American Journal of Engineering Research (AJER)
E-ISSN: 2320-0847 p-ISSN: 2320-0936
Volume-8, Issue-2, pp-199-207
www.ajer.org
Research Paper Open Access

A Review on the Process of Adoptability of Agile Methods in Software Development Practices

Suleiman Abdullahi[1], Lawal Idris Bagiwa[2]
[1]*Department of Mathematics and Computer Sciences, Faculty of Natural and Applied Science, Al-Qlam University, Katsina, P.M.B. 2137 Dutsin-ma Road, Katsina State, Nigeria.*
[2]*Department of Computer Studies, College of Science and Technology, Hassan Usman Katsina Polytechnic P.M.B. 2052 Katsina State, Nigeria.*
Corresponding Author: Suleiman Abdullahi

ABSTRACT: *In agile system for software development is no longer a new concept. However, few organizations are psychologically or technically able to take on an agile approach rapidly and effectively to take advantage of the numerous benefits that it offers to an organization. Those benefits include, but are not limited to, quicker return on investment, better software quality, and higher customer satisfaction. This paper is aimed at reviewing the process involved in various agile methods in software development. The paper provides process that guides organizations in adopting agile practices. The paper consists of agile measurement index and processes in stages: Identification of Discontinuing Factors, Project Level Assessment, Organizational Readiness Assessment and Reconciliation, that together guide and assist the agile adoption efforts of an organizations. The processes helps determine whether or not organizations are ready for agile adoption, and guided by their potential, what set of agile practices can and should be maintained by the organization.*

Keywords: *Agile, Agile Methods, Agile Software Development, Agile Adoption, Software Development Process.*

Date of Submission: 15-02-2019 Date of acceptance: 28-02-2019

I. INTRODUCTION

Agile methods are often welcomed by both managers and programmers as providing a much needed release from the overheads typically perceived as being imposed by traditional software development approaches [21]. Created in the context of small, greenfield projects, agile methods are often seen as unable to scale to larger situations[23]. Their adoption seems to need an all-or-nothing approach, suggesting that "being agile" is binary [4].

In practice, few organizations are able, psychologically or technically, to take on agile development approaches immediately and adopt them successfully over a short period – a full transition often taking a few years [11,17]. Furthermore, it may be inappropriate for them to be fully agile in all aspects of development, perhaps retaining well-known and trusted elements of a more traditional approach within an overall agile project. One way to do this is by the use of situational method engineering [7]. But even then, the method engineer and the software development manager may be unsure how to identify how to adopt agile methods incrementally, which bits to choose as most appropriate for their situation, how to engender enthusiasm in team members[3], how to ensure that their adopted method can mature and grow as the development team's skills mature and how to ensure that the whole of the development team don't succumb to the inherent desire of humankind to "resist change" [1].

In this paper, a complete process to assist managers in complying with the agile software development principle and manifesto as well as assessing the degree of agility they require and how to identify appropriate ways to introduce this agility into their organization was proposed.

1.1 Review Background

Many early attempts to improve software development focused on better ways of defining and detailing requirements, designing comprehensive architectures to support the requirements, and then developing the software in a very regimented, methodical manner to realize the system and supporting architecture.[20]

Although some approaches were considered iterative in nature, they did not go far enough in addressing the needs of effectively managing rapidly changing requirements nor in accelerating the delivery of software[19].
By the late 1990s [8] a majority of the software development processes that had been developed in the 1980s and 1990s were being criticized as bureaucratic, slow, and overly regimented [11]. In the mid-1990s, in reaction to these heavyweight software methods, there was a small contingent of industry thought-leaders promoting innovative approaches to software, enabling development organizations to quickly react and adapt to changing requirements and technologies. They realized that embracing change, and executing in a manner that not only accommodated this change, but fostered it, would result in a much more successful development strategy [6].

The term "agile software development" emerged from a gathering of these industry thought-leaders in Snowbird, UT in 2001[18]. The term was first used in this manner and published in the now famous (or infamous) Agile Manifesto [15].

