Lean Measurement: A Proposed Approach

Sylvie TRUDEL[a][0000-0002-4983-1679] and Olga ORMANDJIEVA[b][0000-0001-5641-0976][1]

[a] Université du Québec à Montréal, Montréal, QC, Canada
[b] Concordia University, Montréal, QC, Canada
trudel.s@uqam.ca; ormandj@cse.concordia.ca

Abstract. Lean approach highly promotes value stream between production steps in order to improve software development processes. The main focus of a Lean approach is to identify and eliminate process waste, called "muda", where non-value-added activities must be eliminated to constantly reduce the overall cycle time. Literature proposes solutions for mitigating the waste in Lean manufacturing and Lean software development. However, an approach covering the measurement process for identifying and eliminating measurement-related muda, is missing. In order to address this, we made a parallel between the software development waste types and software measurement activities, within the metric ecosystem. We focused on using the concept of waste as a lens for identifying non-value-producing measurement process elements. In order to achieve this, we constructed a waste identification approach through which we identified eight software measurement wastes, and proposed guidelines for measurement waste identification and reduction. These guidelines can help companies identify measurement process issues that are present in their process in a timely and efficient manner.

Keywords: Lean, Measurement, ISO 15939, Waste.

1 Introduction

The face of the software measurement universe has been through many transformations over the last couple of decades. The metrics being measured and the usage of their results had to evolve over time, in accordance with the evolution of the software engineering and software management practices. For instance, there was a time when software size was measured in Source Lines of Code (SLOC) for means of productivity and estimation. Today, with the evolution of the software measurement field and the advances in the software engineering body knowledge, software organizations are measuring the software with recognized ISO standards. Software development and management methods have also greatly evolved, where organizations are seeking a never-ending efficiency improvement. In order to improve their efficiency, and also their effectiveness of delivering quality software, many organizations implemented

[1] Copyright ©2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Lean approaches to their development and maintenance activities, including Agile methods and frameworks (i.e. Scaled Agile Framework for Enterprise - SAFe), and the DevOps approach.

In this paper we apply Lean principles to software measurement, more specifically, the Lean approach to identifying waste types and eliminating process waste, to the software measurement process in the context of the metrics ecosystem founded on the ISO standards ISO/IEC/IEEE 15939:2017 on measurement process and ISO/IEC 25021:2012 Systems and software Quality Requirements and Evaluation (SQuaRE). The proposed measurement process improvement and guidelines aim at eliminating or reducing waste in the metrics ecosystem.

2 The Lean approach

The Lean approach has its roots within manufacturing when Toyoda, Ohono, and Shingo defined and introduced fundamental principles of manufacturing at Toyota Motors in Japan in the 1930s[2]. This approach is known as the "Toyota Production System (TPS)". From 1988, TPS was renamed as "Lean Production System" [1]. The main focus of a Lean approach is to identify and eliminate process waste, called "muda" in Japanese, where non-value-added activities must be reduced or eliminated [2].

There are five key principles of the Lean approach[2]:

- **Value** is specified from the customer point of view;
- **Value stream** is defined so that wasted steps are identified and eliminated;
- The product goes through a **Continuous Flow** of its remaining value-added steps;
- The process applies a **Pull** approach where products are not pushed to the next step, but the next step pulls the products when capacity is available;
- Teams are seeking **Perfection** where cycle time must be constantly reduced.

Regarding the wasted steps from the 2nd principle, there were originally seven types of "muda" or types of waste[3], having relationships to one another:

1. **Overproduction**: producing more than necessary, often to compensate for defects;
2. **Waiting** (time on hand): resulting from lack of coordination or misunderstanding of the value stream (e.g. waiting for an approval, material, equipment availability, etc.);
3. [Unnecessary] **transport** (or conveyance): often due to unnecessary distance between two or more steps of the process;
4. **Extra-processing**: adding tasks, activities or materials that do not add value for the customer, often from lack of understanding of its needs;
5. Excess **inventory**: stocking more in-process or final products than the demand, often as a result of overproduction;

[2] https://en.wikipedia.org/wiki/Lean_manufacturing, last accessed 2020/07/08.
[3] Adapted from https://en.wikipedia.org/wiki/The_Toyota_Way, last accessed 2020/07/08.

6. **Motion**: useless motion of people or equipment, including searching for documents;
7. **Defects**: wasting effort on fixing defects, managing rejects, or doing rework.

Later, when the Lean approach was combined with the Six Sigma approach [3], an 8th muda was added:

8. **Underutilization of skills** relates to any form of skills or competencies underutilization, such as delegating tasks to undertrained employees, not using talents where they could bring more value and not benefiting from participation of skilled workers as their improvement ideas are simply ignored.

