A New Approach for Distributed Software Engineering Teams Based on Kanban Method for Reducing Dependency

Maha Khaled Yacoub[1*], Mostafa Abdel Athim Mostafa[2], Ahmed Bahaa Farid[3]

[1] Arab Academy for Science and Technology, Department of Information Systems, Cairo, Egypt, Egyptian.

[2] Arab Academy for Science and Technology and Maritime Transport, Cairo, Egypt, Department of Information Systems, Cairo, Egypt.

Helwan University, Information Systems Department, Helwan, Egypt.

* Corresponding author. Email: Maha.k.yacoub@gmail.com

Manuscript submitted February 2, 2016; accepted April 10, 2016.
doi: 10.17706/jsw.11.12.1231-1241

Abstract: In today's globalization with the shift towards Agility. Agile practices highly evaluates communication to achieve improvement in the software development process and among development team members. Stable communication can be difficult and a challenge in distributed agile environment. Pointless communications and actions in software development projects do not decrease work in progress and may cause failure. Getting rid of such a waste leads to a continuous flow process improvement, decreasing team dependency. This is astudy of a distributed Kanban, for web portal as a proposed solution for mitigating the communication challenges encountered in such environment. The declared benefits of Kanban have gained. significant importance as an agile project management method in the software development industry. Kanban approach helps managing and measuring the flow of work across developing teams. Results show that Kanban has a positive effect, leading the team to communicate together when it is needed. Compared to the previously used Scrum method by the sample teams, the use of Kanban causes dependency reduction for the distributed team by 27%, keeping the pace of the development harmonized across the whole team and improving maturity. This led to 98.9 of a done status for user stories, and 30% customer satisfaction increase.

Key words: Kanban, agile, software development, communication, dependency; software engineering.

1. Introduction

Shifting software development industry from co-located software development to distributed Software Development, led to a great competition between software developments companies around the world. Growing interest continued by applying Agile development practices alongside distributed software development in order to gather the benefits of both approaches. Organizations are more and more likely distributing their production globally. They do that in order to catch some distribution fruits including , but not limited to; time-zone production independence, access to well-educated labour, technical infrastructure maturation,and cost reduction[1], improved proximity to the market and customers, and ability to reduce time to-market, Therefore, software development is increasingly distributed to multiple sites involving different cultures.

Agile software development success process depends on team interaction. The performance of the development team determined by the interaction among team members, team interactions affects in distributed teams. Distribution of teams globally struggle number of challenges. Top issues interactions in distributed development projects are process management, communication, culture, technical and security. With consistent, timely, and effective communication while developing software, in order to achieve and improve the profession, communication is essential and must be effective to ensure that the benefits of global delivery aren't lost between the distributed parties. However, there are several factors contributes in thechallenges that hinder communication in distributed environment.Geographical, cultural and temporal distances as the key barriers for communication in globally distributed environments [2] and could cause communication waste [3]. Results of observations about communication and distance highlight the importance of understanding the dependencies among team members involved in software development and how it affects the communication between members.Dependenciescan be understood in two ways. First, dependencies between collaborators could causea delay waste represented by developer waiting for the analyst, tester waiting for the developer, and so on. The second, dependencies are those between tasks themselves. Waste is defined as any human activity consuming resources but not providing value [4].The concept of waste as a lens used to identify non-value producing communication elements. Eliminating waste helps companies to identify communication issues that are present in their development efforts[5].

Lean thinking was the practice used by Toyota to manage production operations for decades.Later on it was adopted by the software engineering field [6]. As Lean is the most recent addition to the Agile software development, it is used to eliminate all kinds of waste from the development process.

Kanban is one way to execute Lean thinking, in this study we focus on communication waste.Adopting the Kanban method has become more popular in the software development and has obtained powerful practitioner-driven movement born to support. This expectation is based on Kanban's characteristics of adaptability (welcoming changes) and visualization (visualising the progress for an easy management), as well as being successful in driving team members to cooperate and communicate [7].