This term was used as an umbrella reference to a family of emerging lightweight software development methods such as Scrum, Extreme Programming(XP), Dynamic System Development Methods(DSDM), Future Driven Development(FDD), Crystal Clear Method, and Adaptive Software Development(ASD) [2]. Instead of emphasizing up-front planning and detailed requirements, these methods placed significant emphasis on continual planning, empowered teams, collaboration, emergent design, a test-early and often philosophy, and, most importantly, the frequent delivery of working software in short, rapid iterations [9].

Since the publication of the Agile Manifesto, other thought-leaders have continued to evolve agile thinking, drawing on lessons learned in other industries – for example, in the ideals and approaches promoted by Lean Development, and most recently, Kanban[13].

This review addresses the current absence, at least in the public domain, of structured approaches to guide agile adoption efforts combined together. Furthermore, a rigorous formalization of what constitutes agility is also missing. Organizations aspiring to become agile want to know when they are considered "agile," as well as what it means to be "agile". Moreover, guidelines highlighting what is needed to help agile adoption efforts succeed are unavailable. These guidelines are essential for determining if any activities or tasks are overlooked during the adoption process [17].

The lack of a structured approach for agile adoption causes organizations to question how to identify the right practices to adopt, how to determine if they are ready for agile, what the necessary preparations for agile are, and what the potential difficulties that could develop during the adoption process are [6].

1.2 Solution Approach

The Agile Adoption processes reviewed in this paper was a structured and efficient approach to guide agile adoption efforts within projects without overlooking the organizational aspect of the adoption process.
The first component, serves three important purposes.
• It serves as a tool to measure and assess the agile potential of an organization independent of any particular agile method (e.g. XP, Scrum …etc) It provides a scale for identifying the target agile level for a project aspiring to adopt agility.
• The measurement index helps the coach organize and group the agile practices in a structured manner based on essential agile qualities and business values.
• It provides a hierarchy of measurable indicators used to determine the agility of an organization [1,5,23].

II. AGILE AND AGILE SOFTWARE DEVELOPMENT

Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end [10].

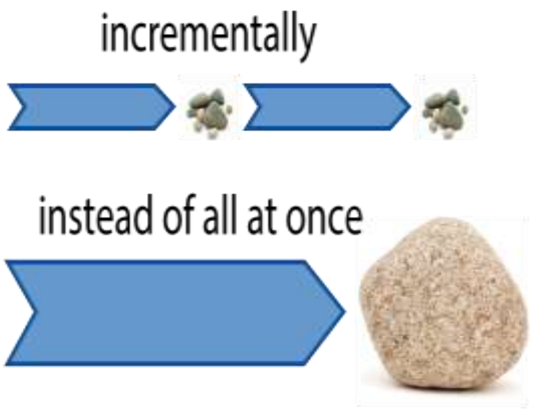


Figure 2.1: Agile and agile software development.

It works by breaking projects down into little bits of user functionality called user stories, prioritizing them, and then continuously delivering them in short two week cycles called iterations [20].

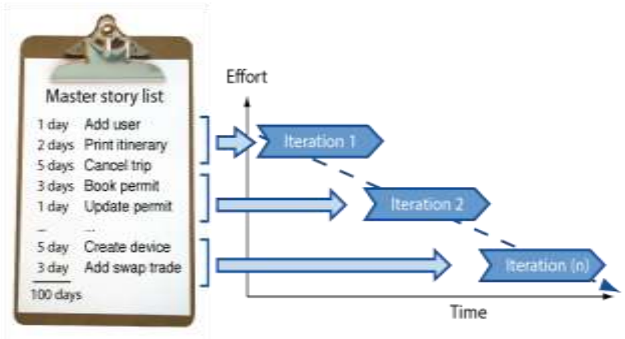


Figure 2.2: Agile and agile software development, iterative process

2.1 Traditional Process Models and Agile Software Development

This section provides an overview of the traditional models and agile model of software development.

2.1.1 Overview of Traditional Process Models

Different kinds of software processes have been used during the history of software development [2]. All of the process models have at least two components in common: analysis and coding [19]. This is a very primitive way of doing software [21]. Calls the method code-and-fix method. The process is very simple, as the steps are the following; 1) write some code, 2) fix the problems found in the code. In other words: get into the real work as fast as possible and think about requirements, design, testing and maintenance later. The method is still widely used and is usable in small in-house tools and scripts. If the size of the code is small enough, no high level planning is needed – especially if the program is intended for personal use only or for the use of the development team who wrote the program. As the program grows bigger, if developed using code-and-fix method, the structure of the software gets so fragmented that new fixes become more and more expensive all the time. Some kind of a process is needed in order to make the development work controlled and predictable [17].