2.1 Applying Lean principles to software development

After the Agile Manifesto was published in 2001[4], it became obvious that Lean principles were also applicable to software development. Agile methods that followed, such as Scrum and more specifically Kanban, have put forward Lean principles built-in these methods [4]:

- The customer defines the value and sets the priorities;
- The process is streamlined in a continuous flow to provide value;
- The work in a given timeframe is planned based on the team capacity and known cycle time;
- The team assesses its process regularly to improve its cycle time, mainly through a practice called "retrospective".

The main purpose of a retrospective session is to improve the software process, which should later be reflected in the reduction of the cycle time or, in other words, in the increase of team's productivity. In conformance with the Lean principles, Mary and Tom Poppendieck explain that in order to improve the cycle time, a software team must first identify and eliminate waste in its process. They make a clear parallel between manufacturing waste types and software development waste types (see Table 1) [4].

Table 1. Parallel between manufacturing and software development waste types (adapted from [4]).

The wastes of Lean manufacturing	The wastes of Lean software development
Overproduction	**Extra features**, such as trying to include more features than the team capacity to end up having no other choice but to work overtime, which may be a source of unnecessary fatigue leading to more defects as developers become tired, thus it lowers the product quality.
Waiting	**Waiting**, resulting in wasting effort. E.g. compiling the software using a slow hard drive when it can be much faster on an SSD drive; waiting for multiple authorizations to acquire or upgrade a needed tool, etc.

[4] Beck, K. et al. (2001). Manifesto for Agile Software Development, URL: http://agilemanifesto.org/, last accessed 2020/06/25.

The wastes of Lean manufacturing	The wastes of Lean software development
Transport	**Task switching**, such as assigning a developer to work concurrently on several projects during the same day.
Extra processing	**Extra processes**, tasks or activities that do not add value from the customer point of view, often caused by inadequate tools, an obsolete process, or misunderstanding of customer needs. E.g. An analysis process requiring that data elements to be defined textually in detail at the end of the use case documents, duplicating information normally found in a centralized data dictionary (related to defects, as copying is likely to introduce inconsistencies).
Inventory	**Partially done work**, such as early completion of analysis on very low priority functionality when the risks of abandoning it are very high. Other examples: starting the development of several functionalities in parallel at the beginning of a sprint, but not completing any of them by the end of the sprint, only to find out later that the user's needs were misunderstood.
Motion	**Motion**, useless motion of people, information or equipment. E.g. Searching over 20 minutes for an analysis or architecture document on the local network that has been misplaced, going back and forth to the supervisor office (at a different location) asking permission for every step to be carried on (related to underutilization of skills).
Defects	**Defects**, such as bugs, unclear requirements, inconsistent data model and inadequate design or architecture.
Underutilization of skills	**Underutilization of skills**, such as assigning software tasks to undertrained developers. Other examples: limited authority and responsibilities, micromanagement, multiple authorizations to upgrade a laptop or a software tool (also relating to waiting), low problem resolution when people are working alone, an employee unable to verify the quality of work he/she produced, etc.

The above listed manufacturing and software development types of waste are likely to have a parallel in software measurement activities and information products. The next section discusses first the metrics ecosystem from the available ISO standards viewpoint. Then the waste types of software measurement are defined and illustrated with practical examples.

3 Metrics Ecosystem

According to many studies on the application of measurements and models in industrial environments, measurement, in order to be effective must be [5]:

1. Focused on specific goals;
2. Applied to all life cycle products, processes, and resources;
3. Interpreted based on characterization and understanding of the organizational context, environment, and goals.

This means that measurement must be defined in top-down fashion, that is, it must be based on goals and models. A bottom-up approach is not likely to work, because the measurements used to evaluate the many observable characteristics in software and how they are interpreted (e.g. duration, cycle time, number of defects, complexity, lines of code, severity of failures, effort, productivity, defect density, etc.) require clarification, which is impossible without the appropriate models and goals. In other words, meaningful measurement entails a well-defined system of elements required to sustain measurement for decision-making.

Below we briefly introduce ISO/IEC 25021 [6], which defines an initial set of Quality Measure Elements (QME) to be used through the product life cycle for the purpose of SQuaRE. The benefits of defining and using the QMEs include [6]:

- To provide guidance for organizations developing and implementing their own QMEs;
- To promote the consistent use of specific QME for measuring and using the product properties that are relevant to different product quality characteristics and sub-characteristics;
- To help identify a set of QMEs that is uniquely required in developing all the quality measures for a given set of product characteristics or a set of sub-characteristics.

Quality Measure (QM) is defined and then QMEs are derived as depicted in **Fig. 1**. The user of the measurement method should identify and collect data related to quantifying the property (**Fig. 1**). Depending on the context of usage and objectives of the QME, several properties and sub-properties can be identified from the artifacts, components, and the behaviour of the target entity. These are the input to the measurement method, as depicted in the figure below. A measurement method must be applied to a property to define and identify a way to quantify a QME (ISO/IEC FDIS 25021: 2012). A measurement function is applied to a QME to generate QM.

