INFORMATYKA EKONOMICZNA BUSINESS INFORMATICS 2(24) • 2012
ISSN 1507-3858

Łukasz D. Sienkiewicz
Wrocław University of Economics
lukasz.sienkiewicz@ue.wroc.pl

SCRUMBAN – THE KANBAN AS AN ADDITION TO SCRUM SOFTWARE DEVELOPMENT METHOD IN A NETWORK ORGANIZATION[1]

Abstract: A large number of press release and scientific publications treat the Scrum as the best approach to software development. Nevertheless, the original Scrum method is not sufficient for managing work in Agile environment within a Network Organization. Due to that, we extended the Scrum-based model [Sienkiewicz, Maciaszek 2011] and the Kanban approach to make the most of both and find a more sufficient approach for managing software development in strongly distributed Agile environment (i.e., Network Organization).

Keywords: Scrum, Third Party Service, Network Organization, Scrumban, Kanban, Agile, Key Performance Indicators, software development.

1. Introduction

Selecting an appropriate approach to systems development is crucial for the success of the project. This is especially important when firms are working as a Network Organization, where the environment is turbulent and uncertain.

In this paper, we propose a Scrum-based model with an addition of a few Kanban artifacts and rules that suit best managing the development of software applications in Network Organizations:

- We propose to include some novel artifacts and rules taken from the Kanban approach that help better manage software development and Third Party Services in Network Organizations.
- We consider the usage of the Kanban software development method as an addition to the Scrum-based model [Sienkiewicz, Maciaszek 2011] with particular emphasis to a distributed environment.

We believe that the improvements proposed in this paper will help better understand the software development method in Network Organizations and will be the basis for more empirical research.

[1] Selected parts of this article were published under nonexclusive copyright in FEDCSIS'2011 proceedings [Sienkiewicz, Maciaszek 2011].

2. Kanban as a software development method

Originally, the idea was designed and used by Toyota to maintain the high rate of improvements within Lean [Holweg 2007; Womack, Jones, Ross 1991] and Just--In-Time [Shingo 1989] production. In general, "The Kanban" is a physical card used to support non-centralized "pull" production control, which had spread to the manufacturing industry and then to software development as a tool of the Lean software development approach.

We assume that the reader is familiar with the Kanban concept in Lean manufacturing, so in this chapter we will focus only on those elements that are crucial for implementing Kanban as a software development method.

2.1. Principles of Kanban in software development

Nowadays, the visualization of software development differs a little bit. Instead of posting task cards on a wall, a commonly seen practice is using whiteboards – called "Kanban Boards". Usually it is a whiteboard with drawn columns and plenty of yellow sticky notes (actually sticky notes have many colors), which are ordered and placed in the correct column that represent the actual status of a task.

Many authors [Norrmalm 2011; Ikonen 2010; Kniberg, Skarin 2009] identify only three important practices (i.e., visualize workflow, limit workflow, and manage workload). In our consideration, we take advantage of the proposal of David J. Anderson, who identified the five most important principles/properties of the Kanban method [Anderson 2010]:

- Visualize The Workflow – prepare named columns (e.g., draw them on white-board) representing all states of the task (e.g., to-do, investigation, development, testing, done), split the work into single items, submit each item on a card (e.g., sticky notes) and put on the board.
- Limit Work In Progress (WIP) – assigned to each column, is an explicit limit to how many items (i.e., tasks) may be in each workflow state (i.e., column).
- Manage Flow – measure the lead-time and try to adjust the whole flow to be as short/small as possible in order to avoid waste in the entire process.
- Make Process Policies Explicit – be sure that all stakeholders have a common explicit understanding of the process policies and are able to build a shared comprehension of the process, problems and suggested improvements.
- Improve Collaboratively the Process – continuously and incrementally all stake-holders should care for and improve the process. Use a scientific approach to implement continuous, incremental and evolutionary changes of work, workflow and process.

In the next sections, we will introduce the Scrumban approach with particular emphasis on the principles of the Kanban software development method, represented in the artifacts and rules of the Scrum-based model.