The rest of the paper is structured as follows: Section two defines the problem. Section three:presents a literature review for the previous research efforts on the case study. Section4: describes the research design proposed method, including the process for collecting the data as well as the design of the survey and interviews. Section 5: presents the Thinkloud case study conducted design and results. Finally, Section 6: concludes the paper summarising the findings.

2. Problem Definition

Though with the global high tech industry super abounds with smart developers yet many teams continue to struggle to deliver projects on budget, on time, and with high quality [8], [9]. While people and technology remain part of the equation, much can be obtained from a systematic view of processes. Adopting wrong concepts as a practical method can crash your organization streamline, adding useless practice that casues wates. Waste is the opposite of value (a capability delivered to the customer through which the customer attains a tangible or intangible benefit) [10], [11]. So whatever feature or functionality or process step that neither adds value nor is used will be considered waste and should be eliminated from the process. Mary and Tom Poppendieck have defined seven wastes that are appropriate for software development. Understanding waste in software development, can help teams eliminate them (or at least reduce their impact) with respect to Agile software development. Generally, you will find these wastes during sprint planning or during sprint execution. Those seven types of waste can be stated as follows: partially done work, extra features, relearning, hand-offs, delays, task switching, defects. The following diagram in Fig. 1 of effects analysis the team dynamics, reflecting the effect of dependencies. As a result,

dependency will increase communication, cycle time, bugs, and stress. Task switching and working under stress may result in more bugs and will eventually decrease productivity, which, in turn, will decrease Throughput. In addition, as long as there is unclear requirements and lack of knowledge there will be a possibility for communication to occur among team. As a result, team members will start depending on each other in a way to accomplish work in progress. So reducing the amount of communication and coordination is a must; hence the team is a radical attack on the communication problem.

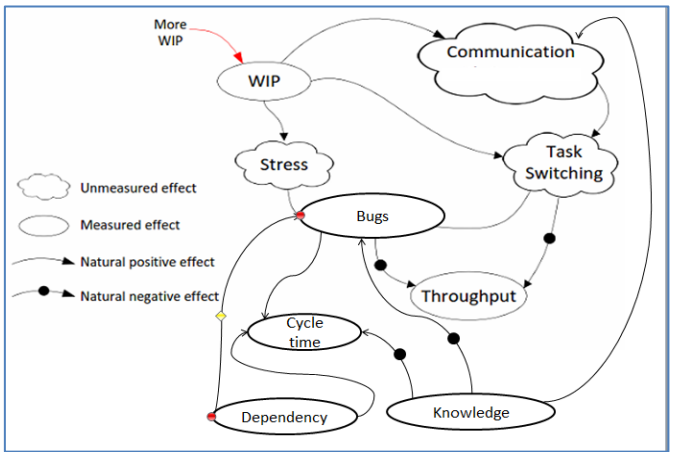

Fig. 1. Diagram of effects analysis the team dynamics, reflecting the effect of dependencies.

Ref. [12] state that there is much potential for conflict in distributed teams as members work across cultural, geographical and time-bound environments. Conflict leads to ineffective communication and as soon as team members stop communicating effectively, barriers begin to form between them leading to a decrease in productivity and interaction [13]. According to the study made by [14], 74% of the problems of distributed development were related to communication [15]. They found that the lack of communication or poor quality of it was often the root cause behind adopting un-suitable agile practices [16].

Using Scrum, Scrum is strict. Close communication is necessary, making the pure Scrum practice problematic [17], where un-needed communication in planned (daily stand up meeting), causing lack of motivation and waste.As a result Scrum promotes a negative impact (waste)Waste is anything that does not contribute to the value creation for the customer; hence, it should be removed (Al-Baik and Miller 2014).