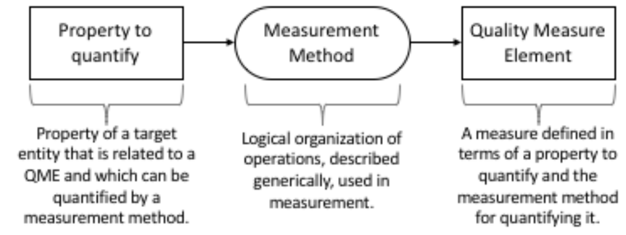
Property to quantify
Measurement Method
Quality Measure Element
Property of a target entity that is related to a QME and which can be quantified by a measurement method.
Logical organization of operations, described generically, used in measurement.
A measure defined in terms of a property to quantify and the measurement method for quantifying it.

Fig. 1. Relationship between property to quantify, measurement method and QME [6].

ISO/IEC 15939: 2017 [7] standard establishes a common measurement process and framework applicable to software engineering and management disciplines. The process is described through a model that defines four measurement activities required to specify what measurement information is required, how the measures and analysis results are to be applied, and how to determine the validity of the analysis results. The measurement information model is a structure linking information needs to the relevant QM and the corresponding QME that provide a basis for decision-making.

Consequently, the Metric's ecosystem structured after ISO/IEC 15939: 2017 [7] should include (see **Fig. 2**):

- Commitment for measurement (step 1 of ISO 15939), where requirements for measurement are defined and communicated, participants are assigned and trained, and other resources are provided for.
- Measurement plan (step 2 of ISO 15939), including identification and prioritization of information needs, selection, specification and definition of measures and tools to satisfy those information needs, and definition of criteria to evaluate information products and the measurement process (all issued from step 2 of ISO 15939).
- Measurement results (out of step 3 of ISO 15939) where the measurement plan is deployed to collect, store, verify and analyze data to provide information items.
- Improvement process of the measurement process (step 4 of ISO 15939).

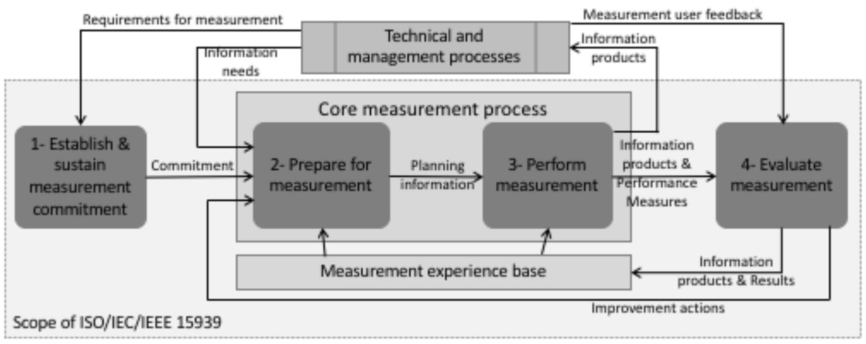
Requirements for measurement
Technical and management processes
Measurement user feedback
Information needs
Information products
Core measurement process
1- Establish & sustain measurement commitment
Commitment
2- Prepare for measurement
Planning information
3- Perform measurement
Information products & Performance Measures
4- Evaluate measurement
Measurement experience base
Information products & Results
Improvement actions
Scope of ISO/IEC/IEEE 15939

Fig. 2. Overview of ISO/IEC/IEEE 15939 [7].

As can be noted from this section, Step 4 of ISO 15939 calls for the improvement of the measurement process. The following section discusses the amendment of the Metric's ecosystem with the Lean principles in order to enhance its processes.

4 Applying Lean principles to software measurement

In this section we elaborate on the mapping of the key Lean principles to the four steps of the measurement processes.

The five key principles of Lean can be applied to software measurement, in support of ISO 15939 for the following reasons:

- To ensure the software measurement process is **value** driven, information needs and related objectives have to be clearly defined as inputs to measurement (step 2 of ISO 15939).
- The measurement **value stream** should be highlighted in the measurement plan (step 2 of ISO 15939), more specifically as a means to define the measurement process graphically; value stream mapping in Lean also allows to understand the cost of measurement.

- The processing of data (collect, store, verify, analyze, report, and record) should follow a **continuous flow**, where automation should play an ever-growing role (step 3 of ISO 15939).
- Every step of the measurement process (step 3 of ISO 15939) should apply a **pull** approach, in order to identify and address bottlenecks and limit the work in progress (e.g. having too much data collected, but not analyzed because verification has not been done).
- Owners of the measurement plan should strive to **perfection** on a regular basis when evaluating measurement (step 4 of ISO 15939), including information items and measurement process.

