Umeå University
Department of Computing Science

Software Engineering 15 p
5DV151

OU 1 - Scrum and Lean Software Development

Submitted	2017-03-24
Author:	Lorenz Gerber (`dv15lgr@cs.umu.se` `lozger03@student.umu.se`)
Instructor:	Jonny Pettersson / Jonas Andersson

1 Introduction

This report has three aims: First to describe the two software processes models ‘Scrum’ and ‘Lean Software Development’ (LSD). Then to devise a number of criteria that can be used to compare software process models and finally to compare ‘Scrum’ and ‘LSD’ according to these criteria.

2 Description of the Models

Both Scrum and LSD are so called agile methodologies. ‘Agile’ methods are such that adopt the ‘Agile Manifesto’ a declaration of values forwarded by a group of software evangelists that envisioned a more light-weight and flexible development process [1]. It is noteworthy that the fathers of ‘Scrum’ are both co-authors on the ‘Manifesto’ which was devised six years after their publication of the Scrum white-paper [5]. According to their homepage, the authors of the LSD white-paper are both prominent agile evangelists themselves [2].

2.1 Scrum

Scrum as a software process model has been described in 1995 by Ken Schwaber and Jeff Sutherland in a OOPLSA (Object-Oriented Programming, Systems, Languages & Applications) proceedings article [5]. It took inspiration from earlier work of two researchers active in product marketing strategies [6]. According to the scrum guide, an official white paper of the method written by it’s inventors Schwaber and Sutherland, Scrum is based on empirical control theory [7]. Scrum is today the probably most adopted agile project management method in the software industry [4, p. 86].

In reference to the *Scrum Guide*, it follows a short description of the individual participants, the events and artifacts of the Scrum methodology [7]. The scrum guide defines the **Scrum Team** to consist of three different groups: First, there is the **Product Owner**. He could be the customer, a representative of the customer or simply somebody that knows best about the product to be developed in terms of requirements. In the scrum guide, it is said that the product owner should ideally be co-located with the rest of the Scrum Team. The *Development Team* is a cross-functional group of developers consisting of between three and nine members. Obviously, the development team is the center of all productivity. Within the development team one member is the **Scrum Master**. He should be an experienced scrum practitioner as his main duty besides being a normal development team member is to coach and advise the rest of the team on ‘living’ the scrum methodology.

The scrum guide describes five events. These are the organizational main activities in which the members of a Scrum Team participate. All activities are centered around the **Sprint**. The sprint is a two to four month long cycle in which the main development work takes place. Ahead of a sprint is the **Sprint Planning**. As the name suggests, here the work preparation for the sprint is conducted. The whole scrum team including the product owner takes part here. A sprint planning for a one month lasting sprint shall not be longer than eight hours. The two main topics for a planning session are ‘what can be done during the next Sprint’ and ’how will the work get done’. During the actual sprint period, informal work meetings, the *Daily Scrum* are held every morning during a proposed fifteen minutes. Here only the development team and the scrum master take part. It has three main topics: ‘What was done yesterday’, ‘what will be done today’ and ‘are there any problems that prevent the team from reaching it’s goal’.

After a sprint, a **Sprint Review** is arranged where again analogous to the Sprint planning the whole Scrum Team including the stakeholders takes part. This evaluation is focused on the product developed but also on how the product relates to it's environmental parameters.

Besides the sprint review, at the end of a sprint the **Sprint Retrospective** takes place. This meeting, in contrast to the sprint review focuses not on the actual product development but on the Development team itself. It shall help to work on improving the work process itself and is proposed to take not more than three hours. After the sprint retrospective, the next cycle starts again with a Sprint planning event.

Besides the actors and events, there are a three artifacts specific for the scrum methodology. The first is the **Product Backlog**. This is a list with product features, requirements and engineering improvements. It is maintained by the product owner. During the sprint planning event, tasks from the product backlog are chosen for the next sprint. All chosen tasks for one sprint cycle define the second artifact, the **Sprint Backlog**. After a sprint, the finished tasks from the sprint backlog plus the tasks finished in earlier sprints together comprise the *increment*: A potentially shippable version of the product.

