

Integrating the Scrum Framework and Lean Six Sigma

Anacleto Correia[1(✉)], António Gonçalves[1], and Sanjay Misra[2]

[1] CINAV, Alfeite, 2810-001 Almada, Portugal
cortez.correia@marinha.pt
[2] Covenant University, Ota, Ogun State, Nigeria

Abstract. Lean Six Sigma approach has been a successful manufacturing quality improvement tool. In the last decades, it has helped many services and product industries to achieve better performance and higher results. No wonder that the approach has aroused the interest of the software development industry. As result, some software companies have been trying to adopt Lean Six Sigma in their development processes, specifically in conjunction with agile methods. Based on some of these experiences this paper intends to propose the integration in the Scrum method the quality procedures defined on the Lean Six Sigma (LSS). By complementing the current generic quality procedures of Scrum method with more formalized and well-proven quality measures from LSS we intend to improve the efficacy and efficiency of Scrum.

Keywords: Agile methodology · Scrum · Lean Six Sigma · Software quality

1 Introduction

Among software development methodologies, agile methodologies, anchored by the values and the principles of the Agile Manifesto, claim their suitability to better manage software development lifecycle. The manifesto highlights a set of values an agile methodology should embody, inter alia, the relevance of individuals and their interactions, the need of a continuous delivery of working software, collaboration with customer and the fast response to change. Among these several lightweight development methods, there is Scrum.

In IT related domains, Scrum has been used to develop software, hardware, embedded software, networks of interacting function, autonomous vehicles. Scrum has also been widely used for products, services, and the management of organizations, such as, schools, government, marketing, or the operation of a process, and almost everything we use in our daily lives, as individuals and societies. As happened with other agile methods, Scrum provides practices that help companies building the right product, at a certain moment, and empowering the teams to continuously redesign the released products, optimizing its value throughout development, and allowing it to address, in each moment, the customers' expectation. Therefore, the main aim of the methodology is to deliver what is needed at appropriate time in each iteration of the development cycle.

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11623, pp. 136–149, 2019.
https://doi.org/10.1007/978-3-030-24308-1_12

Scrum methodology for software development can be successfully applied in context of unclear customer's requirements, tight deadlines and budgets. However, besides the benefits associated to the use of Scrum, and agile technologies in general, concerns can be raised regarding the procedures embedded in the methodology that could ensure an effective measurement of gains in quality, productivity and business satisfaction.

This paper is an attempt to include in the empirical approach for software development methodology Scrum, prescriptive techniques for defining and improving products, processes and services with a focus on reduction of defects and failures, on variation and waste elimination, prioritizing, in a planned and objective way, the achievement of quality and financial results, found in Lean Six Sigma (LSS) methodology.

This paper is organized as follows: in the next section we describe the fundamentals of the Scrum framework, either in its static perspective, with a light ontology depicting its main concepts and their relationships, as well using a dynamic representation of a process model representing the main activities developed iteratively. In Sect. 3 the fundamentals of the Lean approach are described, as well as the Six Sigma method for quality. Both approaches are merged in the Lean Six Sigma method, which are presented also according the same two perspectives previously used (static and dynamic). The section four synthetizes how the integration between Scrum and Lean Six Sigma can be achieved. The work ends by drawing some conclusions and foreseeing future work.

2 Scrum Framework

2.1 Overview

Scrum is a process framework that can embody within it, various processes and techniques on complex products. Scrum intends to give efficacy to product management and development techniques in order to continuously improve the product, the team, and the working environment [1]. Scrum was initially developed for managing and developing products. The method was proved especially effective in iterative and incremental knowledge transfer.

The essence of Scrum is a small team of people. The individual team is highly flexible and adaptive. These strengths continue operating in single, several, many, and networks of teams that develop, release, operate and sustain the work and work products of thousands of people. They collaborate and interoperate through sophisticated development architectures and target release environments.

2.2 Scrum Model

In this section we formalize the main concepts of Scrum [2], as well the relationship among them, through the light ontology presented in Fig. 1.

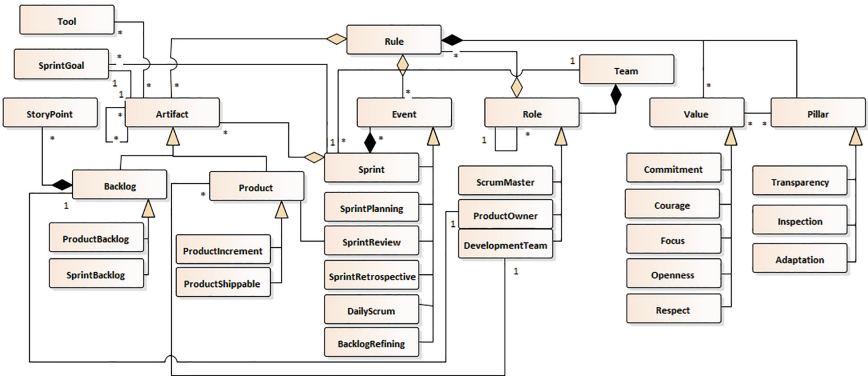


Fig. 1. The model of Scrum.