As mentioned earlier and in conformance with the Lean principles, a software team must first identify waste in the measurement process. The following section explains how the 4th step of ISO 15939 can be used to identify waste in the software measurement process.

5 Identifying waste in the measurement process

Inspired by the work done by the Poppendieck [4], we constructed a waste identification approach by outlining a parallel between manufacturing and software measurement waste types, through which we identified eight software measurement wastes.

The generic concept of waste in the manufacturing processes is used to identify non-value-producing measurement process elements based on the general definition of waste. Table 2 provides a parallel between the eight types of waste in Lean and what would be their equivalence in software measurement.

The value of these software measurement waste types is that they can be used as an input to regular retrospectives on the measurement program (step 4 of ISO 15939). It is expected that organizations derive a checklist for that matter, including those waste types that they might have encountered previously. This checklist would then be part of their measurement experience base.

Table 2. Parallel between manufacturing and software measurement waste types.

The 8 wastes of Lean manufacturing	The 8 wastes of Lean software measurement
Overproduction	**Overproduction**: producing more measurement data and indicators than required by information needs, stated or implied. E.g., collecting measurement data that are never used for analysis or decision-making. For example, collecting software complexity data without analyzing the outliers.
Waiting	**Waiting:** unproductive time or effort derived from unavailable information, people or tools. E.g., spending time reminding people of their missing timesheets several days after they are due, which may lead to imprecise effort data (related to Defects).

The 8 wastes of Lean manufacturing	The 8 wastes of Lean software measurement
Transport	**One-size-fits-all:** Applying an existing measurement plan when the new context requires changes or adaptations. E.g., insisting on having classic monthly project status reports for a new Agile project with 3-week sprints.
Extra Processing	**Extra processing**: non-value-added processing of data and indicators. E.g., processing data analysis not linked with any goal or information needs; setting up an expensive statistical tool only to use basic functions available for free in Excel.
Inventory	**Partially done work**: any incomplete portion of the measurement process. E.g., having accumulated many metrics, analyzed but not shown or used by those who could make decisions based on them.
Motion	**Motion:** useless motion of people, information or equipment. E.g. Moving data manually across applications or tools.
Defects	**Defects**: wrong or inadequate measure or its related measurement results. E.g. inadequate measure for the type of decision that needs to be done with it; data of inappropriate unit of measure; wrong or inadequate quality of data; etc.
Underutilization of Skills	**Under-utilization of skills**: not benefiting adequately or bad utilization of skills and competencies of the workforce. E.g. Tasking a developer to measure the functional size without training or support; not defining potential actions based on measurement results that would enable the workers to take action instead of having to wait for the manager.

6 Conclusion and Future Work

This paper focused on proposing definitions of waste types, corresponding decision-making and resultant waste-elimination actions for the measurement process in support of step 4 of ISO 15939. Waste could be found in the measurement activities, inadequate usage of measurement tools or skills, unreliable data collection, or inappropriate measurement analysis and/or indicators that are impossible to trace to the information needs of the organization. The paper proposed guidelines for step-4 of ISO15939 aiming at identifying waste elements as part of the measurement retrospective analysis activity in step-4. These guidelines can help organizations identify and mitigate the issues in their measurement processes in a timely and efficient manner. The choice of ISO 15939 is justified by the fact that the standard is already widely accepted by the industry, where it is being adapted to their specific information needs of its users.

Our future work will explore the applicability of other key principles from Lean software development, such as elimination of waste, amplification of learning best measurement practices, and optimization of the whole measurement process. Future research directions will tackle case studies and ISO standardization of the waste identification method in Lean software measurement processes.

References

1. Krafcik, J.F.: Triumph of the Lean Production System, MIT Sloan Management Review, Volume 30, Number 1, Fall 1988, Cambridge (MA), USA. (1988).
2. Rousseau, C.: Le Lean Manufacturing (in French), URL: http://leleanmanufacturing.com/, last accessed 2020/04/25.
3. Geoffrey, M.: Eliminate all muda, Manufacturing Engineering, Vol. 126, Nb. 4, (2001).
4. Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit, Addison-Wesley Professional, May 2003, ISBN-13: 978-0321150783, (2003).
5. Fenton, N., Bieman, J.: Software metrics: a rigorous and practical approach. CRC Press, (2014).
6. ISO/IEC 25021:2012: Systems and software engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Quality measure elements, International Organization for Standardization (ISO), Geneva, Switzerland (2012).
7. ISO/IEC/IEEE 15939:2017 Systems and Software Engineering — Measurement Process, ISO, Geneva, Switzerland (2017).