An Analysis of Kanban as a Project Monitoring Tool in Undergraduate Courses

Heydi Miura Machado
Undergraduate Program in System Analysis and Development
Federal University of Technology - Paraná
Cornélio Procópio – PR – Brazil
heydimiuramachado@gmail.com

Alexandre L'Erario, Alessandro Duarte
Department of Computing
Federal University of Technology - Paraná
Cornélio Procópio – PR – Brazil
{alerario, duarte} @utfpr.edu.br

***Abstract*—Project monitoring poses a relevant role for project success and it is considered more fundamental in a distributed environment since it confronts additional risks such as culture, geographical, and temporal barriers. Recognizing the complexity and importance of a distributed project, it is vital that undergraduate students receive practical experience and are trained to face the obstacles this form of development imposes, as well as be able to monitor the project progress. The challenge is to take all of these concepts and apply them to a technology course that will not only provide a mechanism for teachers to monitor the evolution of students' projects but also expose students to a production environment for the application of Kanban in monitoring projects in a distributed environment. To validate the efficiency of the technique introduced in this paper, an experimental test was carried out with undergraduate students from different careers, skills, cultures, personalities, and professional experiences to enhance the obstacles encountered in a real distributed environment. The outcome demonstrates the students are partially able to monitor their project and based on the results, some recommendations were made for the use of the Kanban tool in project monitoring during the education process.**

I. Introduction

In order to mitigate the consequences of globalization, coupled with intense competition, information technology (IT) companies make use of strategies in software development for business progress [1]. One of the most used tactics by technology companies is Distributed Software Development (DSD) [2] . This methodology was widely disseminated as a response to the promising results presented to companies in reducing costs, accessing highly skilled labor[1], [3], [4], increasing productivity and profitability, as well as quality improvements [5]. However, distributed development brings along with its numerous benefits several other complications as side effects [6]. It is widely believed that DSD is riskier and more challenging than co-located development [7].

In this context, Wiredu [8] notes that as monitoring distributed projects becomes more complex, it also becomes more fundamental. According to Scharff [9] and Souza [10], software process monitoring enables an increase in the speed and quality of development , mitigates risks, and improves the relationship between employees and the customer. In contrast, the absence of a monitoring model that can measure the progress of the project may cause negative consequences, such as: low productivity, rework, customer dissatisfaction, unnecessary activities and even an increase in costs.

Several studies and researches in the software development field point to the planning and monitoring of projects as one of the main key success factors. In a survey conducted by Jenkins [11], in which he interviewed 72 software developers, he reported the average effort was overrun by 36%, the average schedule was overrun by 22%, and cost was exceeded by approximately 33%. Overwork triggered an increase in cost, which in turn, could lead the project to failure. In another study conducted by Jhonson [12], he concluded that in the year 1995 American companies spent approximately $ 59 billion on project overheads, and another $ 81 billion on canceled software projects. "All of these sobering statistics suggest a strong need for better project tracking and control in both academia and industry" [13].

The difficulty of monitoring projects is a long-standing problem that has been discussed and studied by several authors. Hence, by verifying the increase of distributed projects and the vital importance and impact that project monitoring has on the software development, the educational aspects addressed in this project is that undergraduate students should be taught and trained to develop the necessary skills to supervise a project along with dealing with DSD obstacles. With this in mind, this paper is aimed at the study, test, and development of a monitoring model using Kanban and Business Process Model and Notation (BPMN) with undergraduate students.

The paper structure is the following: Section II, briefly introduces the literature review. Section III, presents some related works. Section IV, presents the method and procedure adopted. Section V, describes the experiment execution, Section VI and VII talk about the outcomes and conclusion respectively.

II. LITERATURE REVIEW

In order to understand the conceptual foundations of this work, this section briefly explains what Distributed software Development, Project Monitoring, Scrum, Kanban, and BPMN are.

A. Distributed Software Development

According to L'Erario [14] DSD occurs when several websites cooperate and/or collaborate to develop the same product

978-1-5090-5920-1/17/$31.00 ©2017 IEEE

or part of it. Large distribution capacity is only one of the factors in DSD. In agreement with Herbsleb, [3], Dastidar [2], and Scharff [9], the benefits of DSD are related to cost reduction, productivity increase, access for skilled labor, proximity to the market, efficiency and practicality.

However, distributed development and its physical separation among project members has diverse effects on many levels [1]. Several authors have discussed the challenges that DSD poses. Hersleb [1] presents basic difficulties to distributed development cultural problems, inadequate communication, synchronization, and technical issues. Carmel [4] inferred two critical challenges of DSD: coordination and control, caused by the deficiency in communication.

B. Project Monitoring

Monitoring means capturing, analyzing, reporting, and communicating project performance. Monitoring also compares the actual measures of activities progress with the planned measures undertaken prior to the start of the project [15]. Project monitoring is carried out by measuring the progress of activities, recorded in note form, on the schedule itself, directly on the paper, or in the tool [16],[10].

Elements that are monitored and controlled throughout a project are key to ensure everything runs smoothly [10]. In project monitoring there are several perspectives that can be supervised: it is possible to monitor the integration, product quality, cost, risk, scope, schedule, communication, and the activities of a project, among other things [17].

C. Scrum

Scrum is an agile method based on an iterative and incremental software development that adopts a work break-down structure technique [18]. It is a framework which focuses on customer satisfaction, team work, easy adaptation to changes in software, flexibility, faster delivery of working product over documentation, feedback, and inspection [18], [19].

There are three important terms that are frequently used in scrum development: scrum team, product backlog and the sprint. The scrum team consists of a product owner, a development team, and a scrum master [19]. Product backlog is a set of requirements to be converted into sprints sized based on complexity, days, or other unit of measure [20], [21]. In Scrum, the software product is partially delivered in the series of iteration or increments within a predefined period (typically 30 days) called sprint [19].

D. Kanban

The Kanban approach was introduced by the Japanese manufacturing industry in the 1950's [22]. It is a simple scheduling system that regulates production through the use of an instruction card sent along production line.

In software development, Kanban is considered much more than a scheduling system. It enables you to visualize the workflow, limit the progress of the activities at each stage of the flow according to the capacity of the team, measure the cycle time, identify the problems, and maintain a constant workflow [23],[22]. It also guarantees the visualization of the process, once it is able to identify the work assigned to each developer and clearly state priorities and constraints.

According to Ahmad [22], the use of Kanban improves product delivery time, consistency, quality, communication, and coordination among team members and decreased customer complaints.

E. BPMN

In the monitoring context, BPMN is another modeling tool that has been gaining in popularity [24],[10]. It is a graphical notation of business process modeling and orchestration similar to the flowcharts and activity diagrams used in the Unified Modeling Language (UML).

According to Souza [10], the advantages of this tool are due to its ease of use and understanding, likewise its ability to model complex business processes. Its main objective is to provide an easy to understanding graphic notation for all those involved in the business process: from business analysts who create the first outline processes, to developers, implementers, as well as business people who will manage and monitor them [25],[10].

III. Related Works

The first related work (RW1) found is a study conducted by Collofello in 1999 [13]. At that time already, he realized the impact of monitoring on project success and the importance to teach and train undergraduates about project management through a real experience in academia. In his article he reports the challenges posed by the learning process as well as work group causes many academic software engineering teams to struggle with completing their projects on time.

Collofello [13] proposed a model that would teach students to monitor their projects and supervise both the teams progress and individual members. The system consists of individual weekly progress reports, and frequent incremental deliverable from individuals and teams. This way, the instructor was able to view these reports likewise identify students and teams that have not submitted required reports on time.

In 2015, Souza [10] investigated difficulties in monitoring distributed environments and the effectiveness of Kanban and BPMN as monitoring tool (RW2). She proposed a model for work organizations that assisted with supervising and controlling the progress of a DSD project through Kanban and BPMN.

Although her project aimed to develop a monitoring model for active organizations, to validate her project she selected undergraduate and graduate students, and also IT professionals. With the systematization of the collected data, she concluded it is possible to monitor the schedule of the activities with the proposed process. In addition, she reported the importance of a monitoring tool for the project success, since through the monitoring process managers were able to identify the risks and determine the appropriate course of action to address the problem, in other words, to control the project.

As it was presented, project monitoring has been studied for a long time from different perspectives: two of them being educational and industrial. However, the analyzed works present a set of limitations. For a better understanding, Table I exposes the comparison between the related studies and the current one:

TABLE I
COMPARATIVE ANALYSIS OF RELATED WORKS AND THE PRESENT PROJECT

Ref	Monitor Project	Monitor Students	Covers DSD
(RW1)	YES	YES	NO
(RW2)	YES	NO	YES
Present work	YES	YES	YES

IV. IV. METHOD AND PROCEDURE

A. Research Method

In the present project, the method and procedure adopted is the experimental test that is similar to the experimental method. The choice of method is due to the fact that it does not aim solely to validate a hypothesis but also to identify variables that alter the results of the contextualized test, besides clarifying concepts that have not been consolidated yet.

The hypothesis that promotes this experimental test is that the knowledge obtained in undergraduate courses is sufficient for students to be able to monitor their own projects and control them in order to comply with the proposed schedule. And that the use of Kanban as a monitoring tool is of great relevance for students to succeed in their distributed projects.

It is important to point out that the participants selected for the experimental test are undergraduate students who were experiencing for the first time the development of a distributed project and were also learning the technical content of the discipline, at the same time that the monitoring process was being executed. The outcomes therefore would be different if the same approach were conducted with groups of professionals already trained and experienced in the field.

B. Planning

It was defined that the test would select individuals in the process of learning and technical growth. These participants would be undergraduate students from Federal University of Technology – Paraná (UTFPR), Cornélio Procópio campus, with backgrounds in Computer Engineering and Systems Analysis and Development.

In all, 25 students took part in the experimental test. They, were allowed to organize themselves and delimit their teams according to their affinity, in total 7 groups containing 2 to 6 members per group.

The students could choose the subject matter of each development project, as long as it was within the established parameter, be a distributed development project, which was not a problem as the majority of students lived in different cities.

The students were recommended to use GitHub as a version control tool, an online Kanban tool for monitoring their own project, and Scrum as development framework. For the analysis of the experimental test and for the evaluation of the raised hypothesis, the following elements would be monitored:

- Communication
- Coordination and synchronization of activities
- Creation and use of the Kanban framework
- Use of the version repositories and the artifacts made by the groups.

V. EXECUTION

The monitoring model was applied in an experiment performed in the second semester of 2016. This experiment simulated a DSD environment which consisted of two basic processes: one of which covered preparation of basic documentation (Requirements gatherings, schedule, and diagrams), and second the development of a software in Java language.

The proposed monitoring model consisted of mapping the processes described above in activities using BPMN notation. For this, it was defined that the groups would follow an established workflow that was organized into four phases:

A. Preparation

The project execution lasted approximately one semester. In the preparation phase the students were responsible for organizing themselves into groups, defining their respective project themes and collecting the necessary data for its development.

B. Planning

Early in the phase, the participants were instructed to define their product backlog, sprints and plan their projects incrementally to schedule its activities based on the proposed period.

C. Development

The development phase is the performance of activities that involve the execution of sprints, design and tests.

D. Delivery

Considered as the final phase of the project, it covers activities related to the changes and delivery itself.

The BPMN model used in the experimental test can be visualized in Figure 1.

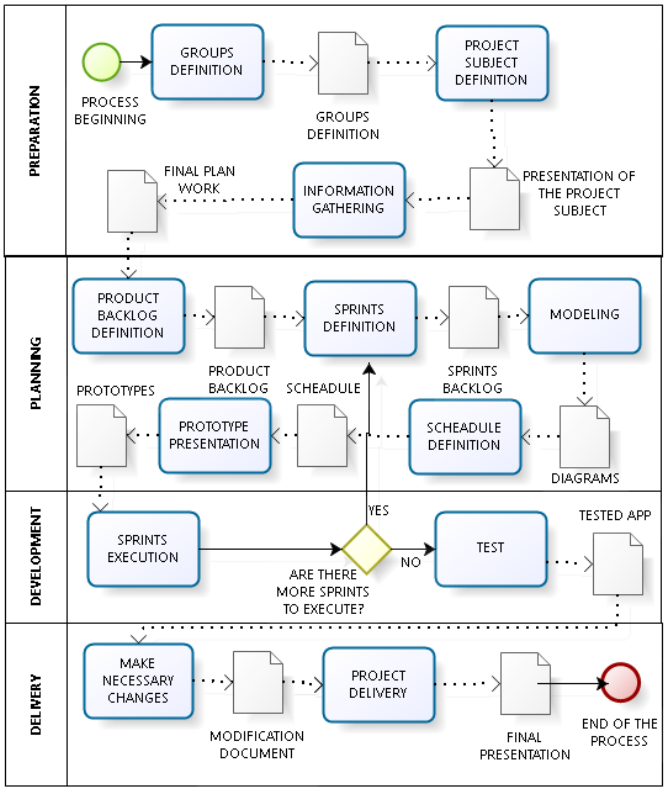

Fig. 1. BPMN Development Process

VI. Analysis and Results

It was possible to note that there was a very large heterogeneity among the intergroup participants. It was observed that the members acquired different degrees of experience and knowledge, often caused by the different focuses of the careers in which the respective students were enrolled, and even which semester they were in. Also, measurable differences in the students' academic curriculum were noticed, varying from the number of completed subjects, failed classes, and to whether or not students obtained education abroad.

Such diversification favored the simulation of a real distributed experience, since the heterogeneity between the groups accompanies the miscegenation encountered in DSD, related to the degree of knowledge, different cultures, experiences, and personalities. The heterogeneity among the participants justifies the great diversification in the way the Kanban board was created. Some groups, for example, have guided the Kanban board to be artifact driven, while others have monitored the activities.

Beyond the different approaches used to build the Kanban board, there was also a difference in the way it was monitored. Some groups tried to use all the features that the chosen tool provided, such as: attaching artifacts, specifying and assigning tasks to members of the team, using comments in activities, tags to group them, setting delivery dates for each activity, etc. While others, only defined the activities to be carried out.

The Table 2 presents how the teams structured their Kanban board.

TABLE II
Comparison Between Kanban Boards.

Management of the Kanban board	T1	T2	T3	T4	T5	T6	T7
Used Kanban Board	Yes	Yes	Yes	Yes		Yes	Yes
Activity-oriented Kanban board	Yes	Yes		Yes		Yes	Yes
Artifact-oriented Kanban board			Yes				
Artifacts attached in kanban board			Yes				
Specification of group member responsible for performing the task		Yes		Yes		Yes	Yes
Only one member of the group managed the kanban board		Yes	Yes			Yes	Yes
Comments on activities/artifacts			Yes				
Detailed task explanation	Yes		Yes				
Subdivided tasks into small activities	Yes	Yes		Yes			Yes
Set delivery date for activities/artifacts		Yes				Yes	
Activities/artifacts defined according to each sprint		Yes					

The number of activities defined per group was another divergence. Some teams have segmented their activities into small tasks to be performed, thus generating a greater number of artifacts and consequently a greater vision of project monitoring and progress; others have segmented their activities into large and laborious tasks to be developed.

Using version control tool, it was possible to check the delivery frequency of the groups' activities and the number of commits performed by them. With the table below it is possible to identify the data collected from each group.

TABLE III
Execution Data Collected from the Groups.

Teams	Number of Members	Lines of Code Developed	Number of Activities	Number of artifacts
Team 1	4	1143	22	2
Team 2	4	1480	29	15
Team 3	4	1091	11	8
Team 4	6	1547	25	25
Team 5	3	852	0	15
Team 6	2	436	6	0
Team 7	2	464	20	4

By the commit frequency of each group and the Kanban board, the professor was able to monitor the development and progress of each team. Also, to identify the groups that accomplished the schedule proposed and the ones that were having trouble to hand in the artifacts on time. Thus, through this analysis the professor was able to intervene in the projects and help the students to analyze their problem and identify possible corrective actions.

From the professor's point of view, the data generated by each of the teams were encapsulated. The teacher could easily get an instant snapshot of the project progress and its partial deliveries. The performance of each one of the teams was usually supervised. However, monitoring the individual performances of each member in the groups was more difficult. In this case, the professor was needed to relate the adopted process and the pushes made by the version control tool.

The information generated by GitHub was used by the professor to monitor individual activities. However, they were confronted with the process outlined by the team with the purpose of verifying if the data was really pertinent to a given user.

In general, the main detected impasses were related to new technologies that teams needed to use in their projects. Subjects correlated to the course did not pose as many bottlenecks as external technologies. For example, the difficulty the students encountered when trying to manipulate the additional Ajax frameworks and the communication with other systems via rest or web service. There was also a team that performed authentication on Google and added elements to the calendar. In this case, there was an excessive demand of time to dominate the technology and consequently a delay in some artifacts delivery.

This procedure applied by the professor, not only identified the need to monitor the evolution of projects, but also possible pertinent subject matter that could be addressed in the course.

Four of the seven groups, despite using the Kanban tool for monitoring the project were unable to succeed (to follow the planned schedule). Three of them accomplished the schedule proposed (Team 2, Team 4, and Team 5). One of the three teams (Team 5), did not use the Kanban tool for monitoring the project's activities. According to the members of the team, a monitoring tool was not necessary. Tracking the timeline, delivery dates of activities, and artifacts were sufficient to monitor the project.

It is believed that it was possible for the students to monitor and comply with the schedule due to the fact that the project was small. In a real distributed development environment, with considerably larger and more complex projects, overseeing a project and being able to accomplish the schedule without a monitoring tool would clearly be more difficult, if not impossible.

Through an analysis of the Kanban boards between the two groups that completed the project according to the schedule and those who were not able to complete it, some key factors in project monitoring were determined:

- The breakdown of activities into small tasks simplified the monitoring of the project's progress for students, since it was possible to measure and estimate more accurately the time spent to complete each activity and to visualize the ones still pending.
- Another element that was very significant for the project monitoring was the number of artifacts generated. As students performed the defined BPMN process, the artifacts were created, and thus, the greater number of artifacts created, the greater the project's progress.
- Using Kanban tool features, such as setting the delivery date, was a very important feature for checking delays. It was possible to verify the time each activity was delayed according to the schedule and activities still pending.
- Defining the individuals responsible for executing each activity was another significant feature, since the members could identify the amount of activities established for each one, were able to self-monitor and measure progress themselves.

VII. Conclusion

Thus, based on the results obtained in the experimental test and the analysis of them, it is concluded that the following hypothesis: "Are students in a learning progress able to monitor their DSD project?" is partially true, considering the presented results produce 3 kinds of outcomes: groups that were able to supervise their projects using Kanban and BPMN, groups that did not use these monitoring tools and were also able to supervise their projects, and those that did use these tools and were not able to accomplish the proposed schedule. Therefore, it is concluded that individuals restricted of professional experience and that are in progress of learning and technical growth, partially can monitor their projects.

According to the results obtained and its analysis, the findings compose highly important features of Kanban for monitoring distributed projects, mainly for those that are in the process of learning. The following is a set of recommendations that are believed to be of great relevance in the monitoring training process for undergraduate students.

- It is recommended the Kanban board be activity oriented rather than artifacts
- Activities should be broken in the most unique and accurate way possible
- Define the responsible for the performance of each task
- Set delivery dates for the activities

The construction of the Kanban board, along with a specified process through BPMN, assisted the teacher in supervising the teams. Although getting individual student data was a more complex activity, the teacher was able to monitor the group as a whole. Deadlocks were commonly detected and culminated in delays on the Kanban board.

References

[1] J. D. Herbsleb and D. Moitra, "Global software development," *IEEE Software*, pp. 16–20, 2001.

[2] S. G. Dastidar and S. Chatterjee, "Distributed software development: Experience and recommendation," *IEEE Software*, 2013.

[3] J. D. Herbsleb, "Global software engineering: The future of socio-technical coordination," *Coordination. IN: Future of Software Engineering (FOSE)*, 2007.

[4] E. Carmel and R. Agarwal, "Tactical approaches for alleviating distance in global software development," *IEEE Software*, 3 2001.

[5] R. Prikladnicki and J. Audy, "Distributed software development: Toward an understanding of the relationship between project team, users and customers," *IEEE Software*, 2003.

[6] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy, "Does distributed development affect software quality? an empirical case study of windows vista," *IEEE Software*, 5 2009.

[7] F. Ciccozzi and I. Crnkové, "Performing a project in a distributed software development course: Lessons learned," *Global Software Engineering (ICGSE) 2010 5th IEEE International Conference*, 2010.

[8] G. O. WIREDU, "Coordination as the challenge of distributed software development," *IN: Workshop of Distributed Software Development*, 2006.

[9] C. Scharff, "An evolving collaborative model of working in students'global software development projects," *Workshop on Collaborative Teaching of Globally Distributed Software Development, ICSE, Honolulu, Hawaii*, 2011.

[10] V. F. de Souza, "Processo para monitoramento de projetos distribulídos de software," Cornélio Procópio, 2015.

[11] D. Phan, D. Vogel, and J. Nunamaker, "The search for perfect project management," 9 1988.

[12] J. Johnson, "Chaos: The dollar drain of it project failures," 1995.

[13] J. S. Collofello and M. Hart, "Monitoring team progress in a software engineering project class," 11.

[14] A. L'Erario and et. al., "A distributed software development environment dynamics model," 2012.

[15] C. G. von Wangenheim, J. C. R. Hauck, and A. von Wangenheim, "Enhancing open source software in alignment with cmmi-dev," *IEEE Software*, 2009.

[16] F. L. G. Pereira and B. J. G. Neto, "Análise das formas de controle dos processos organizacionais," *Congresso Nacional de Excelência em Gestão*, 2010.

[17] G. PMBOK, *Um guia do conjunto de conhecimentos em gerenciamento de projetos: Guia PMBOK*, 3rd ed., 2004.

[18] K. Schwaber and J. Sutherland, "The scrum guide," 2014.

[19] S. Sharma and N. Hasteer, "A comprehensive study on state of scrum development," *International Conference on Computing, Communication and Automation (ICCCA2016)*, 2016.

[20] A. Mundra, S. Misra, and C. A. Dhawale, "Practical scrum-scrum team: Way to produce successful and quality software," *2013 13th International Conference on Computational Science and Its Applications*, 2013.

[21] M. Cristal, D. Wildt, and R. Prikladnicki, "Usage of scrum practices within a global company," *2008 IEEE International Conference on Global Software Engineering Usage*, 2008.

[22] M. O. Ahmad, J. Markkula, and M. Oivo, "Kanban in software development: A systematic literature review," *39th Euromicro Conference Series on Software Engineering and Advanced Applications Kanban*, 2013.

[23] H. Kniberg. Kanban vs scrum - how to make the most of both. [Online]. Available: www.crisp.se/file-uploads/Kanban- vs-Scrum.pdf

[24] S. A. White. Using bpmn to model a bpel process. [Online]. Available: http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[25] O. M. Group. (2016, sep) Bpmn, specification omg. [Online]. Available: http://www.bpmn.org/