Each component within the Scrum framework serves a specific purpose. The *Rules* of Scrum bind together the *Events* and *Artifacts*, and govern the relationships and interaction between them, anchored by the *Values* and *Pillars* of the framework.

The Scrum's *Values* of *Commitment*, *Courage*, *Focus*, *Openness* and *Respect* should be embodied and lived by the *Team*. Successful use of Scrum depends on people becoming more proficient in living these five values.

The *Team* has self-organized and cross-functional members each of one with specific *Role*. They tend to choose the best way to accomplish their work, rather than being directed by others outside the team, and rely on having internally all competencies needed to accomplish the work without depending on external team's members. The team deliver a *Product* iteratively and incrementally, maximizing opportunities for feedback. Incremental deliveries (*ProductIncrement* or *ProductShippable*) ensure that a working product version is always available after each *Sprint*.

Relate with the *Values*, the Scrum's *Pillars* of *Transparency*, *Inspection*, and *Adaptation* are instantiated to bring trust for stakeholders. Through *Transparency* it is required the definition of aspects, based on agreed standards, in order that stakeholders can share a common understanding of the process. Scrum users must frequently inspect *Artifact*s to detect undesirable variances. The *Inspection* should be performed by skilled inspectors at defined point of work. If there are aspects of the process that deviate outside acceptable limits, conducting to unacceptable *Product*, the process or the *Product* being built must be subject of *Adaptation*, i.e. to an adjustment so deviation can be minimized.

The *Roles* played by members of the *Team* are the *ProductOwner*, the *DevelopmentTeam*, and a *ScrumMaster*. The *ProductOwner* is responsible for maximizing the value of the product resulting from work of the *DevelopmentTeam*.

The *ProductOwner* is the person responsible for managing the *ProductBacklog* and represent the desires of those wanting a *Product*. The *DevelopmentTeam* consists of people who do the work for delivering a *ProductIncrement* at the end of each *Sprint*. Only members of the *DevelopmentTeam* create the *ProductIncrement*. The *DevelopmentTeam* is structured and empowered to organize and manage its own work in order

to optimize the overall efficiency and effectiveness. The *ScrumMaster* is responsible for promoting and supporting Scrum, which is attained by helping the *Team* understanding Scrum theory, practices, rules, and values.

The *Sprint*, the core of Scrum, is a maximum duration activity during which a useable and potentially releasable *Product* is created. *Sprint*s have same durations throughout a development effort, with a new *Sprint* starting immediately after the conclusion of the previous. Prescribed *Events* are used in Scrum to create regularity and to minimize the need for meetings not defined in Scrum. All events in Scrum are time-boxed. A *Sprint* contains as prescribed events for inspection and adaptation the *SprintPlanning*, *DailyScrum*, the *SprintReview*, and the *SprintRetrospective*. The *Sprint* is the container of all these events as well as the regular *DevelopmentWork*. Each event in Scrum, other than the *Sprint* itself, is an opportunity to inspect and adapt a functionality, therefore, specifically designed to enable *Transparency*, *Inspection* and *Adaption*.

The *Sprint* is a kind of a mini-project that aims to accomplish a *SprintGoal*, which is embodied by an *Artifact*, that can be planned, designed, built, and be the result *ProductIncrement*.

In Scum, the *Backlog* is a kind of *Artifact*, that consists of an ordered list of what is needed in the *Product*. It is the repository of requirements for any changes to be made to the *Product*. The *ProductOwner* is responsible for the *ProductBacklog*, including its content, availability, and ordering. The *ProductBacklog* evolves as the product and the environment in which it will be used evolves since the initially requirements. The *ProductBacklog* constantly changes to identify what the product requirements to be appropriate, competitive, and useful.

The *ProductBacklog* lists all features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the product in future releases. *ProductBacklog* items have the attributes of a description, order, estimate, and value. *ProductBacklog* items often include test descriptions that will prove its completeness when *ProductShippable*. As a product is used and gains value, and the marketplace provides feedback, the *ProductBacklog* becomes a larger and more exhaustive list. Requirements never stop changing, so a *ProductBacklog* is a living *Artifact*. Changes in *StoryPoints* (business requirements), market conditions, or technology may cause changes in the *ProductBacklog*.

