An Investigation of Approaches to Set Up a Kanban Board, and of Tools to Manage it

ERIKA CORONA, FILIPPO EROS PANI
Department of Electric and Electronic Engineering, Agile Group
University of Cagliari
Piazza d'Armi, 09123 Cagliari
ITALY
{erika.corona, filippo.pani}@diee.unica.it   http://agile.diee.unica.it

Abstract: - This paper presents a survey with the intent to address a series of issues of the Lean-Kanban approach in the software development, and specifically the guidelines and tools used to set-up a Kanban board. Following the Lean principles, a software process can be broken down into steps and managed with a Kanban approach. Despite the recent increase of interest on the subject, there is no standard definition of Kanban system for software development, and the specific practices of Kanban have not yet been rigorously defined. The purpose of this work is a rigorous analysis of the available information, through research questions and answers, to show the state-of the-art about how Kanban approach is presented and used, in particular those related to the Kanban board management, and to study how they are addressed in practice. We used the methods of Evidence-based software engineering, performing a systematic review of the available information. In our opinion, the information gathered might be very useful to people considering Kanban board adoption, and to the whole community of agile developers practicing Lean-Kanban system approach.

Key-Words: - Kanban, Lean, software development, agile methodologies.

1 Introduction

Agile Methodologies is a name referring to a set of practices and processes for software development that have been created by experienced practitioners [1]. Traditional software engineering is said to advocate extensive planning, up-front analysis and design, and codified processes to make development an efficient and predictable activity. By contrast, AMs address the challenge of an unpredictable world by relying on ''people and their creativity rather than on processes" [2]. In contrast to traditional software development processes, where work is typically broken down into a series of sequential steps, agile methods rely on short, iterative cycles and close collaboration between the customers and the development team [3][4].

Very recently, it seems that the application to software of Lean approach and its concepts and practices is becoming increasingly popular. As reported by Hibbs C., Jewett S. and Sullivan M. [5], it is only recently that the Lean principles have been applied to software development. In the beginning it all started with Lean manufacturing [6]; Lean strives to deliver value to the customer more efficiently by finding and eliminating waste (the impediments to productivity and quality). In 2003, Mary and Tom Poppendieck published the first book about applying Lean principles to software development [7]. They identified seven key lean principles: eliminate waste[1], build quality in, create knowledge, defer commitment, deliver fast, respect people and optimize the whole.

Perhaps the most important Lean activity is to build a value stream map. This means to break down a process into individual steps, and identify which steps add value and which steps do not, thus adding to the waste (muda). The goal, then, is to eliminate the waste and improve the value-added steps (kaizen). An important lean tool helping to manage the work flow is the concept of pull system, which is usually visualized using a Kanban board.

In general, we can define the Kanban software process as a WIP (Work In Process) limited pull system visualized by the Kanban board.

Recently, the Kanban approach applied to software development, seem to be one of the hottest topics of Lean. In the recent 3-4 years, Kanban has been applied to software process, and is becoming the key Lean practice in this field. A correct use of

[1]Waste in software development: partially done work; extra processes; extra features; task switching; waiting; motion; defects [10].

the Kanban board helps to minimize WIP, to highlight the constraints, and to coordinate the team work. However, Lean is more than Kanban, and more Lean practices should be used, together with Kanban, to take full advantage of the application of Lean to software development.

Kanban systems are an approach to scheduling work. Kanban shares with typical AMs the fact that requirements are expressed in atomic features (also known as user stories, work items, Minimum Marketable Features, or MMF), to be implemented incrementally.

Kanban systems focus on a continuous flow of work, and disregard fixed iterations. When needed, the team chooses a subset of features from the backlog and moves them to the Kanban board. Then, it develops these features one after the other. Work is delivered as soon as it's ready, and the team only works on one – or very few – feature at a time.

The growing interest on Kanban software development is demonstrated by the publication of various books, and by the proliferation of Web sites on the subject in the past couple of years. In his recent book, David J. Anderson describes about how to apply Kanban concepts to systems and software development [8]. Corey Ladas, in his book Scrumban, writes about the fusion of Scrum and Kanban practices [9]. A third book on the subject is *Kanban and Scrum making the most of both*, by H. Kniberg and M. Skarin, also availabile online [10]. This book analyzes both approaches through practical charts and examples.