3. Scrumban – make the most of both approaches

In Sienkiewicz, Maciaszek [2011], we introduced a Scrum-based model as a holonic view on the Scrum method used for managing software development in a Network Organization. In the approach presented here, we specifically include some novel and exclude some conventional artifacts and rules to make the Scrum software development method sufficient for work with Third Party Services within a Network Organization. The presented examples show that our Scrum-based model is a Scrumban approach, which makes the most of both approaches.

3.1. The Scrum-based model – main assumptions and indicators

The Scrum-based model [Sienkiewicz, Maciaszek 2011] described here, considers Scrum undertakings and practices placed in "Job/Performer Level", which are essential to achieving the goals propagated from "Process Level". Therefore, we skip the ancillary roles of managers and stakeholders due to the low importance for further considerations. These roles are placed at other levels (the term of "Organizational Levels" introduced in Sienkiewicz, Maciaszek [2011]).

The impact of Third Party Services on the core Scrum roles is essential and should not be omitted. In our approach, we distinguish between four types of teamwork, which represent dependency-minimizing layers of holons proposed in our Scrum-based model. Due to that and for the sake of proper adaptation, we propose to add another core role excluded from the stakeholders group and involved in the entire software development process:

- The Third Party Service Provider (*S*) – an organization or individual provides your organization with specialized Third Party Services (e.g., lawyers, accountants, coaches, consultants, translators, internal and external service providers, etc.). This new core role refers to Third Party Services (i.e., Internal and External).

A Network Organization, understood as a network of intercommunicating elements, leads to an exponential growth of communication paths with the addition of new elements. Due to that in [Sienkiewicz, Maciaszek 2011] we proposed a simple three-layer model of a holarchy (as opposed to a network), where the default core roles (i.e., *PO* – Product Owner, *SM* – Scrum Master/s, *ST* – Scrum Team/s) are placed in the first two layers and the new core role *S* (i.e., Third Party Service provider) is placed in the lower layer.

We believe that highlighting the possible multiplicity of dependencies between layers of holons (i.e., many kinds of teamwork between Scrum players) and excluding a new core role *S* as an entity placed in a separate layer, is crucial for the success of the Scrum performed in Network Organizations. By highlighting the roles and dependencies in the software development process, we follow one of the Kanban's principles (i.e., "make process policies explicit"). We believe that building a shared comprehension of a process is possible only when all stakeholders have a common explicit understanding of the process.

Additionally, in Sienkiewicz, Maciaszek [2011] we proposed to change the following artifacts because they work better in a Network Organization:

- Task-feasibility instead of time-estimation – means that we should skip using formal time-estimates and try to commit only those User Stories that we are able to deliver before the next demo session. This change helps us to limit commitments which we are not able to provide.
- Report Meeting instead of Sprint Review Meeting – because regular Sprint Planning sessions involve many resources (i.e., a lot of participants), we propose to limit participants only to representatives of the customer and the team. In our opinion, that kind of meeting should be held more frequently than Sprint Review Meeting (e.g., every week) in order to improve the information flow between the customer and the team.
- Planning on demand instead of Sprint Planning Meeting – because we skip time-estimations we can propose to limit the number of Sprint Planning Meetings and hold them only if really needed (i.e., when the customer needs help from the team because he or she is not able to prioritize Product Backlog without the additional team's expertise).

The proposed changes arise from the fact that in our approach [Sienkiewicz, Maciaszek 2011] we adopted a holonic view and highlighted the variety of teamwork existing between layers of holons.The Scrum-based model and proposed novel artifacts and rules are good examples of using a scientific approach to implement continuous, incremental and evolutionary changes of work, workflow and process, as pointed out in one of the Kanban principles (i.e., "improve collaboratively the process").

The original Scrum uses only one metric usually presented as a burn-down chart (i.e., time-estimation of the amount of remaining work that needs to be done versus the amount of "User Stories" or tasks that are set as "Done!" in Sprint Backlog). We propose to use three additional "Key Performance Indicators (KPI)", which help better manage software development in a Network Organization:

- Reliability – to measure if the team is delivering what they said they would. We compare the difference between the amount of committed Story Points (c_i) and delivered Story Points (d_i) as shown in Formula (1). The values might be presented as the percentage of reliability calculated per Sprint (R_i):

$$R_i = \frac{C_i}{d_i} * 100\%. \tag{1}$$

- Productivity – to measure project velocity. We measure the amount of fixed bugs (b_i) and newly implemented requirements (s_i) as shown in Formula (2). The value of productivity (P_i) should be calculated after each Sprint:

$$P_i = b_i * s_i. \quad (2)$$

- Effectiveness – to measure the effectiveness of the testing service. The measure includes the amount of defects delivered to the customer. Based on this KPI, we can calculate the effectiveness of the internal testing service (as shown in Formula (3)), by measuring the ratio between all found defects (a_i) and those found by external *S* providing complementary testing (e_i). This shows the effectiveness (E_i) of the software development team and the testing services.

$$E_i = \frac{a_i - e_i}{a_i} * 100\%. \quad (3)$$

The introduced KPI-s are crucial to maintaining customer satisfaction. The data required for those calculations should be collected at the end of each Sprint.

One of the Kanban practices says that we should try to "manage flow" to ensure that we avoid all possible waste. Therefore, the KPI-s proposed by us may serve as the metrics that fully cover this Kanban principle. For measuring the flow we can use equations introduced by the traditional Kanban approach known from process management; however, we believe that Reliability (1) and Productivity (2) calculated per-sprint and presented in the timeline are enough for measuring the lead-time.

The Effectiveness (3) proposed as a third metric could be used as one of the indicators that help limit WIP (i.e., "Work In Progress") known as another Kanban practice.

In the next section, we focus on extending our approach by adding Kanban practices to our Scrum-based model.

3.2. Kanban – as an addition to the Scrum-based model

Analyzing the Scrum-based model in terms of Kanban principles, we noticed that Kanban's practices confirmed (more or less) our approach, proposed in Sienkiewicz, Maciaszek [2011].

We take advantage of the Kanban software development method and look at the Kanban-board. The whiteboard used in the proper way brings maximum visualization to managing software development by showing in an easy way the whole flow of tasks, requirements or user stories. Therefore, we can say that also this Kanban principle (i.e., "visualize the workflow") is covered by our Scrumban approach as well as the traditional approach.

The difference between the whiteboards illustrated in Figure 1 is one row added in the second board. This one row is crucial for Kanban's software development method (i.e., row with WIP).

This one small improvement significantly improves the visibility of the possible workload of the team in all stages. There is no one clear rule of how to measure WIP in software development. Due to that, we propose to use The Effectiveness (1) as an

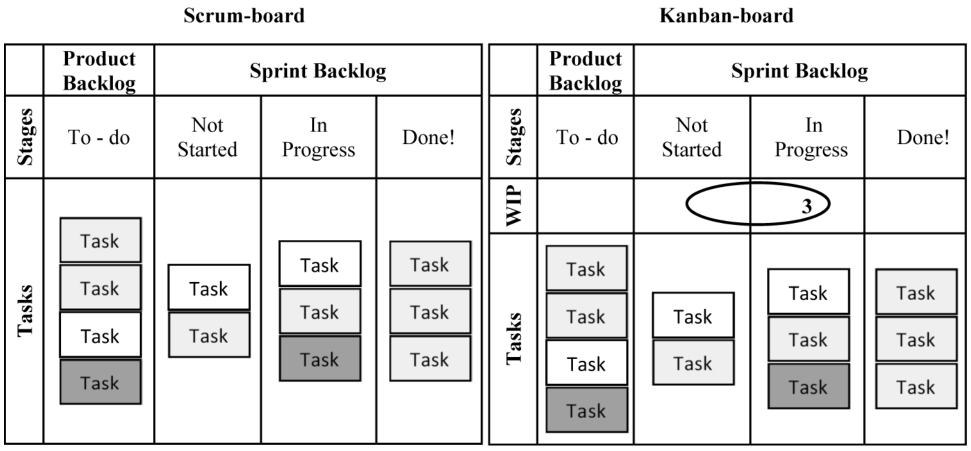

Figure 1. Example of the Scrum-board and the Kanban-board – similarities and differences

Source: author's own study.

indicator measured at the end of each cycle. We believe that this will help better adjust the value of WIP for each stage.

Making this small improvement will help avoid bottlenecks in all stages. A properly set and very visible WIP proposed per stage will decrease the number of analyzed tasks which stayed too long in the implementation phase. Reaching the limit of WIP in any of the stages is an alarm to all stakeholders that help is needed because some obstacles are blocking the whole flow.

Scrum advocates might argue that the traditional Scrum approach has a mechanism for managing "impediments" (i.e., Impediment Backlog includes a list of obstacles). We agree with that opinion; however, we also point out that the traditional Scrum approach says nothing about the limit of "impediments", which is crucial for managing software development process in Network Organizations, where a delay caused by one stakeholder implicates delays in other stakeholders' schedules.

4. Related Work

In the last 30 years, many approaches have been proposed to software application development. We distinguish three general life cycle models (i.e., heavyweight, middleweight and lightweight) which include several different software development methods (e.g., "Code and Fix" [Guntamukkala, Wen, Tam 2006; Royce 1970], waterfall model, spiral model, etc.). It is hard to say which model is better than another due to the fact that different projects require different approaches. On the basis of the research results and experience [Cocburn 2000; Nandhakumar, Avision 1999; Schwaber 2002; 2004; Lindvall et al. 2002], we can assume that the best choice for software development projects starting from scratch and executed in a continuously changing environment is a strongly adaptive methodology known as Agile. By Agile we understand several prescriptive methods (e.g., Scrum, eXtreme Programming, ASD, etc.) [Cocburn 2000; Schwaber 2002; 2004].

In our work we wanted to add some novel artifacts and rules to adapt existing Agile methods to work better in a Networked Organization. We agree with Scrum advocates that using time-boxed delivery cycles (i.e., Sprints), visualization of the project scope (e.g., Product Backlog drawn on Scrum-board, Kanban-board), prescribed roles (e.g., Scrum Master, Product Owner, Scrum Team and proposed Third Party Service Provider role [Sienkiewicz, Maciaszek 2011]) as well as following Scrum rules and organizational meetings (e.g., daily meetings, retrospective meetings) are necessary for the success of a project.

In the literature, a few researchers have already studied the way of adapting the Agile practices in Virtual Organizations and Virtual teams. Usually, Agile software development methods are introduced as a set of principles that need to clarify many different interpretations of Agile Manifesto [Beck et al. 2001]. However, it is very difficult to determine what exactly Agile methodology is and how it should be introduced in Network Organizations.

In our research we focus mostly on the main representatives of Agile software development methodology, and due to the lack of method dedicated to managing distributed software development in Network Organizations, we have selected one (i.e., Scrum) as the best representative.

Obviously, we agree with Scrum advocates that following the Agile Manifesto principles is possible only because Scrum defines precisely the essential roles, principles and artifacts, which makes the method very prescriptive for managing software development. In addition to the traditional Scrum, we propose to add a new role (i.e., Third Party Service provider – *S*) and some extra rules for adapting Scrum and Third Party Services to a Network Organization.

An exploratory case study by Hovorka, Larsen [2006] examined the influence of a Network Organization environment on the ability to develop Agile adoption practices. The authors "investigate the interactions between network structure, social information processing, organization similarity (homophily), and absorptive capacity during the adoption of a large-scale IT system in two network organization environments" [Hovorka, Larsen 2006]. In this paper, we propose an approach that is more detailed. We do not limit ourselves to Agile as a set of good practices, and we propose an analysis that is even more detailed by focusing on one selected method (i.e., Scrum).

By referring to the Network Organization and Third Party Services, we determine how they interrelate with a Scrum and Agile environment. The publication by W. Cellary and W. Picard achieves agility and pro-activity by introducing the model of Collaborative Network Organization that provided a stimulus for us to reflect on the Third Party Services Provider role that is simultaneously a part and a whole in a Network Organization.

5. Conclusion

In this paper we have proposed the "Scrumban" software development method which combines the Scrum-based model and Kanban principles as one flawlessly working software development method.The proposed approach is dedicated to Agile environment with Third Party Services introduced in a Network Organization.

By extending our previous research from Sienkiewicz, Maciaszek [2011], we highlighted the common parts of Kanban and Scrum-based approaches [Sienkiewicz, Maciaszek 2011]. Because there is no easy way to adapt pure Scrum or Kanban software development method to work in a Network Organization, we believe that the proposed approach can serve to advance research and help find the best solution.

References

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

Beck K., Beedle M., van Bennekum A., Cockburn A., Fowler M., Grenning J. et al., 2001, retrieved from Manifesto for Agile Software Development: http://agilemanifesto.org/

Cocburn A., Selecting a project's methodology, *IEEE Software* 2000, vol. 4 (17), pp. 64–71.

Guntamukkala V., Wen J., Tam M.J., An empirical study of selecting software development life cycle models, *Human Systems Management* 2006, vol. 4 (25), pp. 268–278.

Holweg M.,The genealogy of lean production, *Journal of Operations Management* 2007, vol. 25 (2), pp. 420–437.

Hovorka D.S., Larsen K.R., Enabling Agile adoption practices through network organization, *European Journal of Information Systems* 2006, vol. 15 (2), pp. 159–168.

Ikonen M., Leadership in Kanban software development projects: A quasi-controlled experiment, [in:] *Proceedings of the 1st International Conference on Lean Enterprise Software and Systems (LESS) 2010*, 2010, pp. 85–98.

Kniberg H., Skarin M., *Kanban and Scrum – Making the Most of Both*, 11th edition, Crisp, 2009.

Lindvall M., Basili V., Boehm B., Costra P., Dangle K., Shullm F. et al., Empirical findings in Agile methods, [in:] *Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods – XP Agile Universe 2002*, Springer-Verlag, London 2002, pp. 81–92.

Nandhakumar J., Avision J., The fiction of methodological development. A field study of information system development, *Information Technology and People* 1999, vol. 2 (12), pp. 176–191.

Norrmalm T., *Achieving Lean Software Development: Implementation of Agile and Lean Practices in a Manufacturing-Oriented Organization*, Stockholm 2011.

Royce W., Managing the development of large software systems, [in:] *Proceedings of the 9th International Conference on Software Engineering ICSE '87*, Los Angeles 1970, pp. 1–9.

Schwaber K., *Agile Project Management with Scrum*, Microsoft Press, Redmond, Washington 2004.

Schwaber K., *Agile Software Development with Scrum*, Prentice Hall, Upper Saddle River, New Jersey 2002.

Shingo S., *A Study of the Toyota Production System*, (Rev. Sub Edition ed.), Productivity Press, 1989.

Sienkiewicz L.D., Maciaszek L.A., Adapting Scrum for Third Party Services and Network Organizations, [in:] *Proceedings of the Federated Conference on Computer Science and Information Systems, IEEE digital library*, Szczecin 2011, pp. 329–336.

Womack J., Jones D., Ros D., *The Machine That Changed the World*, 1st edition, Harper Perennial, Rawson Associates, New York 1991.

SCRUMBAN – KANBAN JAKO UZUPEŁNIENIE METODY *SCRUM* UŻYWANEJ DO WYTWARZANIA OPROGRAMOWANIA W ORGANIZACJI SIECIOWEJ

Streszczenie: Duża liczba publikacji naukowych przedstawia metodę *Scrum* jako najlepsze podejście do wytwarzania oprogramowania. Jednakże oryginalna metoda *Scrum* nie jest wystarczająca do zarządzania pracą w środowisku Agile'owym wewnątrz organizacji sieciowej. Z tego powodu przeanalizowaliśmy podejście Kanbanowe i nasz model pracy bazujący na Scrumie [Sienkiewicz, Maciaszek 2011] w celu wyodrębnienia dobrych praktyk z obu podejść, co pozwala nam zaproponować podejście lepiej dostosowane do zarządzania wytwarzaniem oprogramowania w rozproszonym Agile'owym środowisku (tj. w organizacji sieciowej).

Słowa kluczowe: *Scrum*, usługodawcy wewnętrzni i zewnętrzni, organizacja sieciowa, *Scrumban*, *Kanban*, *Agile*, kluczowe wskaźniki wydajności, tworzenie oprogramowania.