The *ProductIncrement* is the sum of all the *ProductBacklog* items completed during a *Sprint* and the value of the *ProductIncrement*s of all previous *Sprint*s. At the end of a *Sprint*, the new *ProductIncrement* must be accepted, which means it must be in a useable condition and meet the *Team*'s terms of acceptance. So, the *ProductIncrement* is a body of inspectable, done work at the end of the *Sprint* towards the *SprintGoal*.

Multiple *Team*s often work together on the same *Product*. One *ProductBacklog* is used to describe the upcoming work on the *ProductShippable*. A *ProductBacklog* attribute that groups items may then be employed.

2.3 Scrum's Process Framework

After the static representation of Scrum made in previous section, Fig. 2 presents a dynamic view of the same process through a BPMN process model. There, one can see

that after collecting the business requirements and prioritize them, the *ProductOwner* (bottom lane) discusses the objective that each *Sprint* should achieve and the *ProductBacklog* items that, when the *Sprint* is completed, allow to achieve the *SprintGoal*. The work to be performed in the *Sprint* is planned at the *SprintPlanning*. This plan is usually created by the collaborative work of the entire *Team*.

The *ScrumMaster* (middle lane) ensures that the *SprintPlanning* takes place and whether attendants understand its purpose. The *DevelopmentTeam* (top lane) works to forecast the functionality that will be developed during the *Sprint*. The entire *Team* collaborates on the work of the *Sprint*. The input to the *SprintPlanning* meeting is the *ProductBacklog*, the latest *ProductIncrement*, projected capacity of the *DevelopmentTeam* during the *Sprint*, and past performance of the *DevelopmentTeam*. The number of items selected from the *ProductBacklog* for the *Sprint* is solely up to the *DevelopmentTeam*. Only the *DevelopmentTeam* can assess what it can accomplish over the upcoming *Sprint*. During *SprintPlanning* the *Team* also crafts a *SprintGoal*. The *SprintGoal* is an objective that will be met within the *Sprint* through the implementation of the *ProductBacklog*, and it provides guidance to the *DevelopmentTeam* on why it is building the *ProductIncrement*.

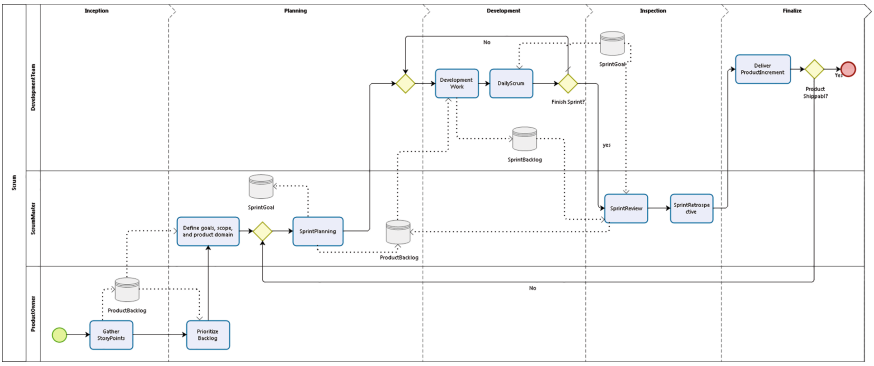


Fig. 2. The Scrum's process framework.

Having set the *SprintGoal* and selected the *ProductBacklog* items for the *Sprint*, the *DevelopmentTeam* will transform the requirements into a working *ProductIncrement* during the *Sprint*. The *ProductBacklog* items selected for the *Sprint* plus the plan for delivering them is called the *SprintBacklog*.

The Work starts by designing the system and may be of varying size, or estimated effort. However, enough work is planned during *SprintPlanning* for the *DevelopmentTeam* to forecast what it believes it can do in the upcoming *Sprint*. Work planned for the first days of the *Sprint* by the *DevelopmentTeam* is decomposed by the end of the *SprintPlanning* meeting. The *DevelopmentTeam* self-organizes to undertake the work in the *SprintBacklog*, both during *SprintPlanning* and as needed throughout the *Sprint*.

The *ProductOwner* can help to clarify the selected *ProductBacklog* items and make trade-offs. If the *DevelopmentTeam* determines it has too much or too little work, it may renegotiate the selected *ProductBacklog* items with the *ProductOwner*.

The *SprintGoal* is an objective set for the *Sprint* that can be met through the implementation of *ProductBacklog*. It provides guidance to the *DevelopmentTeam* on why it is building the *ProductIncrement*. It is created during the *SprintPlanning* meeting. The *SprintGoal* gives the *DevelopmentTeam* some flexibility regarding the functionality implemented within the *Sprint*. The selected *ProductBacklog* items deliver a coherent function, which can be the *SprintGoal*. The *SprintGoal* can be any other coherence that causes the *DevelopmentTeam* to work together rather than on separate initiatives.

As the *DevelopmentTeam* works, it keeps the *SprintGoal* in mind. In order to satisfy the *SprintGoal*, it implements functionality and technology. If the work turns out to be different than the *DevelopmentTeam* expected, they collaborate with the *ProductOwner* to negotiate the scope of *SprintBacklog* within the *Sprint*.

The *DailyScrum* is a time-boxed event, within the *Sprint*, for the *DevelopmentTeam* held every day. During the *DailyScrum* the *DevelopmentTeam* plans work for the next work day. This optimizes team collaboration and performance by inspecting the work since the last *DailyScrum* and forecasting upcoming *Sprint* work.

The *DevelopmentTeam* uses the *DailyScrum* to inspect progress toward the *SprintGoal* and to inspect how progress is trending toward completing the work in the *SprintBacklog*. Every day, the *DevelopmentTeam* should understand how it intends to work together as a self-organizing team to accomplish the *SprintGoal* and create the *ProductIncrement* by the end of the *Sprint*.

The *DevelopmentTeam* often meet immediately after the *DailyScrum* for detailed discussions, or to adapt and re-plan, the rest of the *Sprint*'s work. *DailyScrum*s improve communications and improve the *DevelopmentTeam*'s level of knowledge. This is a key inspect and adapt meeting.

A *SprintReview* is held at the end of the *Sprint* to inspect the *ProductIncrement* and adapt the *ProductBacklog* if needed. During the *SprintReview*, the *Team* and stakeholders collaborate about what was done in the *Sprint*. Based on that and any changes to the *ProductBacklog* during the *Sprint*, the *DevelopmentTeam* collaborate on the next things that could be done to optimize value. The result of the *SprintReview* is a revised *ProductBacklog* that defines the probable *ProductBacklog* items for the next *Sprint*. The *ProductBacklog* may also be adjusted overall to meet new opportunities.

The *SprintRetrospective* is an opportunity for the *Team* to inspect itself and create a plan for improvements to be enacted during the next *Sprint*. The *ScrumMaster* ensures that the event takes place and that attendants understand its purpose. The purposes of the *SprintRetrospective* are: (1) to inspect how the last *Sprint* went with regards to people, relationships, process, and tools; (2) identify and order the major items that went well and potential improvements; and, (3) to create a plan for implementing improvements to the way the *Team* does its work.

The *ScrumMaster* encourages the *Team* to improve the development process and practices to make it more effective for the next *Sprint*. During each *SprintRetrospective*, the *Team* plans ways to increase product quality by improving work processes or

adapting the definition of acceptance, if appropriate and not in conflict with product or organizational standards.

By the end of the *SprintRetrospective*, the *Team* should have identified improvements that it will implement in the next *Sprint*. Implementing these improvements in the next *Sprint* is the adaptation to the inspection of the *Team* itself. Although improvements may be implemented at any time, the *SprintRetrospective* provides a formal opportunity to focus on inspection and adaptation. Also, it is conducted a *ProductBacklog* refinement by adding detail, estimates, and order to items in the *ProductBacklog*. Higher ordered *ProductBacklog* items are usually clearer and more detailed than lower ordered ones, and therefore deemed for selection in the next *SprintPlanning*. *ProductBacklog* items usually acquire this degree of transparency through the above described refining activities. This is an ongoing process in which the *ProductOwner* and the *DevelopmentTeam* collaborate on the details of *ProductBacklog* items. So, during *ProductBacklog* refinement, items are reviewed and revised.

The *DevelopmentTeam* is responsible for all estimates. The *ProductOwner* may influence the *DevelopmentTeam* by helping it understand and select trade-offs, but the people who will perform the work make the final estimate.

At any point in time in a *Sprint*, the total work remaining in the *SprintBacklog* can be summed. The *DevelopmentTeam* tracks this total work remaining at least for every *DailyScrum* to project the likelihood of achieving the *SprintGoal*. By tracking the remaining work throughout the *Sprint*, the *DevelopmentTeam* can manage its progress.

Since Scrum is based on transparency decisions to optimize value and control risk are made based on the perceived state of the *Artifact*s. To the extent that transparency is complete, these decisions have a sound basis. To the extent that the *Artifact*s are incompletely transparent, these decisions can be flawed, value may diminish and risk may increase.

The *ScrumMaster* must work with the *ProductOwner*, *DevelopmentTeam*, and other involved parties to understand if the *Artifact*s are completely transparent. A *ScrumMaster* can detect incomplete transparency by inspecting the *Artifact*s, identifying patterns, and detecting differences between expected and real results. The *ScrumMaster*'s job is to work with the *Team* and the organization to increase the transparency of the *Artifact*s.

When a *ProductBacklog* item or a *ProductIncrement* is described as acceptable, the *Team* must understand and agree in what it means to ensure transparency. This is the definition of acceptance for the *Team* and is used to assess when work is complete on the *ProductIncrement*.

The same definition guide the *DevelopmentTeam* in knowing how many *ProductBacklog* items it can select during a *SprintPlanning*. The purpose of each *Sprint* is to deliver *ProductIncrement*s of potentially releasable functionality that adhere to the *Team*'s current definition of acceptable.

*DevelopmentTeam*s deliver a *ProductIncrement* with useable functionality every *Sprint*, so a *ProductOwner* may choose to immediately release it. If the definition of acceptable for a *ProductIncrement* is part of the conventions, standards or guidelines of the development organization, the *Team* must support it.

3 The Lean Six Sigma Approach

Lean [3] and Six Sigma philosophy had separate origins. While Six Sigma was started as an approach to reduce operational variation and defects, Lean thinking enabled elimination of waste and reduction of cycle time. The primary focus of Six Sigma is on effectiveness whereas Lean focuses on efficiency. As an integrated strategy, Lean Six Sigma has become a method for the excellence of systems by enabling the breakthrough improvement in every part of the system through process enablement, cost reduction, and increased benefits [4].

Typical Lean Six Sigma is aimed at reducing variation, defects, and waste, as well as improving process speed for existing processes and systems. These principles can be applied in a proactive manner to prevent defects and waste, while minimizing the impact of variation and enabling process speed. This is done in the context of designing and developing processes, products, and systems.

In the following sections we dive into the details of Six Sigma and Lean, in order to extract the concepts and procedures that are depicted in the static model of Lean Six Sigma (Fig. 3) and the dynamic process model of Lean Six Sigma Methodology (Fig. 4). These models will be used in Sect. 4 for integration of quality procedures of Lean Six Sigma Methodology and Scrum.

3.1 Six Sigma

Six Sigma is a management approach which creates an environment where the methodology is applied as a business strategy. Therefore, Six Sigma is a disciplined process that focuses on delivering products with a consistent quality. The strength of Six Sigma is the continuous improvement process underpinned by three Pillars: *change empowerment*, *training of resources* and *top management support*.

The target of Six Sigma are operational business processes. A *Process* consists of four parts, *Input*, *Activities*, *Output*, and *Feedback*. The *Output* (y) is the final product delivered to an internal or external customer, while the *Input* (x) is any resource put into a process to be transformed as result. Any change in the inputs causes changes in the output, i.e., $y = f(x)$. Each Input can be classified as of the *Type*: Controllable (C), Non-Controllable (NC), Noise (N), and Critical (X). An important aspect of the process is the *Feedback*. Feedback helps in process control, because it suggests changes to the inputs. Each process should be designed to attain an output according to the requirements of the customer.

Six Sigma is implemented by a cycle of activities known as *DMAIC* (Define, Measure, Analyze, Improve, and Control). In the activities of *Define*, the problem statement and the plan for the improvement initiative are defined. In Six Sigma, in addition to defining the problem, the Six Sigma project team is also gathered in this phase. The activities of *Measure* collect the data from the process and determines the current quality and operational performance levels. Also, the measurement criteria (e.g. how to measure, when to measure, and who will measure) is established. In the activities of *Analyze*, the business process and the data generated from the measurement

activity are studied to understand the root causes of the problem. In the activities of *Improvement*, possible improvement actions are identified and prioritized. These are then tested and the enhanced action plan is finalized. In the last activities, related to *Control*, the Six Sigma team goes for a full-scale implementation of the improvement action plan and sets up controls to monitor the system in order to sustain the gains. In the *DMAIC* cycle of problem-solving focused teams are assigned to a well-defined project that directly influences the organization's bottom line with customer satisfaction and increased quality being by-products.

There are several levels in the Six Sigma team structure. The level of *Champions*, with mandate from top management for the improvement initiatives. They identify the scope of projects, design the deployment strategy, support cultural change, and coach *MasterBlackBelts*. At next level *MasterBlackBelts* train and coach *BlackBelts*, *GreeBelts*, as well as *FunctionalLeaders* of the organization. Another level, in the Six Sigma structure, is addressed by *BlackBelts*. They apply strategies to specific projects, and lead teams to execute projects. Finally, *GreenBelts* support the *BlackBelts* by participating in project teams performing day-to-day jobs related to their work area, as well as training junior *YellowBelts*.

Six Sigma defines some core *Concepts* and *Measures*, such as Opportunity, Defect, Specification limits, Rolled Throughput Yield (RTY), and Defects per Million Opportunity (DPMO). An Opportunity is defined as every chance for a process to deliver an output that is either *right* or *wrong*, as perceived by the customers. A Defect is defined as every result of an opportunity that does not meet customer specifications and does not fall within the Specification limits, delimited by an Upper Specification Limit (USL) and Lower Specification Limit (LSL). RTY is a measure of process efficiency expressed as percentage. DPMO is also known as Non-Defect per Million Opportunities (NPMO) and is a measure of process performance.

Specific *Tools* are employed in the *Activities* of the *DMAIC* process. The *Define* uses tools such as Supply, Input, Process, Output, Customer (SIPOC) Diagram, Voice of Customer (VOC), Critical to Quality (CTQ) Trees, Quality Function Deployment (QFD), Failure Mode and Effects Analysis (FMEA), Cause and Effect (CE) Matrix, and Project Charter. The *Measure* uses tools such as GAGE R and R Variables, Run Charts or Control Charts, Cp, Cpk, Sigma level (Z Level) and Defects per Million Opportunity (DPMO), and Anderson Darling Test. The tools used in the *Analyze* are Simple Linear Regression (SLR), Pareto Charts, Fishbone Diagram, FMEA, and Multi-Vari Charts or Hypothesis Tests. In the *Improve*, the tools that can be used are Brainstorming, Piloting and FMEA, and Design of Experiments (DOE). The *Control* activity uses tools such as Control Charts, Control Plan, and Measurement System Analysis (MSA) Re-analysis. Some of these tools can be used interchangeably between the activities.

The benefits of Six Sigma are the fact that the *DMAIC* process eliminates the root cause of problems and defects in processes. The solution can go through the creation of robust products that mitigate the impact of a variable input or output on a customer's experience. Using Six Sigma reduces variation in a process and thereby reduces wastes in the same process. It ensures customer satisfaction and provides process

standardization. Rework is substantially reduced because one gets it right the very first time. Furthermore, users' requirements are key for Six Sigma.

3.2 Lean

The Lean approach, instantiated by the Toyota production system, is based on five *Pillars*: (1) *Value*, which means the specification of value from the point of view of the customer; (2) *Stream*, by the identification of all the steps in the stream of value and eliminating waste; (3) *Flow*, by allowing the value to flow without discontinuity; (4) *Pull*, by letting the customer be the one to pull value from the process; (5) *Continuous improving* of the process in search of perfection.

Lean thinking uses a slightly different approach to problem solving than Six Sigma approach. Progress is made through the execution of events, wherein a small group of employees assemble together through an incremental and continuous improvement, a process, through a series of quick and focused sessions (*Kaizen events*). This approach intends that the team leave the mentality of long-cycle-time project and gain a mindset for action. The Kaizen approach for execution can be synthetized in the following cycle: (1) *Map* out the *As-Is* of the process and create a baseline; (2) *Establish* a vision for the *to-be* state; (3) *Identify* the gaps and establish opportunities for improvement; (4) *Implement* changes and remove waste from the system; (5) *Evaluate* results and institutionalize continuous improvement.

The key thinking behind the Lean approach is to produce what is needed for the customer, when it is needed, with the minimum amount of resources. Lean achieves this by addressing eight types of wastes: (1) *Waiting* (90 percent of artifact's life is spent in this state); (2) *Overproduction* (producing more than what is necessary); (3) *Rework* (e.g. editing, approving, revising); (4) *Motion* (any movement that does not change the characteristics of the product); 5. *Processing* (when a transformation exceeds the required); (6) *Inventory* (the need of extra space and additional handling); (7) *Intellect* (means not taking advantage of the thinking power and knowledge base of human resources); (8) *Transportation* (the unnecessary movement of parts, material or work-in-process from one operation to another).

3.3 Lean Six Sigma

Lean advocates argue that Six Sigma does not directly address speed, flow, and waste within processes. Six Sigma supporters, on the other hand, pointed out that Lean approach is not capable of solving variation problems or process optimization issues. The logical answer to the dilemma is Lean Six Sigma – the combination of the approaches to organizational performance.

Lean Six Sigma is the improvement methodology that combines the best of the Lean concepts and Six Sigma tools, merged in the model of Fig. 3. The emphasis is on obtaining the best of both improvement methodologies, while minimizing any potential weaknesses. The emphasis is on taking advantage of the value generation focus offered by the Lean method, while maintaining the statistical rigors of the Six Sigma methodology.

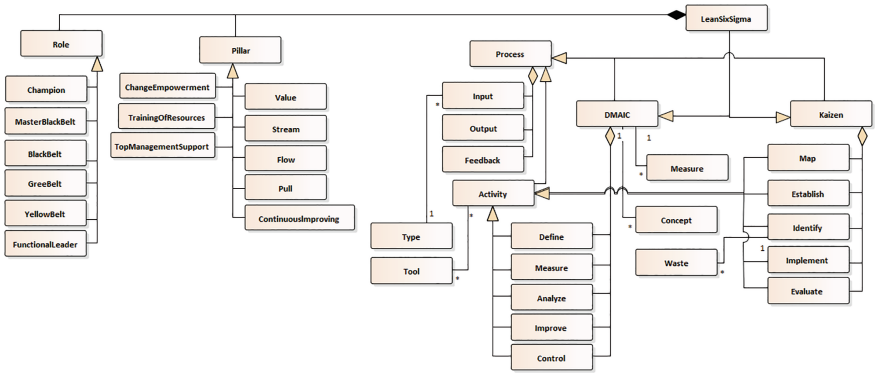


Fig. 3. The model of Lean Six Sigma.

The idea behind adoption of the Lean Six Sigma approach is to enable the organization with a common language, framework, methodology, and process to operating processes, developing and producing products easily and efficiently. The integrated approach of Lean Six Sigma, as depicted in Fig. 4, optimizes the value-creation process and maximizes shareholder value by achieving the fastest rate of improvement in cost, quality, process speed, safety, and customer satisfaction.

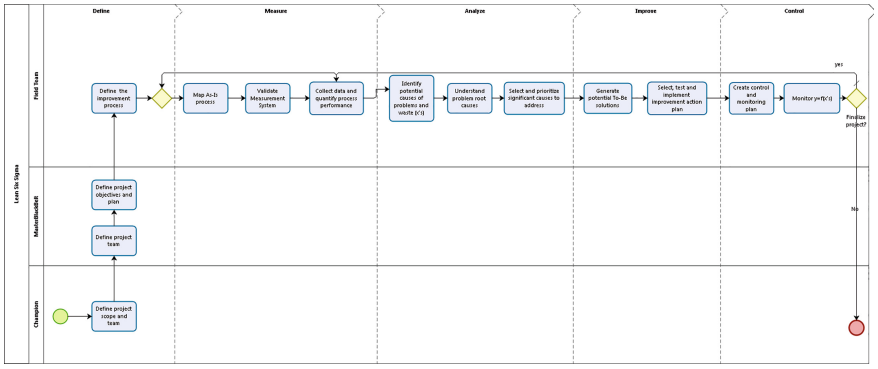


Fig. 4. Process of Lean Six Sigma Methodology.

4 Integration of LSS in Scrum

At the problem-solving level, Lean Six Sigma is a project-driven methodology. The projects executed under the methodology follow the DMAIC process. The aim of Lean Six Sigma is to identify the areas of variation, isolate root causes, optimize processes and thereby reduce or minimize the impact of variation in the products, processes, and services. Variation causes uncertainty, risk, and potential defects.

With the use of the Scrum in software development the aim is to achieve productivity and quality of the process. For assessing productivity in software development,

one must consider the time spent on activities such as coding, testing, bug fixes and deployment of software solution. When assessing the productivity indicator, it is important to consider its relationship with quality. For example, non-compliance, of delivered product, with defined quality standards, result in new tasks for corrections and, consequently, the extension of the development time in order to obtain the intended quality levels.

At the end of a Sprint, the review and retrospective meetings serve as input for the identification of tasks, as well as positive and negative aspects of the Sprint. The results of those meetings are inputs for the use of techniques from LSS, to attain the performance indicators specified by the client. Based on these results, the causes and effects of faults are identified, prioritized and measures with techniques from LSS. The measurements performed are inputs to statistical analyses leading to the definition of more effective action plans. Through the implementation of these plans, the processes are set by enabling continuous improvement and the progressive alignment of performance and quality to the needs and expectations of the customer for the Sprints. The Product Backlog integrates the requirements related with improvements, identified as result of the Sprints, as well as the set of items resulting from the application of the techniques from LSS methodology, during the Sprints.

The integration of the LSS and Scrum intends, through the DMAIC cycle, to contribute for the improvement of productivity and quality indicators of the process undertaken by the Scrum [5]. The set of techniques of LSS methodology suggested to be used in the Sprint, are the following:

Define: (1) inclusion, on the terms of the contract, of the scope and objectives of the use of the LSS, assessment of the associated financial impacts, and definition of the LSS functions of team members alignment with the project; (2) initial assessment of the performance indicator, using descriptive statistics as the baseline of the problem.

Measure: (1) the SIPOC diagram is used to identify the relevant elements of the project for improvement before starting work; (2) each process object of Sprint should be mapped and documented, by including inputs and outputs, the tasks involved, the people accountable, the time spent in the activities, and the granular procedures that are part of these activities; (3) build an cause and effect diagram and analysis for the issues of the process pointed out during the Sprint, Daily Scrum, review and retrospective meetings. In addition to this analysis, map also the relationships between causes and activities, as well as inputs and outputs of the development process; (4) the cause and effect matrix is used to list and prioritize the inputs of the process with the customer's requirements, based on process maps. The purpose of the matrix is to show the process entries that are associated with the characteristics of quality, to assist in the identification of critical inputs; (5) the effort and impact matrix is used for a cost-benefit analysis and prioritization of tasks based on customer's requirements. The matrix is built through the evaluation of the impact on the result of the performance indicator compared to the effort required to perform the improvements; (6) assessment of the initial capacity by determining the value of the corresponding process capability sigma and DPMO; (7) evaluation of the measuring and inspection systems which allows the quantification of reliability of data to obtain.

Analyze: (1) the Fault Tree Analysis diagram is used for the analysis of root causes of specific failures; (2) the analysis of the critical inputs of processes can be made using, for example, regression analysis to model the relationship between two or more variables. The correlation coefficient can explain the degree by which an independent variable explains the variation of dependent variable, and whether the association between the variables is positive or negative.

Improve: (1) build an action plan, for identification of actions, during the Daily Scrum, review and retrospective meetings of the team; (2) update the initial SIPOC considering the implementation of the action plan; (3) update the process map after the implementation of the action plan; (4) the effectiveness of performed actions of improvement is assessed by the final capability of processes.

Control: Build the control plan from the analyses of critical inputs, that really impact on the result of the performance indicator, then transfer the monitoring of the improvements made to the people responsible for the processes.

In order to achieve the proposed integration, the procedures from LSS to be undertaken by Scrum are of the following:

- *Perform the DMAIC cycle of LSS in Sprints*: for each improvement project a Product Backlog is created based on the phases of the DMAIC cycle. The Backlog prioritization can be accomplished based on either users' requirements or analysis of causes of defects/failures of current processes. This analysis enables the prioritization of improvements based on cost-benefit assessment subject to client's needs, undertaken in the current Sprint or future Sprints, in order to achieve productivity and quality gains;
- *Use of process maps in the creation of Sprint Backlogs*: these maps are used in Sprint Backlogs' planning serving as a Work breakdown structure, providing a perspective of the activities that need to be added to the Sprint. The use of these maps allows for the review of processes, the team's training in procedures, assists in the identification of problems, improving internal communication and decreasing the time of Sprints' planning;
- *Identification of causes of defects or faults through the review meetings and retrospectives*: part of the defects or flaws pointed out by team or customer are related to problems in the process. These problems can be analyzed by the team in technical meetings, generating inputs to create cause and effect diagrams. During the analysis of these problems, the use of process maps facilitates the mapping of the causes to the steps and inputs the processes, as well as the identification of critical sources of poor performance of the indicator (e.g. the number of bugs found in the first test cycles);
- *Prioritization of Product Backlogs*: from the analyses carried out with the techniques of the LSS, corrective actions and improvements are prioritized in the Product Backlog by the Product Owner. The prioritization amongst the backlog items for improvements and new developments depend on the decisions of the Product Owner, which need to be aware of the customer needs, the development processes and the techniques of LSS;

- *Use of LSS techniques to improve utilization of the Scrum*: an assessment of the current state of the use of the Scrum is done through a survey to the team and the client. The result of the survey will assist in the identification of strengths and weaknesses. Based on the weak points, improvement projects are defined and initiated.

5 Conclusion

This work intended to integrate the Scrum and Lean Six Sigma methods. By complementing the generic quality procedures of Scrum method with formalized quality measures from LSS in order to improve the efficacy and efficiency of Scrum regarding software development.

The work is developed by describing the fundamentals of the Scrum framework, either in its static perspective, by depicting in a model its main concepts and their relationships, as well as through a dynamic representation of the main activities developed iteratively in Scrum. The same approach was followed to synthetize the fundamentals of the Lean Six Sigma method. The main concepts of this method were also presented through a class model (static perspective) and a process model (dynamic perspective). Those representations of the Scrum framework and Lean Six Sigma method allow the comparison of the approaches in order to supplement the Scrum framework with Lean Six Sigma quality procedures.

The future work intended to perform is the validation, in a real context of a Scrum project, the instantiation of the integrated LSS procedures.

Acknowledgments. This work was supported by Portuguese funds through the Center of Naval Research (CINAV), Portuguese Naval Academy, Portugal.

References

1. Sutherland, J., Schwaber, K.: The scrum guide. The definitive guide to scrum: the rules of the game (2013)
2. Mundra, A., et al.: Practical scrum-scrum team: way to produce successful and quality software. In: Proceedings of 13th International Conference on Computational Science and Its Applications, pp. 119–123. IEEE (2013)
3. Womack, J.P., et al.: Machine That Changed the World. Simon and Schuster, New York (1990)
4. Jugulum, R., Samuel, P.: Design for Lean Six Sigma: A Holistic Approach to Design and Innovation. Wiley, Hoboken (2010)
5. Cunha, T.F.V., et al.: SLeSS: a Scrum and Lean Six Sigma integration approach for the development of software customization for mobile phones. Proceedings of 25th Brazilian Symposium on Software Engineering, pp. 283–292. IEEE (2011)