2.1.2 The waterfall model

The waterfall model is a sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through several phases, typically:

Requirements analysis resulting in a software requirements specification, Software design, Implementation, Testing, Integration, if there are multiple subsystems, Deployment (or Installation) and Maintenance [3].

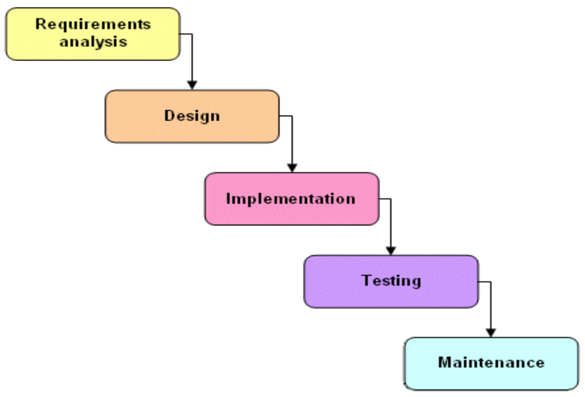


Figure 2.3: The waterfall model of software development

2.2 Agile Software Development

"Agile software development" refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve via collaboration between self-organizing cross-functional teams. The term was coined in the year 2001 when the Agile Manifesto was formulated [9].

Agile software development uses iterative development as a basis but advocates a lighter and more people-centric viewpoint than traditional approaches [17]. Agile processes fundamentally incorporate iteration and the continuous feedback that it provides to successively refine and deliver a software system [22].
There are many variations of agile processes:
Dynamic systems development method (DSDM), Kanban, Srum and XP.

Dynamic systems development method (**DSDM**) is an agile project delivery framework, primarily used as a software development method.[18] First released in 1994, DSDM originally sought to provide some discipline to the rapid application development (RAD) method [17]. In 2007 DSDM became a generic approach to project management and solution delivery[2]. DSDM is an iterative and incremental approach that embraces principles of Agile development, including continuous user/customer involvement [1].

Kanban is a method for managing knowledge work with an emphasis on just-in-time delivery while not overloading the team members [23]. In this approach, the process, from definition of a task to its delivery to the customer, is displayed for participants to see and team members pull work from a queue [17].

Scrum is an iterative and incremental agile software development framework for managing product development. It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal", challenges assumptions of the "traditional, sequential approach" to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines in the project.[10,15,20,21].

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[2,4] it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted.

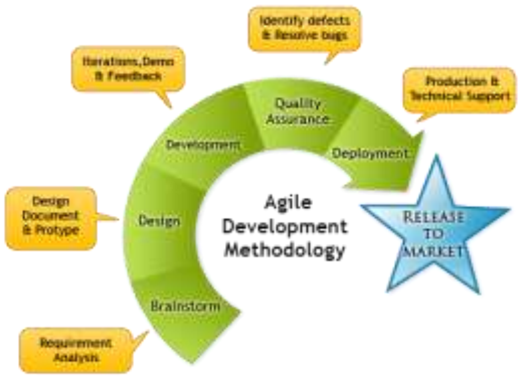

Identify defects & Resolve bugs
Iterations,Demo & Feedback
Production & Technical Support
Quality Assurance
Development
Deployment
Design Document & Protype
Design
Agile Development Methodology
RELEASE TO MARKET
Brainstorm
Requirement Analysis

Figure 2.4: Agile model of Software Development

2.3 Agile Versus Waterfall software development models

Agile software development undoubtedly offers advantages that a waterfall approach can't begin to address. Where the waterfall approach is based in predictability and processes, an Agile approach focuses on adaptability and response time to changing requirements. Another important advantage of Agile over the waterfall model is the recursiveness of the work pattern. This means that we can make modifications to the completed stage in Agile while it is not allowed under waterfall model [16].

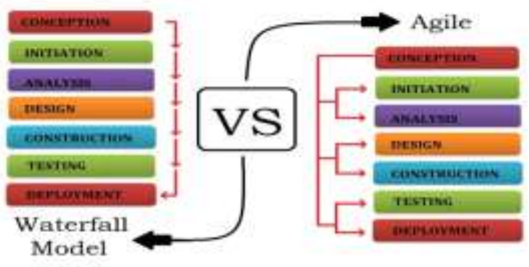


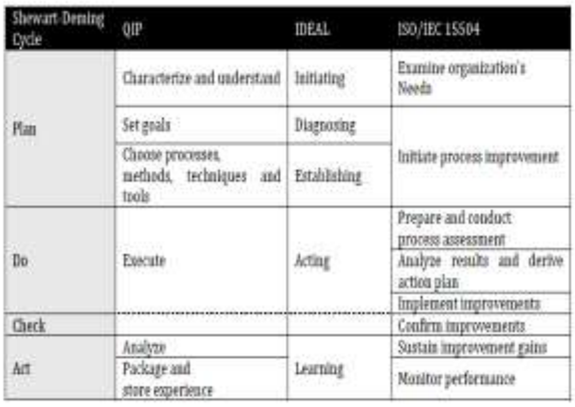
Figure 2.5: Agile Versus Waterfall software development models

2.4 Process Improvement and the Agile Adoption Framework

Since adopting agile practices is essentially a process improvement effort, it is useful to the understanding of the components of the Agile Adoption Framework to discuss some generic process improvement frameworks and models [15].

1. Understand the status of the development process
2. Develop a vision of the desired process
3. List improvement actions in order of priority
4. Generate a plan to accomplish the required actions
5. Commit the resources to execute the plan
6. Start over at step 1[4].

Table 1. Organizational Process Improvement Models



Shewart-Deming Cycle	QIP	IDEAL	ISO/IEC 15504
Plan	Characterize and understand	Initiating	Examine organization's Needs
	Set goals	Diagnosing	Initiate process improvement
	Choose processes, methods, techniques and tools	Establishing	
Do	Execute	Acting	Prepare and conduct process assessment
			Analyze results and derive action plan
			Implement improvements
Check			Confirm improvements
Act	Analyze	Learning	Sustain improvement gains
	Package and store experience		Monitor performance

2.5. Overview of the IDEAL Model

The IDEAL model provides a disciplined engineering approach for improvement. It focuses on managing the improvement program, and establishes the foundation for a long- term improvement strategy. The model consists of five phases:

• **I – Initiating:** Laying the groundwork for a successful improvement effort.
•**D – Diagnosing:** Determining the present state and desired state and developing recommendations for improvement.
• **E – Establishing:** Planning the specifics of how to reach SPI initiative's target.
• **A – Acting:** Doing the work according to the plan.
• **L – Learning:** Learning from the experience and improving the ability to adopt new technologies in the future.[11]

2.6 SPI Lifecycle Models for Agile Development

Moving an organization toward having an agile development process through the adoption of agile practices is a type of process improvement effort. Organizations are increasingly recognizing the need for specific implementation guidance when they adopt new software engineering tools, processes and methods [12].

2.6.1 Agile principles

The Agile Manifesto is based on 12 principles:

1. Customer satisfaction by rapid delivery of useful software
2. Welcome changing requirements, even late in development
3. Working software is delivered frequently (weeks rather than months)[22]
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the principal measure of progress[1]
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Self-organizing teams [2,4,7,10]
12. Regular adaptation to changing circumstances [11]

III. AGILE ADOPTION: THE STAGES PROCESS

The Agile Adoption process is a structured and repeatable approach that would guide and assist agile adoption efforts. It will assist the agile community in supporting the growing demand from organizations that want to adopt agile practices. The main component of the Agile Adoption Framework is the 4- Stage Process, which utilizes the Agile Measurement Index to help an organization adopt agile practices. The Agile Measurement Index is a scale the agile coach uses to identify the agile potential of a project or organization.

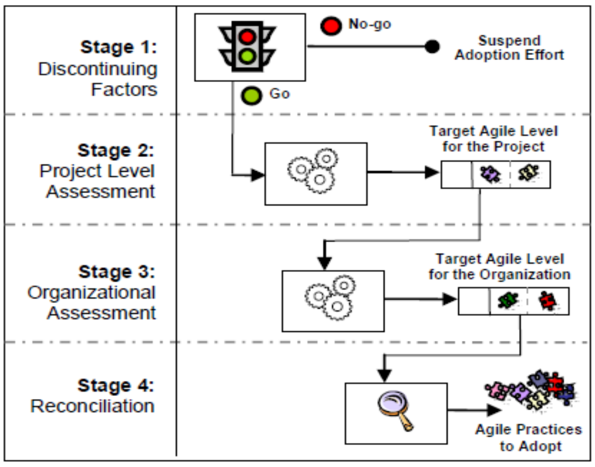


Figure3.1: The 4Stage Process for Agile Adoption

As depicted in Figure 6, the 4- Stage Process consists of four pieces that work together to help the assessor determine if (or when) an organization is ready to move towards agility, or, in other words, make the go/no- go decision, and assists him or her in the process of identifying which agile practices the organization should adopt. The four stages are:

Stage 1: Discontinuing Factors. Discovers the presence of any roadblocks (or showstoppers) that can prevent the adoption process from succeeding.

Stage 2: Project Level Assessment. Utilizes the Agile Measurement Index to determine the target level of agility for a particular project.

Stage 3: Organizational Readiness Assessment. Uses the Agile Measurement Index to assess the extent to which the organization can achieve the target agility level identified for a project.

Stage 4: Reconciliation. Determines the final set of agile practices to be adopted by reconciling the target agile level for a project (from Stage 2) and the readiness of the embodying organization (from Stage 3)[20]

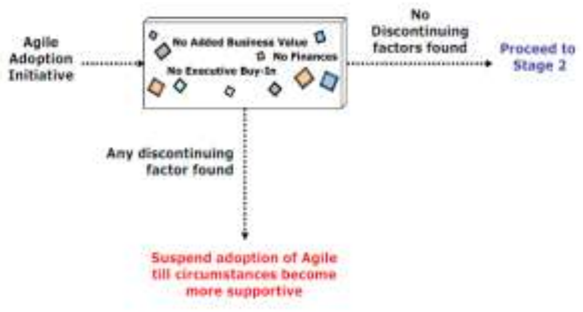

Agile Adoption Initiative
No Added Business Value
No Finances
No Executive Buy-In
No Discontinuing factors found
Proceed to Stage 2
Any discontinuing factor found
Suspend adoption of Agile till circumstances become more supportive

Figure 3.2: Stage 1: Discontinuing Factors

The first step in Stage 1 of the 4- Stage Process is to identify the factors that could adversely impact the agile adoption process. These Discontinuing Factors are organizational characteristics that, if present in an organization, can hinder or jeopardize the success of the agile adoption process. These factors can vary from organization to organization and from one agile consultant to another.

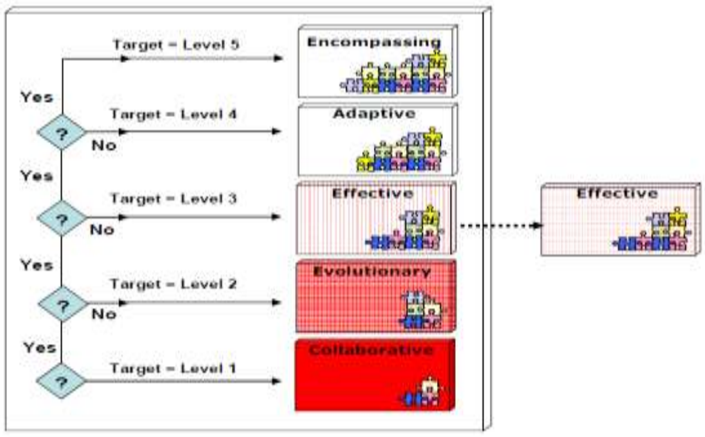


Figure 3.3: Stage 2 : Project Level assessment

After the stakeholders make the decision to go ahead with the agile adoption effort (from Stage 1), the next stage looks at the individual projects that will adopt agile practices and determines which level of agility (based on the Agile measurement index) each should adopt. Since each project is different and is surrounded by unique circumstances, each project needs to adopt a level of agility that is best suitable for it.

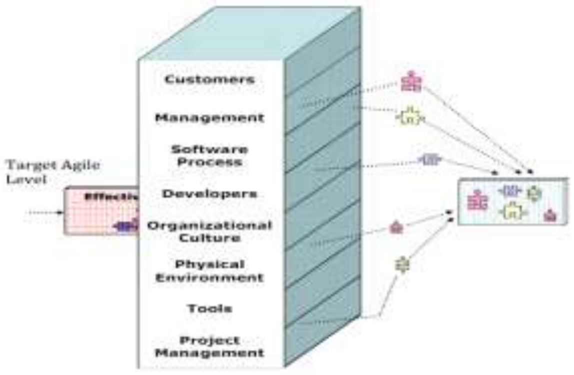


Figure 3.4: Stage 3: Organizational Readiness Assessment

Identifying the target level for a project does not necessarily mean that that level is achievable. Determining the achievable level requires an assessment of the readiness of the organization to adopt each of the agile practices up to, and including, the target level.

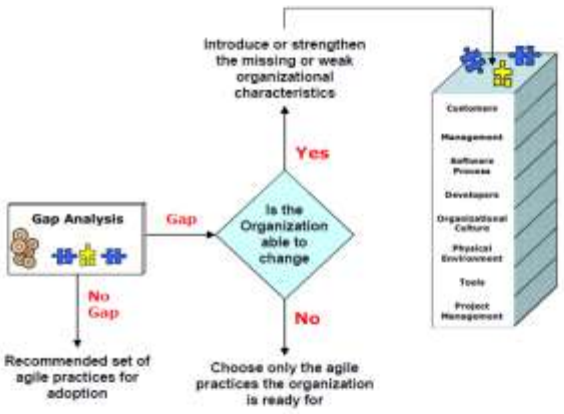

Introduce or strengthen the missing or weak organizational characteristics
Yes
Gap Analysis
Gap
Is the Organization able to change
No Gap
No
Recommended set of agile practices for adoption
Choose only the agile practices the organization is ready for
Customers
Management
Software Process
Developers
Organizational Culture
Physical Environment
Tools
Project Management

Figure 3.4: Stage 4: Reconciliation

Following the organizational readiness assessment, the agile level achievable by the organization is known. Prior to that, Stage 2 had identified the agile level that the project aspires to adopt. Therefore, the final step, reconciliation, is necessary to determine the agile practices the project finally adopts. In essence, during this stage the assessor analyzes the results of the organizational assessment and makes a set of recommendations to the organization on how to proceed, especially if the organization's readiness level is less than the project target level[17].

IV. CONCLUSION

This paper provided a brief overview of the agile software development processes as applied to different types of organizations. The paper also discussed the different stages of agile software development. The review also looked at factors that should be considered when adopting agile method by organizations.

ACKNOWLEDGEMENT

We are heartily thankful to all those who contributed in one way or the other to see that this research reach its final stage.

REFERENCES

[1]. Agarwal, A., Shankar, R. and Tiwari, M. (2006). Modeling the metrics of lean, agile and leagile supply chain: An ANP-based approach. European Journal of Operational Research, 173(1), pp.211-225.

[2]. Agilenutshell.com, (2014). Agile and agile software develpment. [online] Available at: http://www.agilenutshell.com/ [Accessed 8 Dec. 2018].

[3]. Agilesherpa.org, (2014). Agile Development - A Brief History | Agile Sherpa | Agile Sherpa. [online] Available at: http://www.agilesherpa.org/intro_to_agile/a_brief_history_of_agile/ [Accessed 9 Dec. 2018].

[4]. Akbar, R., Hassan, M., Abdullah, A., Safdar, S. and Qureshi, M. (2011). Directions and Advancements in Global Software Development: A Summarized Review of GSD and Agile Methods. Research J. of Information Technology, 3(2), pp.69-80.

[5]. Amescua, A., BermoÌ • n, L., GarciÌ • a, J. and SaÌ • nchez-Segura, M. (2010). Knowledge repository to improve agile development processes learning. IET Softw., 4(6), p.434.

[6]. Ben Naylor, J., Naim, M. and Berry, D. (1999). Leagility: Integrating the lean and agile manufacturing paradigms in the total supply chain. International Journal of Production Economics, 62(1-2), pp.107-118.

[7]. C., Williams. (2011). A comparative Analysis of the Agile and Traditional Software Development Processes Productivity. International Conference of the Chilean Computer Science Society. [online] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6363385 [Accessed 6 Nov. 2018].

[8]. Chan, F. and Kumar, V. (2009). Performance optimization of a leagility inspired supply chain model: a CFGTSA algorithm based approach. International Journal of Production Research, 47(3), pp.777-799.

[9]. Comparativeagility.com, (2014). Comparative Agility. [online] Available at: http://www.comparativeagility.com/ [Accessed 10 Oct. 2018].

[10]. Jain, A. and Rani, R. (2011). Analytical Study of Agile Methodology with Cloud Computing. International Journal of Computer ApplicationsÂ® (IJCA). [online] Available at: http://research.ijcaonline.org/rtmc/number14/rtmc1076.pdf [Accessed 9 Dec. 2014].

[11]. Javdani, T. et al (2014). How Human Aspects Impress Agile Software Development Transition and Adoption. International Journal of Software Engineering and Its Applications, [online] 8(1). Available at: http:// research.ijcaonline.org/ rtmc/number14/ rtmc1076.pdf [Accessed 17 Nov. 2018].

[12]. Kaisti, M., Rantala, V., Mujunen, T., Hyrynsalmi, S., KÃ¶nnÃ¶lÃ¤, K., MÃ¤kilÃ¤, T. and Lehtonen, T. (2013). Agile methods for embedded systems development - a literature review and a mapping study. EURASIP Journal on Embedded Systems, 2013(1), p.15.

[13]. Manamendra, M. (2013). Improvements for agile manifesto and make agile applicable for undergraduate research projects. The 8th International Conference on Computer Science & Education (ICCSE).

[14]. Modrak, V. and Marton, D. (2013). Development of Metrics and a Complexity Scale for the Topology of Assembly Supply Chains. Entropy, 15(10), pp.4285-4299.

[15]. Nageswara Rao, K., Kavita Naidu, G. and Chakka, P. (2011). A Study of the Agile Software Development Methods, Applicability and Implications in Industry. International Journal of Software Engineering and Its Applications, [online] 5(2), pp.35-44. Available at: http://www.sersc.org/journals/IJSEIA/vol5_no2_2011/4.pdf [Accessed 1 Feb. 2019].

[16]. Olague, H., Etzkorn, L., Gholston, S. and Quattlebaum, S. (2007). Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes. IIEEE Trans. Software Eng., 33(6), pp.402-419.

[17]. Popli, R., Anita, and Chauhan, N. (2013). A Mapping Model for Transforming Traditional Software Development Methods to Agile Methodology. IJSEA, 4(4), pp.53-64.

[18]. Ramesh, G. and Devadasan, S. (2007). Agility assessment through qualification and quantification tools: a case study in an Indian pump manufacturing company. International Journal of Mass Customisation, [online] 2(1), pp.139-160. Available at: http://inderscience.metapress.com/content/2wnt4d4hm5v56bb6/ [Accessed 8 Jan. 2019].

[19]. Sun, Y., Zhang, Z. and Soehartono, E. (2006). A framework to assist supply chain agility development decision-making: a case study in an aerospace manufacturing company. IJASM, 1(3), p.244.

[20]. Versionone.com, (2014). Benefits of Agile Scrum Software Development | VersionOne. [online] Available at: http://www.versionone.com/agile-101/agile-software-development-benefits/ [Accessed 6 Dec. 2018].

[21]. Wikipedia, (2014). Agile software development. [online] Available at: http://en.wikipedia.org/wiki/Agile_software_development [Accessed 3 Dec. 2018].

[22]. Wikipedia, (2014). Software development process. [online] Available at: http:// en.wikipedia.org/wiki/ Software_development_process [Accessed 8 Jan. 2019].

[23]. Yang, S. and Li, T. (2002). Agility evaluation of mass customization product manufacturing. Journal of Materials Processing Technology, 129(1-3), pp.640-644.