2.2 Lean Software Development, LSD

Lean software development has been presented for a large crowd in a book written by the agile evangelists Poppendieck and Poppendieck [3]. They took their inspiration from industry and car manufacturing where lean production was already widely accepted, pioneered and developed by the Japanese car manufacturer Toyota [8].

The following description of LSD was mostly condensed from the introduction chapter of the white-paper book defining Lean Software Development [3]. First, the terms **Lean Principles** and **Practices** are coined. Lean principles are today mostly known from the manufacturing industry. It is pointed out that principles should be seen as universal high-level guidelines. In LSD seven lean principles are devised, they will be further described below. The authors then describe that practices are the actual events which should embody and translate the high-level guidelines into action. However, according to them, it is not feasible to transfer 'best practices' from other fields to software development as practices are very dependent on the context. Hence, the authors suggest that Lean principles should be used to define and further refine agile practices which have already shown to be of use in software engineering. They further suggest that lean principles can be used as a theoretical framework to reason and explain why agile practices work.

Below follow the seven Lean Principles from the book and a short account of how they are set to work in Software Engineering.

1. Eliminate Waste
   Anything that is not directly adding value to the product is seen as waste. In software development some sources of waste mentioned are 'partially done work', 'extra features' that are not requested, 'task switching' or 'waiting'.

2. Amplify learning
   The authors argue that learning is at the very heart of software development. They devise therefore several practices that shall improve and amplify the learning processes. Some of them are to generally increase feedback loops in the work processes another tool is iteration in planning and development processes.

3. Decide as late as possible
   This principle relates to keeping options along the development way to be able to adapt to changed external conditions. In software development, this can be translated

into such practices as modular code, using clear defined interfaces and writing generic code. This allows to adapt to new conditions at later stages of a project.

4. Deliver as fast as possible
   Here the authors argue, that the preconditions for fast delivery of a product, which obviously is a desired property, is to work on short iterative cycles. This is said to take pressure from developers as each task is small and manageable on the other side it allows for fine grained control and adaption throughout the process.

5. Empower the team
   This chapter and principle is strongly based in scientific studies about organization theory. The punchline is that loosely managed teams in a rather flat hierarchy provide the most productive work unit.

6. Build integrity in
   This principle is about property of the developed product. They define *perceived integrity* as *...the totality of the product achieving a balance of function, usability, reliability and economy that delights the customer* and *conceptual integrity* as the fact that the *...systems central concepts work together as a smooth, cohesive whole..* A lot of the practices devised to obtain these principles seem to stem from research in human computer interaction (HCI). Additional to practices from HCI, software engineering practices such as refactoring and testing are mentioned.

7. See the whole
   This chapter describes the principle of how the above production unit principles integrate into the whole which here is the company. The largest part of the chapter describes how different form of contracts fit into the lean agile production system.

3 Evaluation Criteria

To compare the two agile methodologies 'Scrum' and 'Lean Software Development', some pre-evaluations have to be conducted that will allow to choose a suitable reference of comparison. While writing the summaries of the two methodologies, it became obvious that the approaches of the two methodologies or frameworks are quite different. LSD is more of a theoretical framework with a number of examples who the high-level principles can be implemented as agile practices. The Scrum methodology on the other hand is a very concrete, practical set of instructions for how to perform and organize software development. Hence, the criteria to compare the two methods need to reflect this difference. Below follows a number of criteria that will be used to compare the two methodologies.

- Which white-paper was chosen for the comparison. It seemed most relevant to choose for both methodologies a recent document.
- On which **Level of Abstraction** does the methodology operates: Is it high-level theoretical framework, a collection of pure applied practices or somewhere in between.
- Who is the potential **target group** of the white paper describing the methodology.
- What is the main **center of focus** in the methodology. Does it relate mostly to tangible or intangible subjects. Tangible ones would be the actual developer, or the group of developers while intangible subjects are abstract processes in general.