3. Literature Review

Since software processes are largely dynamic and cooperative activities, the impact of software process technology with its modelling languages, editors, and interpreters has only been small. In the 1980s, crucial support processes, such as learning, technical communication, requirements negotiations, and customer interaction, were poorly described in software process practices[18]. The reasons for most project failures were not in the technical sector but seen in the managerial, due to poor team performance. This is caused by careless attention to teamwork issues since people have a intentions to concentrate on the technical aspects (hardware and software) rather than the people ware [19].

3.1. Scrum Challenges

Ref. [20] reported some results from surveys by the Design for Hybrid Agile Adoption Institute. 30% of respondents are using distributed Agile, 40% use local Agile development, and about 85% of them have distributed teams. In addition, the 30% of respondents that stated they are using distributed Agile explained their use of this approach by the advantages and successes they achieved from this kind of development, such as reducing the development cost, accessing the talent pool and resources, increasing

team productivity, and decreasing the cost of having high quality software[21].

The key features of agile methods are continuous requirements gathering; frequent face-to-face communication; daily meetings, pair programming; refactoring; continuous integration; early expert customer feedback; and documentation [22].

Based on theoretical predictions and empirical observations;dependencies between software components create dependencies between the developers implementing those components[23].

Tom Allen (professor at MIT, 2008) mentioned that the rich interaction in Scrum of face-to-face work, there is very convincing evidence that the frequency of communication drops off sharply with physical separation among co-workers' offices, and that the sphere of frequent communication is surprisingly small [14].

A systematic review [3] studied the application of Scrum practices in global software development using 27 literature studies, and analysed the challenges of using Scrum with distributed teams can be categorized into three types of deficiency: communication, trust, and control[4].

Each organized human activity involves the division of labour into various tasks, as well as the coordination of those tasks until the activity is done [16]. Coordination is the management of these dependencies. If there are no dependencies, there is nothing to coordinate and communicate. But if a task depends on the result of another task, or if the execution of a task depends on the expertise of the person that performs the task, the dependencies must be managed. Coordination of work is therefore an important challenge for teamwork and team leadership [18] that may cause communication waste [8].

Team size is another important aspect we tend to ignore. In Scrum, the team size should not exceed 9 people. Two or more Scrum teams will require more time for synchronizing, so you should be prepared to pay for communication overhead. Without extensive communication and work on soft skills, we may not achieve the expected performance boost for Scrum. [21] As a result effective Distributed Scrum is possible only when the customer is ready to invest in communication tools and cross-location business trips [3].

Communication tools themselves can create challenges. Sound quality of teleconferencing tools may be poor. (Williams and Stout, 2008)This can create misunderstanding and communication overhead can occur when messages need to be repeated several times [14].

Ref. [5] discussed the influence of the software architecture in the coordination of distributed software development. They argue that "the more cleanly separated the modules, the more likely the organization can successfully develop them at different sites", because this will remove the communication required among the different sites. According to [6], Kanban model is getting popularity from last few years due to its flexibility, responsiveness and reliability.

3.2. Kanban Agile/Lean Practice

Kanban means sign board or visualisation of inventory used in the scheduling system for just in time (JIT) production.It was developed by TaiichiOhno, and was introduced in the Japanese manufacturing industry in the 1950s. Kanban development concept was about finding a system to maintain and improve production. It is a flow management and control mechanism for pull-driven JIT production. Kanban has proven an excellent way of promoting improvement as it runs the production system as a whole. Kanban was successfully used in practice by Toyota [19].

Research by the National Institute of Standards and Technology found that effective implementation of lean manufacturing techniques can generate significant benefits; improve quality 25-75%, reduce work in process up to 90%, improve productivity 10-40%, reduce lead time up to 95%, multiple other benefits related to improved product flow, and enhance teamwork and communication, as shown in Fig. 2.

