define some characteristics of digital and traditional IT.

7. Rajan Padayachee, Machdel Matthee, and Alta Van der Merwe in 2017 discuss some new technologies use within the workplace. They discuss about CIO (Chief information officers) and their 6 key decision making dimensions which guide technology decision making in agile business environment and strategies to support business transformation.

8. Nesma Keshta and Yasser Morgan in 2017 IEEE conduct review on traditional process and Agile Process based on project domain and team in their, “Comparison between Traditional Plan-based and Agile Software Process According to team size & Project Domain” paper. Witch prove that agile software development is most preferred approach used in software development. In this paper they

Agile vs Traditional Model/Methodology:
Base paper “Comparison between Traditional Plan-based and Agile Software Processes According to Team Size & Project Domain” concludes that each methodology can appropriately solve specific problems according to the requirements of the project. Accordingly, if an organization decides to support only a single methodology, it has to limit the acceptance of projects according to the chosen methodology. However, traditional methods and agile method do not overlap. Thus, they can coexist in the same organization. Organizations should have all the tools, and then according to the current running project, decide which tool to use. T. Dyba, & T. Dingsøyr, summarize the differences between Agile development and traditional development basis on the of an unpredictable world, as well as emphasizing the value competent people and their relationships bring to software development. Agile methods address the challenge of an unpredictable world, emphasizing the value competent people and their relationships bring to software development. Different researchers compare traditional and agile approaches, in their different perspectives, are summarized in Table 2 (All sources from additional information).

One main difference between the traditional and agile methodologies is the sequence of the phases in which the software development project is completed. The traditional method uses a linear approach where the stages of the software development process must be completed in a sequential order. This means that a stage must be completed before the next one begins.

These stages usually comprise the following:

- Requirements gathering and documentation
- System design
- Code and unit testing
- System testing
- User acceptance testing
- Bug fixes
- Product delivery

On the other hand, the agile methodology uses an iterative and team-based approach. Its main objective is to quickly deliver the application with complete and functional components. Instead of completing the software development tasks in sequence, they are completed in sprints that run from around 1 to 4 weeks and where a list of deliverables is completed in each sprint. The tasks that do not get completed within the sprint are then reprioritized and included in future sprints. This also means that the different stages of the software development life cycle can be revisited as needed.

The typical agile approach involves the following stages:

1. Project initiation
2. Sprint planning
3. Demos

With the traditional method, the details of the entire project have been visualized and defined before the project starts. In contrast, the agile methodology allows for more flexibility in that changes can more easily be made even after the project starts. It is best employed if the scope of the project cannot be clearly defined in advance. This also means that making unplanned software development changes with the traditional method is costlier than with agile.
Digital Transformation Using Tool

Digital transformation is the change associated with the application of digital technology in all aspects of human the transformation stage means that digital usages inherently enable new types of innovation and creativity in a project domain rather than simply enhance and support traditional methods. Digital transformation is the integration of digital technology into all areas of a business, fundamentally changing how you operate and deliver value to customers. Digital transformation closes the gap between what digital customers already expect and what analog businesses actually deliver.

1. Jenkins

Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.

Jenkins provides good support for providing continuous deployment and delivery. If you look at the flow of any software development through deployment, it will be as shown below.

![Jenkins Workflow](image)

CONTINUOUS INTEGRATION AND DEPLOYMENT WITH JENKINS—LIFE CYCLE:

1. Continuous Integration

Figure 5: Software development through deployment.

The main part of Continuous deployment is to ensure that the entire process which is shown above is automated.

Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. The following flowchart demonstrates a very simple workflow of how Jenkins works.

![Jenkins Workflow](image)

Figure 6: Jenkins workflow
Continuous integration is the process of merging development works with the branch(code) several times in a day constantly. Most of the work here is done by automated.

2. Continuous Delivery

Continuous delivery is the practice of delivery of code to an environment, whether it is a QA team, so they can review it. After the issues get resolved, they can deploy in production.

3. Continuous Deployment

You can think of continuous deployment as the next step of continuous delivery, when each change that passes the automated tests are deployed to production. Continuous deployment relies on an infrastructure that automates and instruments the process of testing, integration and deployment of new features.

Most of the time testing and deployment steps do not change frequently and in order to keep the developer focus on writing code we do the automation of testing and deployment. This automation is called “continuous integration and deployment”.

**LIFE CYCLE TO WORK**

Here is the complete flow of the system.

1. Install Jenkins and configure
2. Adding Github webhook to push events to Jenkins
3. You made some changes in your project.
4. You push those changes in Github on master or any branch.
5. Github will notify Jenkins about the new push. (We have to configure with jenkins & Git)
6. Jenkins will then run the commands you ask it to run.
7. Those commands will contain following,
8. Test script.

Deployment script will be added to Project only and Jenkins will use that to communicate to server by ssh and perform the push. Basically, what you do manually such as running test command, login to server, performing Git pull and then restarting build server, Jenkins will do all of them automatically by writing the script.

![Jenkins Server](Figure 7: Jenkins Server)
Jenkins Pipeline

Jenkins Pipeline introduced a world where pipelines are defined and managed from the version control repository that houses the code the pipeline delivers. Simply place the pipeline code in a file called Jenkins file and keep it in your project's repository. This is a powerful idea. It means the definition of the process, which the code must follow to be delivered, lives with the project's code itself. It means the process definition's history is recorded with the project's code history. It means any pipeline can be shared and replicated at any time.

Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins.

A continuous delivery (CD) pipeline is an automated expression of your process for getting software from version control right through to your users and customers. Every change to your software (committed in source control) goes through a complex process on its way to being released. This process involves building the software in a reliable and repeatable manner, as well as progressing the built software (called a "build") through multiple stages of testing and deployment.

Conclusion:

Agile software development methodologies are evolutionary and incremental models have become increasingly popular in software development industry. Through, in many organizations, agile system development methods at adoption stage, agile methods might start to become well-established processes of these small, mid-level, even large organizations. There is increasing need to have a deeper understanding of agile methods in use in software development industry; as well as, have a better understanding – the benefits of agile approach as for accepting agile methods into their development style and for cope-up with their dynamic business needs.

References:


