Robotics and Computer-Integrated Manufacturing 43 (2017) 59–67

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Full length Article

A statistical analysis of the effects of Scrum and Kanban on software development projects

Howard Lei *, Farnaz Ganjeizadeh, Pradeep Kumar Jayachandran, Pinar Ozcan

Department of Engineering, California State University - East Bay, Hayward, CA 94542, USA

ARTICLE INFO

Article history:
Received 10 April 2015
Received in revised form
16 September 2015
Accepted 9 December 2015
Available online 17 December 2015

Keywords:
Project management factors for software development
Agile movement
Scrum methodology
Kanban methodology

ABSTRACT

Traditionally, software development processes have relied on the use of the "Waterfall" and "Vee" models. Later, Agile methodologies were used to handle the challenges of managing complex projects during the development phase. Agile methodologies are a group of incremental and iterative methods that are more effective, and have been used in project management. Kanban and Scrum are two powerful Agile project management approaches in software development. The objective of Scrum and Kanban is achieved by optimizing the development process by identifying the tasks, managing time more effectively, and setting-up teams. A review of the literature reveals that there is a lack of statistical evidence to conclude which methodology is more effective in dealing with the traditional project management factors of budget handling, risk control, quality of the project, available resources, having clear project scope, and schedule handling. This research statistically compares the effectiveness of the Scrum and Kanban methods in terms of their effects on the project management factors for software development projects. Numerical analysis is performed based on survey responses from those with experience in the Scrum and Kanban methods. Results suggest that both Scrum and Kanban lead to the development of successful projects, and that the Kanban method can be better than the Scrum method in terms of managing project schedule.

1. Introduction

While project management methodologies have been used for a very long time and date back to the Egyptian era, organizations adopted the methods only half a century ago. During the mid 1900s, the defense, navy, and space research industries were the first to adopt effective project management methodologies to achieve organizational goals. By the early 1990s, with the boom in hardware and software engineering industries, project management methodologies found many takers and have proved effective in helping organizations achieve tremendous results in its products. Adopting one of the project management methodologies made organizations more efficient in terms of planning, setting timelines and budgets, and improving quality of the products that were produced.

By the late 1950s, there have been many trial-and-error methods in managing software development projects. The early methods were used to find better ways of gathering and defining project requirements, analyzing problems, and conducting systematic implementations of problems. Some of the methods were incremental and iterative in nature [1] and others were linear and sequential, known as "Waterfall Model" [2]. The Waterfall model assumes that the team has nearly perfect information about the project requirements, the solutions, and ultimately the goal. Hence, changes in requirements were not encouraged, and became an expensive affair. Nevertheless, the sequence of steps in the Waterfall model is rarely followed in the actual system design [3], and it had become evident that the approach lacked effectiveness in addressing the needs of customers, managing rapidly changing scope, delivery time, and cost of the project [4]. The Vee Process model is yet another system process model that starts with a user need and ends with a completed system [3]. In this model, testing and verification are performed at each stage of the system development, starting with the low-level components and ending with the higher-level components until the entire system has been verified. In the mid-1990s, other software development methods evolved due to problems of these so-called "heavyweight software methodologies," which are complex and require detailed documentation and expensive design [5].

In 2001, the Agile movement was introduced in response to the failures of the Waterfall software development methodology [6]. One of the models based on the Agile movement, known as Scrum, is based on principles of lean manufacturing [6]. A different methodology based on the Agile movement is called Kanban,

* Corresponding author.
E-mail address: howard.lei@csueastbay.edu (H. Lei).

http://dx.doi.org/10.1016/j.rcim.2015.12.001
0736-5845/

which was inspired by the Toyota Production System [7] and by Lean manufacturing [8]. Recently, the Kanban and Scrum methodologies are two powerful methods adopted by organizations for software development. Although there has been a debate for years about which of these methods are preferred, there has not been sufficient evidence based on statistical analysis for selecting a preferred method. The work of Sjoberg et al. in 2012 attempted to quantify the effects of Kanban versus Scrum in a case study with one company, but the authors suggested that their study should be replicated in other environments [9].

The focus of this research is to provide statistical evidence to determine if there is a significant difference between the Scrum and Kanban methods in terms of their effects on different project management factors. The project management factors examined in this work are based on the 6-point star model associated with the Project Management Body of Knowledge [10]. The factors include project scope, budget, schedule, risk, quality, and resources.

In this work, the data is first collected via a web-based survey. Each survey question relates to one of the six project management factors listed above. Next, the correlation between these factors and the success of a project is verified by a Pearson correlation analysis. Finally, Kanban and Scrum methodologies are compared to each other in terms of their effect on the six project management factors. Core results of this work have been previously published and presented at the FAIM 2014 conference in San Antonio, Texas [11]. However, this paper describes the history and backgrounds of the methodologies in greater detail, presents results in greater detail, and provides additional analysis of the results.

This paper is structured as follows: Section 2 presents the history of the Agile movement, and provides background on Scrum and Kanban theory. Section 3 describes the approach and techniques used in this work, and Section 4 provides a statistical analysis and discussion of the results. Section 5 gives a conclusion, and suggests future work.

2. Agile software development methodologies

In this section, the evolution of Agile software development, the creation of Scrum and Kanban methodologies, and their applications, are explained. Figures are also included to illustrate their usage for project development.

2.1. History of Agile software development

Prior to the emerging of “Agile software development,” there had been many trial-and-error-based methods in software development by the late 1950s. These methods were employed to find better ways of defining the project requirements, analyzing the problem, and implementing it in a systematic manner. Some of these were incremental and iterative in nature [1] and others were linear and sequential, known as the “Waterfall Model” [2]. These approaches generally lacked the effectiveness in addressing the needs of customers, managing rapidly changing project scope, delivery time and cost [4]. In the mid-1990s, other software development methods evolved due to the problems of these “heavyweight software methodologies,” which are complex methods with expensive design [5].

As a response to the heavyweight software methodologies, lightweight methodologies, including iterative and incremental methods, had been developed and implemented. Examples of lightweight (Agile) methodologies are Scrum, Extreme Programming, Dynamic Systems Development Method (DSDM), Feature-Driven Development (FDD) and Adaptive Software Development (ASD) [12]. In 2001, these lightweight methodologies had been discussed and the Manifesto for Agile Software Development had been published to define the framework and goals of these methods [13].

2.2. Early history of Scrum

Scrum is a project management methodology for Agile software development that uses iteration and incrementation. It has been designed to manage rapidly-changing project requirements by improving communication between project developers, project owners, and other team members. In 1986, Hirotaka Takeuchi and Ikujiro Nonaka named Scrum the new product development standard in auto and consumer product companies [14]. Scrum has been defined, formalized, and published as the first Agile methodology for software development [15]. In 1993, Jeff Sutherland, John Scumniotales, and Jeff McKenna, at Easel Company, had used Scrum for software development projects for the first time [16]. In 2002, Schwaber and Beedle wrote the book “Agile with Scrum” to describe Scrum methodology [17]. Although Scrum had become a common methodology since then, a study of Agile software development shows that only 3% of the existing scientific evidence on Agile software development focuses on Scrum [18], which this paper aims to address. The following sections describe Scrum methodology, including its implementation process and practices.

2.3. Scrum theory

Scrum, based on empirical process control theory, is an iterative and incremental project management methodology to control risk and optimize the predictability of a project. Transparency, inspection, and adaptation, which are defined below, are three important factors in the Scrum process [19].

Transparency: The process must be visible to everyone who is involved in the project.

Inspection: Scrum users must inspect Scrum artifacts frequently to detect problems in early stages.

Adaptation: If an inspector determines that some aspects of the project are unacceptable and outside of the project scope, the process can be adjusted to avoid further problems.

It should be noted that it is crucial to apply these factors during different project development phases. Details pertaining to these factors are presented in the following sections.

2.4. Content of Scrum

Scrum consists of Scrum teams, events, artifacts and rules. The rules are essential to bind teams, events and artifacts together during the project. They also provide an agreeable structure for resolving conflicts within a project. The following sections explain Scrum team, events and artifacts in detail.

2.4.1. Scrum team

The Scrum team consists of a Product Owner, a Scrum Master, and Development Team Members. Teams are self-organized and cross-functional. Therefore, they have control of the project and know how to accomplish the goals without relying on directions from people outside the team [19]. The team delivers products iteratively and incrementally, maximizing the feedback they receive. Below are descriptions of the different Scrum team roles:

The *Product owner* is responsible for managing the Product Backlog, the list of requirements of the product, and maximizing the value of the project. His roles also include explaining the Product Backlog items and goals of the project to the Development Team, ensuring that the team understands these goals and performs at a high level.

The *Scrum Master* manages the Product Backlog and instructs

the Development Team on creating clear Product Backlog items. The Scrum Master also communicates with the team to ensure that the team understands the long-term plans of the project. In addition, he works with other Scrum Masters to increase the effectiveness of Scrum in the organization.

The *Development Team* is responsible for implementing and delivering the releasable product at the end of each "Sprint," which is a period of time (referred to as a time-box) to create a usable increment of the product. The team controls the implementation of the end-product. Development Team members manage their own work and are self-organized, and are not grouped into sub-teams. The size of the team is an important issue; a small team may suffer from problems in the lack of skill, while a large team may suffer from development complexity. It has been found that the ideal size of a Development Team is seven members [19].

2.4.2. Scrum Events

Scrum uses time-boxed events with project development and project planning phases. Events in Scrum are designed to inspect artifacts and to adapt new methods for solving the project's problems. The goals of these events are to enable transparency, adaptation, and inspection in the development process [19]. Fig. 1 shows the components of each Scrum Event.

The *Sprint* is the heart of the Scrum process. It is a time-box to create a useable, finished product. Each Sprint can be considered as a one-month project with a plan of what needs to be built, and how it needs to be built. The development goals of each Sprint, along with the Development Team, should not be changed during the course of the Sprint. However, the Product Owner and the Development Team may redefine project scope as needed. The Product Owner can also cancel Sprints if any change occurs in the company direction, market needs, or technology.

The Scrum team plans the goals of each Sprint, along with the product's implementation process, in the Sprint Planning Meeting. The overall goal of each sprint is to create a usable and potentially releasable product, known as the "Done" product. The Scrum team members should discuss and have a common understanding of what constitutes a "Done" product. The Sprint Planning Meeting duration is usually eight hours, which occurs once a month prior to each Sprint [19]. In addition to this meeting, there is 15-min daily Scrum meeting where team members update one another on their progress, their future goals for the next meeting, and difficulties they have experienced each day.

At the end of each Sprint, a Sprint review meeting is held to discuss what each team member did during the product-development iteration. This meeting can be a product demonstration to the Product Owner, or occasionally to both the Product Owner and the customers. After the Sprint Review Meeting and prior to the next Sprint, a *Sprint Retrospective* meet is held to inspect how the last Sprint went in terms of communication, human resources, process, and tools, and to identify potential improvements for future Sprints. This meeting usually takes several hours [19].

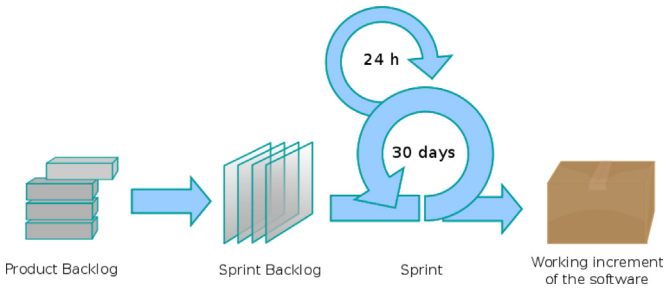

Fig. 1. Components of a Scrum Event. Figure obtained from: Wikimedia.org.

2.4.3. Scrum artifacts

The Scrum artifacts consist of the Product Backlog, the Sprint Backlog, and the definition of what a "Done" product is after each *Increment*, which is the sum of the Product Backlog items completed through the Sprints.

The *Product Backlog* contains the list of requirements, functions, enhancements, and corrections needed in the product. It shows the functionality of the product from technical and business perspectives. The Product Owner is responsible for the creation of the list, and explaining the project perspective to the team. The product backlog is dynamic, as it evolves whenever progress in the project is made.

The *Sprint Backlog* is the list of items in the Product Backlog that are selected for the specific Sprint. The Development Team clarifies the functionalities of the product that will be implemented in the next Sprint, and the work that is needed [19]. During the Sprint, if the Development Team realizes that there is more work required, the team adds the work to the Sprint Backlog. The remaining work in the Sprint Backlog can be tracked by the team in order to manage the Sprint's progress.

Fig. 2 shows an example of a project task board based on the Scrum process. This team uses post-it notes to keep track of the list of tasks during a Scrum Sprint.

2.5. Kanban theory

In the following sections, Kanban theory is presented to complete the background knowledge that this work is based on. Kanban is another project management methodology for software development that emphasizes "just-in-time" delivery. The main focus of Kanban is to accurately state what work needs to be done, and when it needs to be done. It does so by prioritizing tasks, and defining workflow as well as lead-time to delivery [20]. The Kanban process explicitly presents the most important tasks that need the most attention in order to reduce the risk of their incompletion, and also to increase flexibility amongst other tasks in the project.

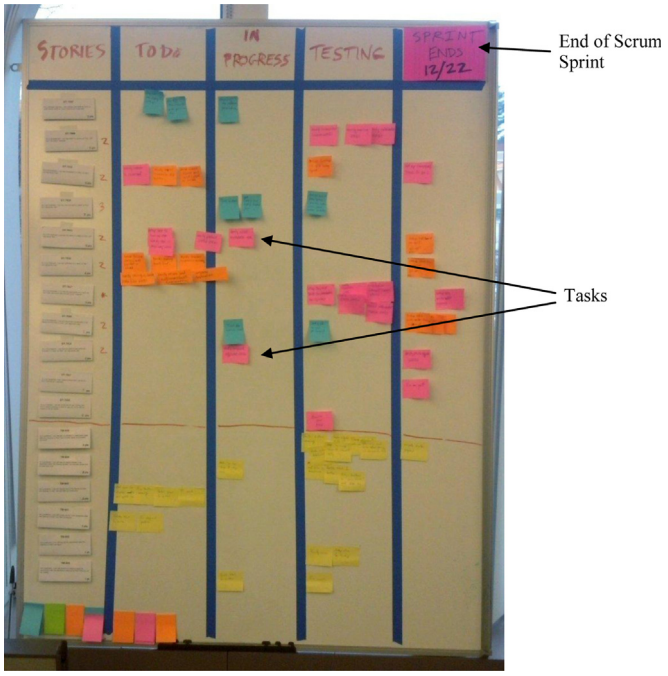

Fig. 2. Scrum task board example. Figure produced by Logan Ingalls (Task board) [CC BY 2.0 (http://creativecommons.org/licenses/by/2.0)], via Wikimedia Commons.

2.5.1. Kanban principles

The following are the basic principles of Kanban for software development:

- Limiting Work in Process (WIP).
- Pulling value through the development process.
- Making the development process visible.
- Increasing throughput.
- Using a fixed backlog.
- Embedding quality.

Kanban methodology focuses on having the right work done at the right time, given the skill sets of the developers. Developers may have different skill sets and work speeds. Project developers start by implementing project components that add value to the project [20]. In addition, project developers do not implement unnecessary features, do not write more specifications than they can code, do not write more code than they can test, and do not test more code than they can deploy. Therefore, Kanban methodology eliminates waste in every step, and is suitable for software engineering projects [20].

2.5.2. Visualization of the workflow

The visualization of the workflow, which shows the progress of the project, is a significant aspect of Kanban methodology. The following sections explain workflow visualization in greater detail.

2.5.2.1. Creating the card wall. The "card wall" is used to visualize the process, tasks, and goals clearly throughout the Kanban-based project. All steps required for implementing the project are clearly identified, and all required tasks are written onto cards or sticky notes and added to the Kanban backlog. When a task is completed in a given step, it goes downstream to the next step, and another task from the backlog is pulled from upstream [21]. The tasks in the backlog go through a number of steps until they are completed [21]. In order to deliver the project in a timely manner, limits are placed on the maximum number of tasks in any given step. If a step has fewer tasks than what the limits specify, it may accept more tasks from upstream. Otherwise, it must wait until one of its tasks completes and moved downstream before accepting another task. A queue is used to allow for slack between steps. After a given step of a task completes, it can either be given directly to the next step, or be put on hold by being placed in a queue before the next step [21]. An overview of the Kanban process is illustrated in Fig. 3.

2.5.2.2. High project visibility. Kanban methodology allows for high project visibility by visually presenting the tasks that developers are working on. The methodology also shows the bottlenecks resulting from overloading, and the gaps between workflows. Using colored cards or post-it notes enables a team to visualize the progress of the project. The visuals are helpful for tracking the time and cost constraints of the project, and for meeting the deadline. A figure containing an example of a card wall used in an actual project is shown in Fig. 4.

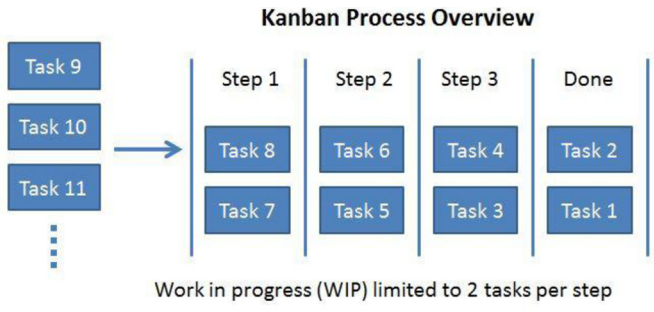

Fig. 3. Kanban Process overview.

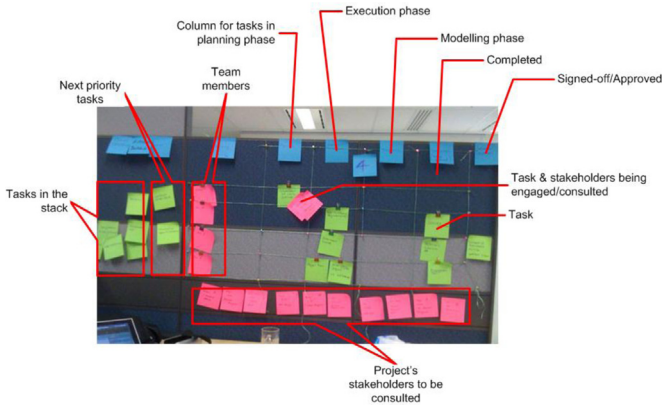

Fig. 4. Kanban card wall example used in a project. Figure produced by Matthew Hodgson, under the Creative Commons License Attribution-NonCommercial 2.0 Generic license.

2.6. Similarities of Scrum and Kanban

Kniberg and Skaring stated that Kanban and Scrum are similar in the following areas: being Lean and Agile, having the ability to break down the work into smaller pieces, having self-organized teams, focusing on delivering releasable software early and often, adapting changes quickly, limiting WIP, using pull-scheduling, and using transparency [22].

In addition, Keogh notes that both methodologies are more efficient than the Waterfall method. They contain feedback and improvement mechanisms, and clearly account for project scope. They also place importance on project value and on achieving deliverables [23].

2.7. Differences between Scrum and Kanban

Kniberg's work summarized some main differences between Scrum and Kanban, which are observational in nature. The differences include whether time-boxed iterations are needed, whether a team commits to a specific amount of work for a given product iteration, whether cross-functional teams are prescribes, whether WIP is limited, whether work must be broken down such that they can be completed within a prescribed period of time, among others [22].

In addition, Sahota explains the different contexts that Scrum or Kanban fit. Kanban can handle a lot of project interrupts, support personnel with specialized roles and different skill sets, and excel at repeatable work. Kanban also works well for larger teams since communication and planning overhead are lower. In contrary, Scrum is better at projects requiring deep collaboration and innovation. Scrum works best with small cross-functional teams, and encourages generalists [24].

3. Research methodology

While similarities and differences of Scrum and Kanban are given, the existing comparisons are not based on numerical data, but rather people's experiences and opinions on the implementations of the two methodologies. Furthermore, the comparisons are not based on the traditional project input, output, and process factors described using the popular six-pointed star model used in project management, shown in Fig. 5.

Traditionally, the factors that affect the success of a project are

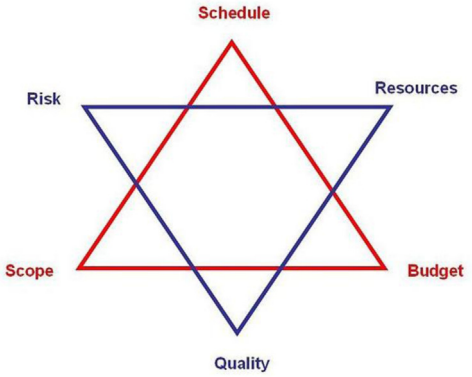

Fig. 5. Project management star per PMBOK 4.0. Figure obtained from: wikipedia.org/wiki/File:TripleConstraint.jpg.

Cost, Time, and Scope [25]. Later, the Project Management Body of Knowledge (PMBOK 4.0) defined an advanced triple-constraint model based on six factors (schedule, scope, budget, risk, resources, and quality) that are important to the success of a project [26,27]. The six factors are shown in the six-point star. The factors of schedule, scope, and budget on one triangle are the input/output factors of the project, while the factors of risk, resources, and quality on the other triangle are the process factors. The Schedule factor focuses on on-time project completion. The Scope factor pertains to the mission and goals of the project and its requirements. The Budget factor focuses on meeting the budget requirements and on achieving targeted return-on-investment. The Risk factor relates to how well project risks are managed. The Resources factor pertains to human and material resources available during the project, and the Quality factor relates to the overall success of the project. This work aims to statistically compare the effectiveness of Scrum and Kanban in terms of the six factors.

3.1. Data collection

A web-based survey (using http://www.surveymonkey.com) is used to collect numerical responses to questions pertaining to each of the project management factors in the six-pointed star model for both Scrum- and Kanban-based projects. The survey was conducted over the period of a month (April 2012–May 2012), and responses were collected from employees who are involved in software development projects at companies using Scrum or Kanban.

Table 1 shows the survey questions that relate to each of the factors of the six-pointed star model. For each survey question, participants gave one of five responses according to the Likert scale: Strongly Disagree, Disagree, Neutral, Agree, and Strongly Agree. Each Likert scale response is associated with a numerical score, shown in Table 2, and the scores are used to generate numerical survey response data.

Data collected through the survey are divided into two subsets – one for data related to Scrum projects, and the other for data related to Kanban projects. A total of 35 people responded. Among them, 60% (21 respondents) used the Scrum methodology in their software development projects, 40% (14 respondents) used Kanban (see Table 3).

Table 4 shows the business sectors of the survey respondents, while Table 5 shows the company sizes of the respondents. We also determined that eight of the 35 respondents are project managers, seven are software engineers, and three are assistant project managers. 32.4% of the respondents stated that they worked less than 2 years, 32.4% stated that they worked from 5 to 10 years, and 26.5% stated that they worked from 2 to 5 years.

Table 1
Project management factors, survey questions and question numbers.

Factor	Survey question	Question number
Schedule	• Project teams are aware of the project status.	1.1
	• Project teams can adapt the changes quickly.	1.2
	• Project is delivered on time according to schedule.	1.3
Scope	• Project usually has a well-defined scope.	2.1
	• PM methodology is effective to make the scope clearer.	2.2
Budget	• Project is delivered within budget.	3.1
	• The project provides good Return on Investment.	3.2
Risk	• Project risks and opportunities are managed.	4.1
	• Business objectives are met.	4.2
Resources	• Human and material resources are mostly available.	5.1
	• Teams can work well together to achieve expected results.	5.2
Quality	• Quality requirements are met.	6.1
	• Client satisfaction is met.	6.2
	• The project is successful overall.	6.3

Table 2
Numerical responses corresponding to the Likert scale.

Likert scale response	Score
Strongly disagree	1
Disagree	2
Neutral	3
Agree	4
Strongly agree	5

Table 3
Number of respondents using Scrum and Kanban.

Methodology	Number of respondents
Scrum	21
Kanban	14

Table 4
Business sectors of survey respondents.

Business sector	Number of respondents
Information technology	6
Consultancy	4
Education	3
E-commerce	2
Bank and finance	2
Warranty	2
Web services	1
Tourism	1

Respondents were also asked regarding the number of people working on software projects. 45.7% of the respondents stated that there are 10 to 20 people working on software projects, and 42.9% stated that there are less than 10 people.

Table 6 summarizes the survey responses for users of Scrum, and Table 7 summarizes responses for users of Kanban. The average scores are also shown for the responses to each question, computed by first multiplying the number of responses falling into each Likert scale category (i.e. Strongly Agree, etc.) by the score (1–5) associated with that category. The products for the five Likert

Table 5
Company size of survey respondents.

Size of company	Percentage of respondents (%)
Less than 50	25.7
50–100	37.1
100–500	28.6
More than 500	8.6

categories are summed, and divided by the number of responses to the question, to compute the average score for the question.

Tables 6 and 7 show that there are noticeable differences between the average scores for some of the questions for users of Scrum compared to users of Kanban. For example, respondents who used Scrum had an average score of 3.76 for question 1.1 ("Project teams are aware of project status") while respondents who used Kanban had an average score of 4.36 for the same question. This suggests that Kanban users are more aware of project status. The following sections further analyze and discuss the results.

4. Results and discussion

This section provides statistical analysis based on the numerical results from the survey responses, in terms of how the project management factors relate to one another, and how Kanban and Scrum affects each of the factors. A discussion of the results is also provided.

4.1. Correlation between project success and project management factors

The quality factor of the six-point star model is related to the overall success of the project, since the survey questions for the quality factor examine whether project quality requirements and client satisfaction are met, and whether the project is successful overall. We performed a sanity check to ensure that the quality factor is positively correlated with the other five project management factors for our data. A positive correlation between each of the other factors with the quality factor suggests that there is merit in examining how Scrum and Kanban affects each of the individual factors, which could affect the overall success of the project.

We computed the average score of the 2 or 3 questions pertaining to each factor for each participant. For example, if a respondent answered "Strongly Agree" (score of 5) for Question 1.1, "Agree" (score of 4) for Question 1.2, and "Neutral" (score of 3) for Question 1.3, then the average score for the respondent for the Schedule factor (with questions 1.1, 1.2, and 1.3) is a 4.0. Table 8 shows the average scores of each factor for all 35 respondents.

Pearson's correlation value is computed between the average scores of all respondents for the quality factor to the average scores for the other five project management factors. A correlation value of 1 indicates perfect correlation, while a value of 0 indicates no correlation. Table 9 shows the correlation values. Note that all significance-level values are less than 0.05, indicating that the correlation is significant at the 95% confidence level.

Correlation results show that the quality factor, which is an indicator for project success, is positively correlated with each of the other five factors of the six-star model. The correlation values are significant, and range from 0.60 to 0.85. This suggests that examining how Scrum and Kanban affects each of the factors of the six-star model can provide insight on how the methodologies affect the overall success of the project.

4.2. Statistical comparison of Kanban and Scrum based on Survey Results

Lastly, we compared the Scrum versus Kanban methodologies in terms of how they affect each of the individual project management factors. Table 10 shows the mean and standard deviation

Table 6
Summary of survey responses for Scrum users.

Factor	Question (number)	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Avg. score	Response count
Schedule	Project teams are aware of project status (1.1)	4.8% (1)	9.5% (2)	23.8% (5)	28.6% (6)	33.3% (7)	3.76	21
	Project teams can adapt changes quickly (1.2)	4.8% (1)	14.3% (3)	14.3% (3)	38.1% (8)	28.6% (6)	3.71	21
	Project is delivered on time according to schedule (1.3)	9.5% (2)	9.5% (2)	28.6% (6)	23.8% (5)	28.6% (6)	3.52	21
Scope	Project usually has well defined scope (2.1)	0.0% (0)	28.6% (6)	14.3% (3)	42.9% (9)	14.3% (3)	3.43	21
	Project management methodology effective to make scope clearer (2.2)	4.8% (1)	4.8% (1)	28.6% (6)	33.3% (7)	28.6% (6)	3.76	21
Budget	Project is delivered within budget (3.1)	9.5% (2)	0.0% (0)	33.3% (7)	33.3% (7)	23.8% (5)	3.62	21
	Project provides good return on investment (3.2)	4.8% (1)	0.0% (0)	19.0% (4)	33.3% (7)	42.9% (9)	4.10	21
Risk	Project risks and opportunities are managed (4.1)	4.8% (1)	4.8% (1)	28.6% (6)	47.6% (10)	14.3% (3)	3.62	21
	Business objectives are met (4.2)	4.8% (1)	0.0% (0)	14.3% (3)	47.6% (10)	33.3% (7)	4.05	21
Resources	Human/material resources are mostly available (5.1)	4.8% (1)	14.3% (3)	38.1% (8)	23.8% (5)	19.0% (4)	3.38	21
	Teams work well together to achieve expected results (5.2)	4.8% (1)	0.0% (0)	0.0% (0)	42.9% (9)	52.4% (11)	4.38	21
Quality	Quality requirements are met (6.1)	4.8% (1)	0.0% (0)	14.3% (3)	52.4% (11)	28.6% (6)	4.00	21
	Client satisfaction is met (6.2)	4.8% (1)	4.8% (1)	19.0% (4)	52.4% (11)	19.0% (4)	3.76	21
	Project is successful overall (6.3)	4.8% (1)	0.0% (0)	4.8% (1)	52.4% (11)	38.1% (8)	4.19	21

Table 7
Summary of survey responses for Kanban users.

Factor	Question (number)	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Avg. score	Response count
Schedule	Project teams are aware of project status (1.1)	0.0% (0)	0.0% (0)	14.3% (2)	35.7% (5)	50.0% (7)	4.36	14
	Project teams can adapt changes quickly (1.2)	0.0% (0)	7.1% (1)	0.0% (0)	64.3% (9)	28.6% (4)	4.14	14
	Project is delivered on time according to schedule (1.3)	0.0% (0)	14.3% (2)	21.4% (3)	50.0% (7)	14.3% (2)	3.64	14
Scope	Project usually has well defined scope (2.1)	0.0% (0)	35.7% (5)	28.6% (4)	21.4% (3)	14.3% (2)	3.14	14
	Project management methodology effective to make scope clearer (2.2)	0.0% (0)	0.0% (0)	7.1% (1)	64.3% (9)	28.6% (4)	4.21	14
Budget	Project is delivered within budget (3.1)	0.0% (0)	14.3% (2)	7.1% (1)	64.3% (9)	14.3% (2)	3.79	14
	Project provides good return on investment (3.2)	0.0% (0)	14.3% (2)	14.3% (2)	50.0% (7)	21.4% (3)	3.79	14
Risk	Project risks and opportunities are managed (4.1)	7.1% (1)	0.0% (0)	42.9% (6)	28.6% (4)	21.4% (3)	3.57	14
	Business objectives are met (4.2)	0.0% (0)	0.0% (0)	7.1% (1)	57.1% (8)	35.7% (5)	4.29	14
Resources	Human/material resources are mostly available (5.1)	0.0% (0)	14.3% (2)	21.4% (3)	64.3% (9)	0.0% (0)	3.5	14
	Teams work well together to achieve expected results (5.2)	0.0% (0)	0.0% (0)	14.3% (2)	35.7% (5)	50.0% (7)	4.36	14
Quality	Quality requirements are met (6.1)	0.0% (0)	7.1% (1)	21.4% (3)	42.9% (6)	28.6% (4)	3.93	14
	Client satisfaction is met (6.2)	0.0% (0)	14.3% (2)	21.4% (3)	35.7% (5)	28.6% (4)	3.79	14
	Project is successful overall (6.3)	0.0% (0)	0.0% (0)	0.0% (0)	64.3% (9)	35.7% (5)	4.36	14

Table 8
Average scores of each factor for each respondent.

Respondent number	**Schedule** factor average	**Scope** factor average	**Budget** factor average	**Risk** factor average	**Resources** factor average	**Quality** factor average
1	3.33	3	3	3	3.5	3.67
2	3.33	3.5	3	3.5	3	4.00
3	3.33	3	3	3	3.5	3.33
4	2.33	3	3.5	3.5	3.5	3.33
5	2.67	3	4	4	4.5	3.67
6	2.67	2.5	3	3	4	3.00
7	4.00	3.5	4	4	3.5	4.00
8	1.00	1.5	1	1	1	1.00
9	4.67	4	5	4.5	5	5.00
10	3.67	4.5	5	5	4.5	4.33
11	4.67	4	5	3.5	5	4.00
12	5.00	5	5	5	5	5.00
13	5.00	4.5	4.5	4	4.5	4.33
14	2.67	4	2.5	3.5	4.5	4.33
15	4.33	3.5	5	4.5	3	4.67
16	5.00	4	4	4	4.5	4.33
17	3.00	4.5	4.5	4.5	3.5	5.00
18	4.00	3.5	4	4.5	4	4.33
19	4.00	4.5	4.5	4	4	4.33
20	4.33	4	3.5	4	4	4.00
21	4.00	2.5	4	4.5	3.5	4.00
22	4.00	4	4	3.5	4	4.00
23	4.00	3.5	4	3.5	3.5	4.00
24	3.67	3	3.5	3.5	3.5	3.33
25	3.67	3	3.5	3.5	3.5	3.33
26	2.67	4	3.5	5	3.5	4.00
27	4.33	3	3	3.5	4	4.00
28	4.67	5	5	4	4.5	5.00
29	4.67	4	4.5	4	4.5	4.67
30	4.00	3.5	4.5	4	4.5	3.67
31	4.33	3	3	4	4.5	3.67
32	3.67	5	3	5	4	5.00
33	4.67	3	3.5	3	3	3.33
34	3.67	4	4	4	3.5	4.67
35	4.67	3.5	4	4.5	4.5	3.67

Table 9
Correlation between success of the project and project management factors for Scrum and Kanban.

Factor	Pearson's correlation with quality factor
Scope	0.85
Schedule	0.60
Budget	0.71
Risk	0.81
Resources	0.63

Table 10
Average and standard deviation values of scores for each factor, for both Scrum and Kanban.

Factor	Scrum		Kanban	
	Mean	**Std. Dev.**	**Mean**	**Std. Dev.**
Schedule	**3.67**	**1.20**	**4.05**	**0.85**
Scope	3.60	1.08	3.68	1.02
Budget	3.87	**1.12**	3.79	**0.92**
Risk	3.83	0.99	3.93	0.94
Resources	3.88	**1.13**	3.93	**0.86**
Quality	**3.98**	0.96	**4.02**	0.87

values of the scores for each survey question across all respondents, separated according to whether respondents used Scrum or Kanban. Since there are 21 Scrum respondents, and there are 2 or 3 questions pertaining to each factor, a total of 42 or 63 scores are used to compute the average and standard deviation value of each factor for Scrum. There are 14 Kanban respondents, implying that a total of 28 or 42 scores are used to compute the average and standard deviation values of each factor for Kanban.

A significance-level test for differences between the means given the standard deviations for each factor in Table 10 suggests that the means are not significantly different at the 95% confidence

Table 11
Averaged scores for questions pertaining to the Schedule factor, based on company size, and usage of Scrum or Kanban.

Company size	Schedule factor average for **Scrum**	Schedule factor average for **Kanban**
Less than 50	3.72	4.22
50–100	3.56	4.08
100–500	4.00	4.00
More than 500	3.33	3.67

level. However, interesting observations can nevertheless be made based on the results.

The results suggest that Kanban is better than Scrum for the Schedule factor (mean of 4.05 for Kanban versus 3.67 for Scrum), which correspond to the following survey questions:

- Project teams are aware of the project status.
- Project is delivered on time according to schedule.
- Project teams can adapt the changes quickly.

We performed additional analysis examining the effect of company size on the performance of Scrum versus Kanban for the Schedule factor. Survey scores for all questions pertaining to the Schedule factor (questions 1.1, 1.2, and 1.3) are averaged separately for respondents working on Scrum- and Kanban-based projects for company sizes of less than 50, 50–100, 100–500, and more than 500. Results are shown in Table 11.

Results suggest that Kanban leads to better performance for the Schedule factor for smaller company sizes (less than 50, and 50–100). For companies of less than 50, Kanban has an average score of 4.22 for questions pertaining to the Schedule factor, while Scrum has an average score of 3.72. For larger company sizes, there is less difference between Scrum and Kanban. Section 2 mentions that one advantage of Kanban is that it reduces planning overhead and works well for teams where members have specialized roles and different skill sets. This could be the case for those survey respondents who worked in the smaller companies.

With the exception of the Budget factor, the mean scores for Kanban are also consistently slightly greater than the Scrum scores in Table 10. Examining the standard deviation values suggest that there is generally greater variance to the Scrum scores compared to Kanban scores, which means that there is greater variance in the answers to the survey questions for Scrum users. For the Schedule, Budget, and Resources factors, the standard deviations for Scrum is at least 0.2 higher than the standard deviation for Kanban. This suggests that projects using the Kanban methodology can experience greater consistency in terms of the project management factors.

Overall, our findings indicate that both Kanban and Scrum approaches can lead to successful software development projects, with average scores of around 4.0 for the quality factor (3.98 for Scrum, 4.02 for Kanban). Thus, on average, the survey respondents responded with the “Agree” response to questions pertaining to the quality factor. Whether Scrum or Kanban ought to be used likely depends on the context, practical needs, and resources of the project.

5. Conclusion

There has been a debate for years about whether Scrum or Kanban is more efficient than the other, and a review of literature indicates that there is a lack of statistical evidence on this topic. This research was conducted to see if there is a statistically-verifiable difference between these two methodologies impacting the a set of six project management factors for software-development projects: budget handling, risk control, quality of the project, amount of available resources, clear project scope, and schedule handling. These factors were chosen as they were listed as factors to be monitored and managed in the Project Management Body of Knowledge (PMBOK 4.0).

In order to compare the effectiveness of Kanban versus Scrum on software projects, a survey was designed to include various questions about the company, software projects, project management methodology, implementation, and project feedback. Mean, standard deviation, and correlation results were computed to compare the effects that Scrum and Kanban have on the factors. While the results imply that there is no statistically significant difference at the 95% confidence level between Kanban and Scrum for the factors, the results do suggest that Kanban performs better than Scrum in terms of managing project schedule (i.e. the schedule factor). Results also suggest that projects using the Kanban methodology can experience greater consistency in terms of the project management factors. Results also suggest that overall, both Scrum and Kanban lead to successful software projects, where on average, the survey respondents responded with the “Agree” response to questions pertaining to the quality factor. Companies should be aware of the differences in the practical implementation of these methodologies, and choose one or the other based on the context, practical needs, and resources of the project.

In the future, we plan to consider the impact of additional non-quantitative techniques, such as team commitment, work organization, schedule managing, allocation of resources, and visibility. We will also consider collecting additional survey responses to questions pertaining to all project management factors.

References

[1] C. Larman, V.R. Basili, Iterative and incremental development: a brief history, IEEE Comput. Soc. 36 (6) (2003) 47–56, http://dx.doi.org/10.1109/MC.2003.1204375.
[2] H.D. Benington, Production of large computer programs, in: Proceeding of the Navy Symposium on Advanced Programming Methods for Digital Computers Office of Naval Research, Department of the Navy Washington, DC, 1956, pp. 15–28.
[3] B. Blanchard, W. Fabrycky, Systems Engineering and Analysis, 4th ed., Prentice Hall, Upper Saddle River, NJ, 2011.
[4] R.T. Futrell, D.F. Shafer, L. Shafer, Selecting software development life cycles, Quality Software Project Management, 1st ed., Prentice Hall, Upper Saddle River, NJ (2002), pp. 101–161.
[5] M.A. Awad, A Comparison between Agile and Traditional Software Development Methodologies (Unpublished honours programme's dissertation), School of Computer Science and Software Engineering, The University of Western Australia, Crawley, WA, Australia, 2005.
[6] M. James, Scrum Methodology ⟨http://scrummethodology.com⟩, 2009 (retrieved 8.04.15).
[7] T. Ohno, Toyota Production System: Beyond large-scale production, Productivity Press, Portland, Oregon (1988), p. 29.
[8] J.P. Womack, T.J. Daniel, R. Daniel, The Machine That Changed the World: The Story of Lean Production, Harper Collins, New York, NY, 1990.
[9] D. SjØberg, A. Johnsen, J. Solberg, Quantifying the effect of using Kanban versus Scrum: a case study, in: IEEE Software, September/October, 2012.
[10] Project Management Institute, Project Management Proccesses for a Project, A Guide to the Project Management Body of Knowledge, Newtown Square, PA: Project Management Institute (2008), p. 47.
[11] F. Ganjeizadeh, H. Zong, P. Ozcan, E. Olivar, Effectiveness comparison between Kanban and Scrum on software development projects, in: Proceedings of the 24th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM), 1, 2, 2014, pp. 607–615.
[12] J. Highsmith, Adaptive software development, Agile Software Development Ecosystems, Addison-Wesley Professional, Indianapolis, IN (2002), pp. 309–321.
[13] M. Fowler, J. Highsmith, The Agile Manifesto. From Agile Alliance: ⟨http://agilemanifesto.org⟩, 2001 (retrieved 14.06.12).
[14] H. Takeuchi, N. Ikujiro, The new product development game, Harv. Bus. Rev. 64 (1) (1986), http://dx.doi.org/10.1016/0737-6782(86)90053-6 137-14.
[15] J. Sutherland, Agile can scale: inventing and reinventing Scrum in five companies, Cut. IT J. 14 (2001) 5–11.
[16] J. Sutherland, Agile development: lessons learned from the first scrum, Cut.

Agil. Proj. Manag. Advis. Serv. 5 (2004) 20, Executive Update.
[17] M. Beedle, K. Schwaber, Get Ready for Scrum! Agile Software Development with Scrum, 1st ed., Prentice Hall, Upper Saddle River, NJ (2002), pp. 23–30.
[18] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Inf. Softw. Technol. 50 (9–10) (2008) 833–859, http://dx.doi.org/10.1016/j.infsof.2008.01.006.
[19] K. Schwaber, J. Sutherland, The Scrum Guide, the Definitive Guide to scrum: The Rules of the Game. From ⟨http://www.scrum.org/Portals/0/Documents/ScrumGuides/Scrum_Guide.pdf⟩, 2011 (retrieved 14.06.12).
[20] D. Anderson, Kanban-Successful Evolutionary Change for Your Technology Business, WA: David J. Anderson & Associates Inc, Seattle, 2010.
[21] D. Joyce, Kanban for Software Engineering, Kanban Images. From ⟨http://leanandkanban.files.wordpress.com/2009/04/kanban-for-software-engineering-apr-242.pdf⟩, 2009 (retrieved 14.06.12).
[22] H. Kniberg, M. Skarin, Kanban vs Scrum: How to Make the Most of Both. From C4media: ⟨http://www.infoq.com/minibooks/kanban-scrum-minibook⟩, 2009 (retrieved 14.06.12).
[23] L. Keogh, Scrum and Kanban both the Same, only Different. from ⟨http://lizkeogh.com/2011/11/20/scrum-and-kanban-both-the-same-only-different/⟩, 2011 (retrieved 14.06.12).
[24] M. Sahota, Scrum or kanban? yes!. From ⟨http://agilitrix.com/2010/05/scrum-or-kanban-yes⟩, 2010 (retrieved 14.06.12).
[25] C. Chatfield, T. Johnson, Microsoft Office Project 2007 Step by Step. From ⟨http://office.microsoft.com/en-us/project-help/a-short-course-in-project-management-HA010235482.aspx#BMtime⟩, 2007 (retrieved 14.07.12).
[26] W.R. Duncan, Project management processes, A Guide to Project Management Body of Knowledge, MD: Automated Graphic Systems, White Plains, (1996), pp. 27–39.
[27] K. Schwaber, Agile Project Management with Scrum, Microsoft Press, Redmond, WA, 2004.