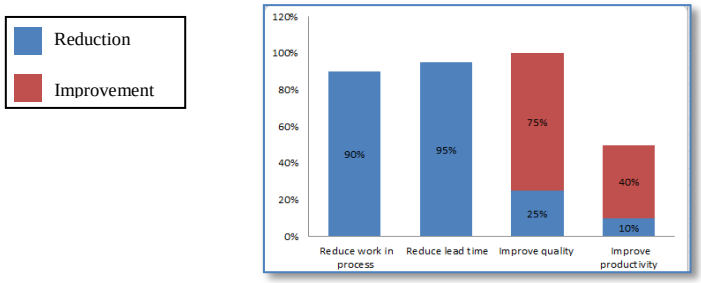

Fig. 2. The positive impact of Kanbanby the national institute of standards and technology.

Kanban encourages the pull technique. In traditional software development, the push approach is used where the work items are given to each team member, who is then charged to finish as many of them as quickly as possible. As a result delays occur in the whole process when the next member in the line is overloaded or has a problem with his/her work. The reason behind that is that traditional development work is in the form of a chain in which one team member's work item is handed over to another.

However, with Kanban things work in another way. It promotes a pull system Instead of pushing work items. Each member of a team has one item to work on at a time. When he/she finishes it, he/she will automatically pull the next item to work on. Kanban aims to communicate priorities, provide visibility to the software development process, and highlight bottlenecks, resulting constant flow of releasing work items to the clients, as the developers focus only on those few items at a given time. Fig. 3 bellow shows the typical structure of a Kanban board and its principles in action.

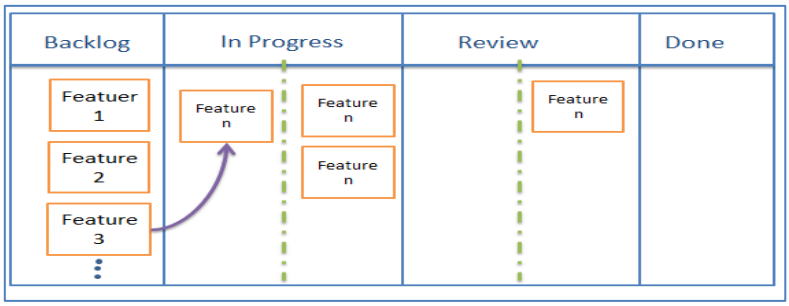

Fig. 3. Example of one implementation of Kanban board used in our software development project.

Kanban has five core principles: visualise workflow, limit work in progress, measure and manage flow, make process policies explicit, and use models to recognise improvement and opportunities. Kanban principles are applied using a board that visualises the flow of activities of the process in various columns. Cards are used for each working item on the Kanban board to show its current state. The flow of work items through the process is optimised by limiting the work in progress in each activity column to a maximum number of items that can be pulled into the column. In this manner, the team effectively visualises their workflow, limits work in progress items in each stage, and monitor the cycle time from start to finish [2].

3.3. Positive Correlation between Dependency and Communication

Ref. [5] recognized the relationship between communication and dependencies. He concluded that by reducing dependencies, it is possible to reduce developers' dependencies on one another, creating a managerial advantage. A breakdown in communication efforts constitutes a major problem in software development streamlining the process.That is why software engineers have recognized the need to deal with dependencies between components and software team. The large number of dependencies among different software artefacts and the dependencies among activities in the software development process and are one of the reasons for these problems [5].

4. Distributed Software Teams Based on Kanban Methodology

With the entrance of Kanban to the software engineering field in 2004, new principles showed up in the literature review section. Kanban is looking for continues delivery, team communication improvement, rapid software development, improving the flue of value and optimizing cycle time. This section presents all steps taken in designing and conducting the case study [24].

4.1. Research Approach

At first stage, Kanban has been implemented in a three software development projects. Each project is developed through a geographically distributed team. The second stage was A Kanban survey that was conducted among software development teams. The survey was conducted using an Internet survey tool. The survey included questions on different Agile methodologies usage, and if team member is willing to use Kanban, to decrease the challenges faced adopting any other method.

5. Conducted Kanban Design and Results

The key results in this section briefly presented from the study. Firstly, the analysis of the questionnaire responses will be reviewed. Secondly, Kanban use case findings will be discussed.

5.1. Questionnaire Results

Based on the previous questionnaire (Table 1), 50 responses were analysed based on team members during the development process. The table below presents the responses:

Table 1. Present the Questionnaire Ranking Response

Other Agile practices impact team communication	Ranking				
	1	2	3	4	5
Dependencies: -Waiting for long time				×	
- Dose Dependence diminishes over time				×	
- Rearrange work difficulty			×		
Communications: -un-needed communication					×
Meetings: -a lot of formal meetings				×	
- daily meeting are boring					×
- Dose meetings affect your motivation and creativity				×	
Visualization: -degree of understanding the work process		×			
- What do you think of the work flow being on board		×			
Bugs: -degree of finding bugs			×		
- degree of locating a problem			×		
Information:-degree of finding information			×		
Changing requirement: -degree of changing requirements					×

The results reflect that there is a high percentage of software developing team members faces some obstacles when adopting Scrum as agile practices. Responses show that Scrum is the most popular agile practices used; although Scrum and Scrumban are used for many years, teams struggle(70% using scrum) from getting to information needed easily(30% using Kanban), as developers get bored from informal stand-up meeting every morning (90% scrum, 10% Kanban)and their motivations of continuing their works is no more available. Stakeholders confirmed that for new initiative the business needed an approach to speed-up the delivery of individual business tasks as they agreed that Scrum would not serve. In addition, frequent changes in the middle of the sprint often require significant re-planning efforts. (Kanban achieves 80% flexibility in responding changes than scrum 20%) Stakeholders mentioned that their initiative required quite a bit of flexibility on the team's part because they anticipated that the priorities could change

every two or three days.

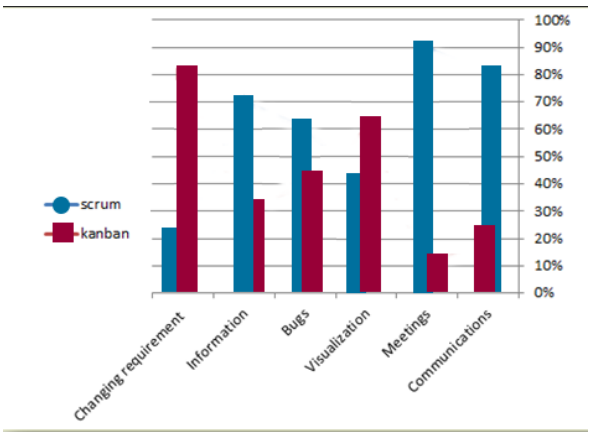

Fig. 3. Present the questionnaire response, and how welcome the use the Kanban for its benefits.

Finally yet importantly, if the business stakeholder requests the Scrum team to deliver some features ASAP in the middle of the sprint, it can result in another conflict because the Scrum team is working on a plan to deliver working increments by the end of the sprint. This is why some Scrum teams use the "red line" to increase flexibility, but this workaround may lead to a significant drop in performance. Finally, without extensive communication and work on soft skills we may not achieve the expected performance boost. In the case of distributed (not co-located) there is the potential risk of getting stuck in the very first "forming" stage forever, because the next stage, "storming," is hardly possible without being physically present in one location with other team members. As some added that effective, Distributed Scrum is possible only when the customer is ready to invest in communication tools and cross-location business trips (intensive communication in scrum 80%).

5.2. Kanban Use Case and Results

Regarding to this case study we were able to confirm that it is better to use Kanban in globally distributed developing project. Gaining benefits such as; knowing wher is the issues and solve it, catching improvement and identifying benefits of Kanban usage. Thinkloud is a small software development company (3 teams,each of 5 members) with their headquarters in Egypt. The company focuses on web development projects working in a distributed environment. Fig. 4 bellow represents the software development distribution of each of the three teams along with distances between the sitesand several locations like in Africa, NorthAmerica, and Asia. Scrum as an Agile method was the adopted practice to manage the project flow. Almost all participants had Scrum or distributed software development experience than just this project.

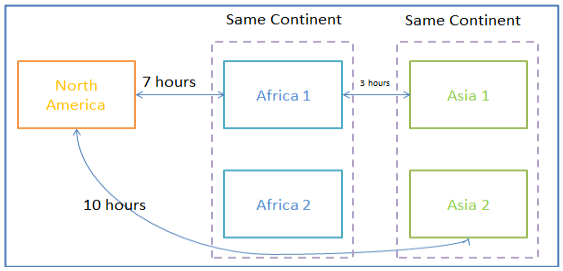

Fig. 4. The software development team along with distances between the sites.

The Web development project has been on going about 6 months in parallel with this study. Before

that,there was a development project that had been postponed to unknown time due to choose the wrong practice. As a result, the previous project stopped and we lost one of the developing team. Therefore, between the previous development project and the current project, there was a change in the used practice, in which there was a shift from Scrum to Kanban. The web site was developed with using the following tools; a Git based version control (Bitbucket) that is web based hosting service for projects that use Gitas a revision control systems, a JavaScriptIntegrated Development Environment(IDE) for complex client-side development and server-side development (WebStorm) with Node.jsweb development, an issue and project tracking tool (Jira), to track any kind of unit of work (that might be an issue, bug, story, project task, etc.) through a predefined workflow. Participants had earlier experience of those tools andevery developer had at least 5 years of experience at the web development. Due to Scrum, the used principles before adopting Kanban where created in the use cases where created in the backlog. Depending on the product owner, tasks were prioritized. Sprints were applied, which ended in time limited of two weeks. Retrospectives were held regularly at the end of each sprint. After the sprint reviews, the whole team participate for hunting bugs. At last, all changes were deployed to production. Team meetings were held via Skype video vedio. All meetings were distributed and everyone was participating from own computer from any location.

That did not last for long, after the two sprints the developing team did not know where the product was going. There was a lack of motivation because the boring daily meetings that developer start getting lazy to attend it. Only 40% (Jira charts) of the tasks of the Sprint were done, due to dependencies. As a result, unfinished tasks were shifted to another sprint with the new stories, and the developing team is no close any more to the predicted cycle time for the project. Scrum is very strict, product owner is not allowed to modify any task or change any its priority during sprint. There for task might be accomplished but not as the client needs, or time might be spent on task that is no more needed. According to Scrum to time box bulk of stories that need to be done successfully gives the 100% certainty that there will be no changes from the customer putting. Furthermore, Scrum gives stress on the developer due to what the big number of tasks.

With Kanban, things were different and exciting. Playing Kanban increased the team's enthusiasm due to the Kanban philosophy. Same steps were taken to create stories in the backlog. Electronic Jira was used as a task tool and its Kanban Board across continents to visualize and manage projects. Jira Boards can be as simple or as complex as its needed: for this project, the board was ranged as follow: To Do – In Progress – Review –Done model.Inside Jira Kanban Board, cards are used to capture ideas, or to organize tasks. Cards can also be linked to each other, or to other boards. Cards can contain chat about ideas or tasks; right on the card itself, as a result it gives the opportunity for developers to participate to share ideas specially in solving bugs. Conversations were pushed to team members as emails, or they remain forever connected to the card making it easy to retrieve even months after a board has finished. Every card has its history contained within it to be able to see who did what, and when in a glance.The product owner can prioritize cards in the backlog any time while developing. In General, high-value stories are placed at the top.Doing this will signal the team members that the task is now important and needs immediate attention. Next phase is the In Progress column. Any team member can pull a task card from the first column. Previous agreement was done between team members to assign who is chosen to pull and work on a user story. The information on the cards in the In Progress column illustrates the task,the name of the team member currently working on it, product owner name, the date on which the task was ordered, and when the task development started. AWIP limit off our assets on the In Progress column, so each team can pull a maximum off our features from the back logs. Each team member works on the task to prepare the user story. Once the user story has reached the Done state on the Jira Kanban task board, discussion with product owner occur. Each team member pulls the approved card until it is released after passing through the Review column. Finally,the

cards reach the Release phase. For measuring the flow,average lead-time is used at the organization on all levels. Those measurements are represented on Jira various reports presented in charts as the Cumulative Flow Chart, control chart and the velocity charts. Those charts Shows the statuses of issues over time, track the amount of work completed and calculate the cycle time for the project. This helps to identify potential bottlenecks that need to be investigated, providing team members with the information they need.

Table 2. List of Challenges and Possible Solutions

Challenges	Positive Solution using Kanban
decreasing dependency between collaborators	Reduce waste : 1-eliminate un-needed communication (daily meetings) 2-minimize delay waste by the WIP limits that should be designed by the team to reduce such effect.
decreasing dependency between tasks themselves	This is minimized by techniques of splitting user stories to be as independent as possible and working on small features.

Distributed Kanban is less problematic than distributed Scrum. Scrum needs good investment in effective communication for faster delivery and better productivity for the daily face-to-face meetings. As a result, using Kanban for decreasing communication resulted in decreasing dependencies by 27%, and this has led to raising the collaboration effectiveness, and reducing waste. Leads to a reduction in delays and getting close to the predicted cycle time. So minimizing the cases of,developer waits for the analyst/designer, and tester waits for the developer. In Kanban, this is minimized by the WIP limits that should be designed by the team to reduce such effect. Dependencies between tasks themselves also are solved using Kanban, so it is minimized by techniques of splitting user stories to be as independent as possible. Table II shows a list of challenges and possible solutions.

6. Conclusion & Future Work

As development becomes increasingly distributed, little is known about how the loss of rich opportunities to interact influences the development cycle.With Kanban developing team works on delivering features no more place for unneeded communication with daily boring meetings or sprints that hold bulk of tasks that puts stress on the developer, customer interventions are not welcomed in the middle of the sprint.Compared to the usage of the widely used Scrum method, The usage of Kanban in managing the geographically distributed team has resulted in 98.8% of the stories have been done, 30% Increase in customer satisfaction, and reduction of the distributed team's dependency reduction by 27%.Future studies are important and needed to explore Kanban uses with all its benefits and usages in the software development field. Further studies could be conduct on applying Kanban on large scale within global software development companies.

References

[1] Komi-Sirviö, S., & Tihinen, M. (2005). Lessons learned by participants of distributed software development. *Knowledge and Process Management, 12(2)*, 108-122.

[2] Conchúir, ÓE., *et al*. (2006). Global software development: never mind the problems–are there really any benefits?

[3] Trickett, P. K., Noll, J. G., & Putnam, F. W. (2011). The impact of sexual abuse on female development: Lessons from a multigenerational, longitudinal research study. *Development and Psychopathology, 23(2)*, 453-476.

[4] Korkala, M., & Maurer, F., (2014). Waste identification as the means for improving communication in globally distributed agile software development. *Journal of Systems and Software, 95*, 122-140.

[5] Aad, G., *et al*., (2010). The ATLAS simulation infrastructure. *The European Physical Journal C, 70(3)*, 823-874.

[6] Kankanhalli, A., Tan, B., & Wei K., (2007). Managing conflict in global virtual teams. *Journal of Management Information Systems, 24(3),* 237-274.

[7] Schlenkrich, L. & Upfold, C. (2008). A guideline for virtual team managers: The key to effective social interaction and communication. *Proceedings of the 2nd European Conference on Information Management and Evaluation.* Academic Conferences Limited.

[8] Tihinen, M. Measurement-based management of global software development projects.

[9] Mahnic, V. (2012). A capstone course on agile software development using Scrum. *IEEE Transactions on Education, 55(1)*, 99-106.

[10] Ikonen, M., *et al*. (2011). On the impact of kanban on software project work: An empirical case study investigation. *Proceedings of 2011 16th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS),* IEEE.

[11] Venkatesh, V., *et al*. (2003). User acceptance of information technology: Toward a unified view. *MIS quarterly,* 425-478.

[12] Jalali, S., & Wohlin, C. (2010). Agile practices in global software engineering-A systematic map. *Proceedings of 2010 5th IEEE International Conference on Global Software Engineering (ICGSE).* IEEE.

[13] Herbsleb, J. D., *et al*. (2000). Distance, dependencies, and delay in a global collaboration. *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work.*

[14] Hossain, E., Bannerman P. L., & Jeffery, D. R. (2011). Scrum Practices in Global Software Development: A Research Framework. *Proceedings of 12th International Conference on Product-Focused Software Process Improvement.* Springer.

[15] Mintzberg, H. (2004). *Managers, Not MBAs: A Hard Look at the Soft Practice of Managing and Management Development.* Berrett-Koehler Publishers.

[16] Salas, E., Sims, D. E. & Burke C. S., (2005). Is there a "Big Five" in teamwork? *Small group research, 36(5)*, 555-599.

[17] Hole, S. & Moe N. B., (2008). A case study of coordination in distributed agile software development. *Software Process Improvement,* Springer. 189-200.

[18] Stober, T., & Hansmann, U. (2010). *Best Practices for Large Software Development Projects*. Springer.

[19] Bannerman, P. L., Hossain, E., & Jeffery, R. (2012). Scrum practice mitigation of global software development coordination challenges: A distinctive advantage? *Proceedings of 2012 45th Hawaii International Conference on System Science (HICSS)*. IEEE.

[20] De, S. C., *et al.* (2008). *Exploiting the Relationship between Software Dependencies and Coordination through Visualization*.

[21] Flora, H. K., Wang, X., & Chande, S. (2014). Adopting an agile approach for the development of mobile applications. *International Journal of Computer Applications, 94(17)*, 43-50.

[22] Scheer, A. W. (2012). *Business Process Engineering: Reference Models for Industrial Enterprises.* Springer Science & Business Media.

[23] Ahmad, M. O., *et al*., (2015). This is working document. Final version will be ready after 15 th November. *Proceedings of the Tenth International Conference on Software Engineering Advances ICSEA 2015*.

[24] Cottmeyer, M. (2008). The good and bad of agile offshore development, *Proceedings of AGILE'08. Conference in Agile.*

Maha Khaled Yacoub *was* graduated from Helwan University. She is a postgraduate studies. She recived her higher diploma in information system from Helwan University, in 2010. She recived her Ms.C (ongoing) from Arab Academy for science and technology. and a research interest in software engineering. She is a demonstrator in faculty of commerce and Business Adminstration- Helwan University in Cairo, Egypt. at Helwan University at Business Information System.

Mostafa Abdel Athim Mostafa was graduated from MTC. He recived his BS.C in computer engineering with excellence degree 1980. he recived his Ms.C from Cairo university in 1985, and a Ph. D from Cranfield University in UK in 1992. He research interest in software engineering, Internet of Things and cloud computing. He promoted to full professor in computer science 2004 and He is now work as vice president of Arab Academy for science and technology.

Ahmed Bahaa Farid is a professor in faculty of computers and Information- Helwan University in Cairo, Egypt. Moreover, he is an ISO committee member. Ahmed is a winner of the world-class Microsoft Most Valuable Professional (MVP) technical award in software engineering track (under different names) for the last ten years in a row. Additionally, he is a member of the Microsoft Regional Directors group. He has been invited to contribute as a speaker, and expert in many technical events including and not limited to; Microsoft ALM Summit 3 (Microsoft Redmond, WA), Microsoft TechEd 2010 (Berlin, Germany), and Microsoft TechEd 2011 (Dubai, UAE). Ahmed has a set of research papers that are published in IEEE explore and Thomson Reuters Indexed Journal.