However, we stress that, despite the recent increase of interest on the subject, there is no standard definition of Kanban system for software development. Moreover, the specific practices of Kanban – how to specify a feature, which activities best represent the software process, how to represent tasks, how to deal with exceptions and emergencies, and so on – have not yet been rigorously defined.

The purpose of this work is the analysis of the available information – through Evidence-based software engineering techniques – to highlight and discuss the state-of the-art about how Kanban approach is presented and used.

The paper is organized as follows: in Section 2, we give an overview of Kanban approach; in Section 3 we present the methodology followed for the survey and the related research questions; Section 4 presents the results and Section 5 concludes the paper.

2 The Kanban Approach

2.1 Kanban Axioms

As reported by Corey Ladas [9], the whole Lean/Kanban approach is based on two axioms.

The first is: "*It is possible to divide the work into small value adding increments that can be independently scheduled*". As said before, these increments can be called features, user stories, work items, or MMF. From now on, we will use the term "feature". This axiom is the same as in AMs, which in turn are always features-driven.

The second Ladas' axiom is *It is possible to develop any value-adding increment in a continuous flow from requirement to deployment*. Following this axiom, software development process can be decomposed in a sequence of well defined activities, to be performed one after the other by the members of a feature team on the specific features to be implemented [11-14]. For instance, a requirement analysis phase is followed by a design phase, then by an implementation phase, by a testing phase, by an integration phase and eventually by a deployment phase. For the Kanban approach to work, we need that all features are processed by the same sequence of steps.

These axioms generally hold, except perhaps at the beginning of the development of a software system, when an up-front analysis and architectural design phase is needed (as for instance explicitly prescribed in FDD methodology). In the case of addition of functionalities to an already developed system, or of maintenance and bug-fixing activities, these axioms clearly hold.

2.2 Kanban Overview

Kanban - meaning "signboard" - is a concept related to lean and just in time (JIT) production. According to Taiichi Ohno, Kanban is one of the means through which JIT is achieved [15]. Kanban is not an inventory control system, but it can be considered as a system for visualizing work, making it flow, reducing waste, and maximizing customer value. It is a pull system, because it uses the rate of demand to control the rate of production, passing demand from the end customer up through the chain of customer-store processes.

In practice, setting up a Kanban system, also in the light of Ladas' axioms, typically includes the following steps:

1. Map the flow, finding the activities.
2. Express the requirements through a set of

features.
3. Depending on the activities and the team composition, devise a maximum limit for the features under work in each activity.
4. Set-up the Kanban board, highlighting the activities and how deal with specific issues. For instance: input queue, slack buffers and "Done" buffers; task management within activities; multi-project management through lanes or other means; high-priority features, special cause circumstances in which it is allowed to break limits; management of bugs, and of features to rework before their release
5. Devise the policy to assign developers to activities and tasks, and to deal with issues related to flow (blocks, tuning of limits, etc.).
6. Decide the format and typical scheduling of meetings. For instance: daily stand-up meeting; meetings with customer and product owner; planning meetings; review meetings, including process improvement meetings.
7. Devise how releases of single features, and of working versions of the system, are delivered.
8. Devise the specific technical practices to use (design, programming, testing, etc.).
9. Decide what tools, statistical methods and diagrams to use to manage the process.

As said before, there is no a standard, or at least a commonly shared way, to perform these tasks. The aim of the followings of this paper is to highlight the specific Kanban issues, in particular those related to the Kanban board management, and study how they are addressed in practice, through a survey.

3 Method

Evidence-based software engineering (EBSE) aims to apply an evidence-based approach to software engineering research and practice. This research follows Kitchenham's methodological guide-lines for systematic reviews [16]. The research questions (RQs) of this review are the following:

Q1. What are the main characteristics of the Kanban boards actually used?

Q2. What are the main activities defining the software process, and what are their typical limits to limit WIP (for a typical development team)?

Q3. What is the information typically shown on the cards representing the work units?

Q4. What diagrams/statistics are used for a quantitative process management?

A systematic literature review (SLR) is the main method of synthesis for supporting EBSE. We performed a qualitative survey, covering both the literature and the main websites on the topic, with the aim to answer the research questions. Usually, surveys similar to the presented one are performed through an SLR of scientific papers that appeared in the literature on the subject [16]. The Kanban approach in software development, however, is still in its infancy, and there is almost no paper at present published in the scientific literature. Moreover, information about how a software development approach is applied inside an organization is often considered confidential, and it is not easy to obtain such information through interviews. consequently, our sources were the three books published on the subject so far [8], [9] and [10], and the documents available on the Web. In particular, we performed the Web survey starting from:

- the Web sites of the well known organizations working on Kanban (Limited Wip Society [17], Lean Software and Systems Consortium), and the links found there;
- the results of Web searches in the main search engines, with the keywords: "Lean", "Kanban", "software development".

We used as information sources the documents and the presentations found on these Web sites and the relevant Web pages. The survey was conducted through the analysis of various Kanban Boards reported in figures and photos, together with the analysis of the related text. All data obtained and analyzed are reported and discussed in section 4.

3.1 The Issues Studied

Despite its growing adoption, the Kanban system approach is still in its childhood and, as said before, there are no standard ways to address some key issues. In our opinion, the information gathered in our survey might be very useful to people considering Kanban adoption, and to the whole community of agile developers practicing Kanban system approach.

In the following of this section, we briefly discuss what are the issues we considered, and why. We will focus on describing the visual aspect of the Kanban board, its activities and the features. However, one must keep in mind that such a visual aspect always reflects the practices and the workflow organization decided by the team.

The Kanban board. The board is the main tool used to visualize and coordinate teamwork. Its columns show a sequence of activities, where the cards representing the features under work are put. For each activity, there are limits to the number of features, to obtain an overall limited WIP. The activities can be represented by a single column, or columns for in progress and done features can be present. An activity can also be preceded or followed by slack buffer columns, holding the features to be pulled into the next activity. The board may also have columns holding the features not yet under work, to be pulled into the first activity, and holding the features completed, or live.

Other variants of the Kanban board include boards with horizontal lanes, representing different projects, with an emergency lane for urgent features, with zones holding cards representing bugs or open issues. The developers are often represented on the board, using their names or avatars, to highlight the features they are currently working on.

Feature representation and management. On the Kanban board, the features are typically represented using cards. The color of the card may have a meaning. The information written on the card is not standard. It may include the starting date, the due date, if present, the description of the feature, a priority level, the developer currently working on it, and other information.

When features represent a substantial amount of work, they can be divided in tasks, in turn represented by cards, usually smaller and/or of different color and attached to a specific zone of the column of the activity the feature is under work. Also bugs, rework, acceptance tests related to a feature can be represented with cards.

When a feature gets stuck in an activity for some reason related to poor software quality, or undecided requirements, the work flow can be badly affected. The way this issue is resolved is often reflected in the feature representation – for instance it can be marked with two big red starts, meaning panic [10] – or in a zone on the board holding these features.

Statistics and diagrams. The use of statistics and diagrams to monitor the process is integral part of the Kanban approach. The quantities computed and monitored, however, may vary. They can be lead/cycle time, development time, engineering time, days blocked, number of bugs, throughput, and so on. These data are usually shown in diagrams, affixed to the walls of the workplace, or in any case continuously updated and made public. The most used diagram is the Cumulative Flow Diagram (CFD), used to show WIP and average lead time, and to highlight issues and bottlenecks.

4 Results and Discussion

The first RQ regards the layout of the Kanban boards actually used by developers. We collected the following data from 14 observed Kanban boards [12-14][17-24].

The number of activities ranges from one to six, with a median of 4 and an average of 3.7. So, the typical number of activities we found is 4. All boards but one divide the columns of at least some activities in two areas: in progress, where the features under work are put, and done, where the features completed wait to be pulled to the next activity. Most boards use also slack buffers before some activities.

Regarding the queue of the features to be implemented (Input queue), most boards have a limit on it, ranging from 2 to 10. The names given to this queue are very different, for each of the boards analyzed. On the contrary, most boards have no limit on the queue of features completed (Output queue). Also in this case, the names of the queue are very different from board to board, the most popular being “Done”.

6 boards on 14 have an “express lane” where urgent features are put, which can overcome the limits on the activities. This figure may look low, but remember that several of the studied boards are simplified boards, intended for didactic purposes. 5 boards have “lanes”, highlighting features belonging to different projects which are carried on concurrently by the team. Only three boards explicitly show activities divided in task. With this analysis we have answered **Q1**.

Let us now pass to **Q2**. First, let us note that it is patent from the board analysis presented previously that the same concepts are named differently in the various boards. The same variability can be found in the names of the activities. So, we tried to put together the activities that look very similar, albeit having different names, for instance: “Development”, “Dev.”, “Code”, “Coding”. For each activity and for each Kanban board studied, we collect different main characteristics (buffers and limit): some activities might refer to the same activity in different boards – for instance “Specification” and “Analyze”, “Build” and “Development”. However, there are boards where both activities are included, so we did not merge them in our analysis. Overall, there are the following broad categories of activities:

Specification/Analysis : this is typically the first activity. Its limits vary from 1 to 8, with an average of 3.7, a median equal to 3, and a mode equal to 2. Since the value 8 seems an outlier, the preferred limits to this activity are 2 and 3.

Build/Development: this is the activity referring to actually writing the code. Its limits vary from 2 to

10. The mean, median and mode of these limits are all equal to 4.

Test/Acceptance: these activities refer to writing and/or executing tests on the system. Their limits vary from 2 to 8, with an average of 3.2, a median equal to 3, and a mode equal to 2. Also in this case, the median and the mode look the most representative values.

Deploy/Release: this is the last typical activity when a system is developed. Only in four cases there are explicit limits, ranging from 4 to 6. The lack of limits is due to the fact that in some processes release is not really a full-scale activity, but it refers to the acceptance of the released features by the product owner, or other stakeholder.

Documentation: in two cases, this activity is explicitly recorded on the Kanban board. In one of these board, the limit is 2, while in the other it is not specified.

We stress that some Kanban boards are in fact organized in multiple tiers, and the sequence of activities is not linear, but activities are part of higher-level tiers, in turn executed in sequence, or in other ways. However, we believe that our analysis summarizes well how Kanban teams divide development into activities, and give hints on the possible choices of their limits. So, with its discussion we believe we answered **Q2**.

Regarding the result of the study about how features are named and represented on cards in the Kanban board, we were able to get information only on five different boards, because in the other boards we considered, the cards were only sketched [8][13][14][19][22]. All feature cards show a description of the feature, the date the feature entered the system, and are related with the developer in charge of it, often represented with another card, or a "stick avatar". All examined boards make use of cards of different colors to highlight the kind of feature; some of them use also cards of different size, typically to discriminate between the features and the tasks obtained by decomposing the features, bugs, issues and the like. Some cards use smaller sticker cards on them, to denote issues or their state, and some cards allow to show specific states of the feature, such as high priority, late or blocked. With this analysis we answered **Q3**.

Let us now pass to **Q4**: for each Kanban implementation studied we figured out what kind of quantitative tools were used by the associated team. This analysis was not simple, because in many presentations the main goal was to describe the Kanban approach and the board, with minimal or no emphasis on these tools.

Overall, we were able to find information only in seven sources, on overall 14 considered, about statistics used by the analyzed Kanban implementations [8][10][11][19][21][25][26]. All authors use the Cumulative Flow Diagram, which is one of the distinctive characteristics of the Kanban approach, and the Lead time per feature statistics. Some authors use the diagram showing the throughput of the development process, that is the number of features (weighted with the needed effort) completed per week or per month). Other statistics are used, but are less spread. This answers **Q4**.

We believe we shown the most comprehensive comparison of Kanban tools available to date.

5 Conclusion

Agile development methodologies have gained significant adoption in a variety of software development domains. Nowadays, the fastest growing AM is perhaps the Lean approach, using the Kanban board for its practical implementation. However, despite the strong increase of interest on Lean-Kanban, there is no standard definition of Kanban system for software development, and the specific practices of Kanban have not yet been rigorously defined. To address this issue, in this work we presented a rigorous analysis of the available information, through research questions and answers, to show the state-of the-art about how Lean-Kanban approach is presented and used. In particular, we formulated and answered four research questions related to the Kanban board management, the use of diagrams and statistical tools. We used the methods of Evidence-based software engineering, performing a systematic review of the available information.

We examined 14 different Kanban boards, looking for similarities and differences in the board layout, and in the activities used for decomposing the software development work. We also analysed how work items, or features, are graphically represented in cards on the boards, and which graphical and statistical tools are typically used by Lean-Kanban teams. The results from this review can help both insiders' and outsiders' perception and understanding of how the Lean-Kanban approach is actually implemented. This information, derived from literature and Web site analysis has the potential to suggest possible directions for Lean development standardization and improvement, and to be useful to people considering Kanban adoption. As further works, the Kanban Approach can be evaluate as a process using simulation modeling approach for software development [27], useful to better understand the process and to evaluate its effectiveness [28], and as a products measuring new metrics [29] and analyzing the evolution of the system [30]. We also intend to analyze and compare software tools for managing virtual Kanban boards.

References:

[1] *Manifesto for Agile Software Development*, www.agilemanifesto.org/

[2] Dyba T., Dingsøyr T., Empirical studies of agile software development: A systematic review. *Information and Software Technology*, Vol. 50, nos. 9/10, 2008, pp. 833-859.

[3] Abrahamsson P., Warsta J., Siponen M.T. and Ronkainen J., New Directions on Agile Methods: A Comparative Analysis. In: *Proceedings of the International Conference on Software Engineering*, Portland, Oregon, USA, 2003.

[4] Beck K., *Extreme Programming Explained: Embrace Change*, second ed., Addison-Wesley, 2004.

[5] Hibbs C., Jewett S. and Sullivan M., *The Art of the software Development*, O'Reilly, 2009.

[6] Womack J.P., Jones D.T., Roos D., *The machine that Changed the World*, Simon&Shuster, 1990.

[7] Poppendieck M. and Poppendieck T., *Lean software development: An agile toolkit*, Addison Wesley, 2003.

[8] Anderson D.J., *Kanban: Successful Evolutionary Change for Your Technology Business*, Blue Hole Press, 2010.

[9] Ladas C., *Scrumban*, Modus Cooperandi Press, 2008.

[10] Kniberg H. and Skarin M., *Kanban and Scrum making the most of both*, C4Media Inc, 2010.

[11] leansoftwareengineering.com

[12] www.lostechies.com

[13] agileconsulting.blogspot.com/

[14] www.crisp.se/

[15] Ohno T., *Just-In-Time for Today and Tomorrow*, Productivity Press, 1988.

[16] Kitchenham, B. and Charters, S., *Guidelines for performing Systematic Literature Reviews in Software Engineering. Engineering 2*, 2007.

[17] www.limitedwipsociety.org

[18] Anderson D.: Kanban Primer, Better Software, January/February 2009

[19] ninjaferret.co.uk

[20] leanandkanban.wordpress.com

[21] blog.brodzinski.com

[22] www.agileproductdesign.com

[23] Sundén J., Hammarberg M. and Achouiantz C.,www.slideshare.net/marcusoftnet/kanbanboards

[24] blog.crisp.se/mattiasskarin/2010/12/03/1291361993216.html

[25] manicprogrammer.com/cs/blogs/willeke

[26] availagility.co.uk/

[27] Turnu, I., Melis, M., Cau, A., Setzu, A., Concas, G., Mannaro, K., Modeling and simulation of open source development using an agile practice, *Journal of Systems Architecture 52 (11)*, 2006, pp. 610-618.

[28] Melis, M., Turnu, I., Cau, A., Concas, G.: Evaluating the impact of test-first programming and pair programming through software process simulation, *Software Process Improvement and Practice 11 (4)*, 2006, pp. 345-360.

[29] Concas, G., Marchesi, M., Murgia, A., Pinna, S., Tonelli, R.: Assessing traditional and new metrics for object-oriented systems, *Proceedings International Conference on Software Engineering*, 2010, pp. 24-31,

[30] Turnu, I., Concas, G., Marchesi, M., Pinna, S., Tonelli, R.: A modified Yule process to model the evolution of some object-oriented system properties, *Information Sciences 181 (4)*, 2001, pp. 883-902.