- Does the framework describe how it **interfaces** with it's non-agile surrounding. As described in the course textbook about software engineering, one often weak point of agile methodologies is how they can be embedded in a non-agile environment [4, p.91]. Does the respective framework provides solutions to this by default.
- What is the **organizational unit** the framework relates to. Does the framework relates to the whole company or is it applicable mostly to a single small production unit, the developer or development team.
- Which **agile practices** and development techniques does the methodology implements.
- Does the methodology provides a **step-by-step implementation** description. How should a conventional working company transform to the agile

The evaluation of the above criteria is carried out in the following section while an overview of the evaluation is presented in table 1.

4 Comparison Scrum vs LSD

Despite both Scrum and LSD being agile methodologies, they seem to have quite a different scope and target audience. This becomes already clear purely from the shape factor of the white-paper: A 17 page pamphlet for Scrum and a 240 page book for LSD. Scrum seems to be like a hands-on guide of how to actually work in an agile framework. The white paper is short and gives only few hints to the theory behind. Lean Software Development on the other side chooses much more an academic approach of explaining and trying to connect theory with practical knowledge. The theoretical background in LSD stems from ‘Lean Production’ and as such from the Toyota Production System. In Scrum, the theoretical background is much less clear and not much highlighted. There would most likely be secondary literature that describes the theoretical background of scrum in more detail, but in the white-papers it is absent. This seems to be by purpose and is as a statement in itself.

Scrum puts the agile principle *Individuals and interactions over processes and tools* in the center while LSD is, maybe due to it's history, to a large extent concerned with processes.

Being almost a pure practical description, the scrum guide does not mention much about how a scrum team should interface with non a non agile surrounding. Here the LSD methodology gives more clues. Maybe also because this usually happens on a higher organizational level. It will rather be the managers, the potential target group of the LSD literature, that have to cope the integration of agile development teams into a more traditional managed corporate environment.

Without having a real reference for the statement, it seems that 'Scrum' could be termed as a bottom up approach for implementing agile operations. The LSD description in contrary seems more to describe top-down implementation. This would at least fit together with the identified target group of the respective methodologies.

References

[1] Manifesto for agile software development. `http://agilemenifesto.org`, 2001. accessed: 2017-03-24.

Table 1 Comparison of the two agile methodologies ‘Scrum’ and ‘Lean Software Development’

	Scrum	LSD
Form of Whitepaper	Pamphlet [7] 17 pages, precise description of the practical steps with a clear beginning and end.	Book [3] 240 pages, varying levels of details, collection of topics
Level of Abstraction	complete package of practices, usually presented with little or no theoretical justifications	selection of high-level principles, many based in well researched fields with concrete examples of practical tools to obtain results
Target Group	practitioners, developers	project leader, team leader, technology managers
Center of Focus	focuses mostly on the production team	focuses mostly on processes, how they can be optimized
Interfaces	does not explicitly mention interface or embedding within larger organization	coming from practical production industry, LSD provides several clues of how to implement agile practices into a larger organization
Organizational Unit	clear focus on the development team	discusses more general
Agile Practices	implements: user stories, on-site customer, small releases, incremental planning, collective ownership	most of the current agile practices are mentioned in the book as tools, they are however not chosen as a specific set, rather as individual tools for certain situations
Step-by-step Implementation	precise, easy to understand practical description of how to ’do’ scrum	not really the aim, the ’tools’ description in the LSD book are more on a example basis, the methodology remains close to principles even for the proposed (practical) tools

[2] Tom Poppendieck and Mary Poppendieck. The lean mindset. http://www.poppendieck.com/people.html. accessed: 2017-03-24.

[3] Tom Poppendieck and Mary Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison Wesley, 2003.

[4] Ian Sommerville. *Software Engineering*. Pearson, Essex, England, 2016.

[5] Jeffrey Victor Sutherland and Ken Schwaber. Business object design and implementation: Oopsla '95 workshop proceedings. *OOPSLA '95 workshop proceedings*, 1995.

[6] Hirotaka Takeuchi and Ikujiro Nonaka. New new product development game. *Harvard Business Review*, 1986.

[7] The scrum guide. http://www.scrumguides.org/scrum_guide.html, 2016. accessed: 2017-03-23.

[8] The origin of the toyota production system. http://www.toyota-global.com/company/vision_philosophy/toyota_production_system/origin_of_the_toyota_production_system.html, 2017. accessed: 2017-03-24.