Master Thesis Software Engineering Thesis no: 1MSE:2013-01 July 2013



Lean software development measures - A systematic mapping

Markus Feyh

School of Engineering Blekinge Institute of Technology SE-371 79 Karlskrona Sweden This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 10 weeks of full time studies.

Contact Information: Author: Markus Feyh E-mail: <u>marfeyh@gmail.com</u>

University advisor: Dr. Kai Petersen School of Engineering

School of Engineering Blekinge Institute of Technology SE-371 79 Karlskrona Sweden

Internet	: www.bth.se/com
Phone	: +46 455 38 50 00
Fax	: +46 455 38 50 57

ABSTRACT

Context. Process improvement using lean software development uses measures to answer information needs. Measures are important in process improvement as they identify whether improvements have been made or further improvements are needed.

Objectives. This study aims to identify the measures proposed in literature for lean software development and structure them according to ISO/IEC 15939.

Methods. The research methodology consists of systematic mapping and uses thematic analysis. **Results**. Lean software development literature has become more frequently published from 1996 to 2013. The most common research types were evaluation research and experience reports. Themes were identified in measures resulting in the identification of 22 base measures, 13 derived measures and 14 indicators in total. Identified measures were then structured using a proposed meta-model adapted from ISO/IEC 15939. Using the proposed meta-model twelve models of measures were instantiated.

Conclusions. Gaps exist in the lean principles for deferring commitment and respecting people. Structuring measures in models presents opportunities to identify shared dependencies in measures. Lean software development research guidelines were defined. Further research into the comparison of indicators, the industrial use of measures and the representation of models of measures is needed.

Keywords: lean software development, measures, systematic mapping, metrics, indicators, ISO/IEC 15939, measurement information model

Lean software development measures a systematic mapping

Markus Feyh School of Computing Blekinge Institute of Technology Karlskrona, Sweden Email: marfeyh@gmail.com

Abstract—Background: Process improvement using lean software development uses measures to answer information needs. Measures are important in process improvement as they identify whether improvements have been made or further improvements are needed.

Objective: This study aims to identify the measures proposed in literature for lean software development and structure them according to ISO/IEC 15939.

Method: The research methodology consists of systematic mapping and uses thematic analysis.

Result: Lean software development literature has become more frequently published from 1996 to 2013. The most common research types were evaluation research and experience reports. Themes were identified in measures resulting in the identification of 22 base measures, 13 derived measures and 14 indicators in total. Identified measures were then structured using a proposed meta-model adapted from ISO/IEC 15939. Using the proposed meta-model twelve models of measures were instantiated.

Conclusion: Gaps exist in the lean principles for deferring commitment and respecting people. Structuring measures in models presents opportunities to identify shared dependencies in measures. Lean software development research guidelines were defined. Further research into the comparison of indicators, the industrial use of measures and the representation of models of measures is needed.

I. INTRODUCTION

Lean software development is based on the application of Toyota's product development system [1] to software development [S1]. It is considered to be agile and focused on creating customer value [2], yet at the same time optimizing the end to end flow of work [3]. The field of lean software development has only recently emerged as a software process [S2], [S3]. Furthermore, many of the processes improvements have resulted from using lean measures [S4], [S5]. Many of the measures found in lean software development have been adapted from lean manufacturing. As a result, the basis for lean software development can be traced back through lean product development to lean manufacturing [S2]. Lean manufacturing contrasts with lean software development in that inventory, work-in-progress and defects are not physically visible [S2].

In the field of lean software development Poppendieck and Poppendieck [S6] recommend to "measure up" by focusing on measures that optimize the whole process instead of suboptimizing. This is echoed in research when the flow of work, i.e. inventory, through the value stream has been investigated [S7]. Given leans emphasis on the end-to-end flow, an understanding of what measures are actually researched, presented or used in literature within the field is important.

There is no structured mapping of lean software development measures. Therefore, the need for a greater understanding of measures in lean software development literature has led to this thesis. In order to gain insight in lean measures, a systematic mapping that explored measures of lean software development was conducted by the author. A systematic mapping structures an area of research, with the aim of discovering research gaps. As a means for structuring measures in lean software development, ISO/IEC 15939 [4] has been used. The specific contributions of this thesis are:

- identification of base measures, derived measures and indicators in lean software development (see Section V),
- modelling of the relations of measures (see Figure 5),
- analysis of publication trends over time and research types (see Section IV),
- A meta-model that is used to classify and structure measures is adapted from ISO/IEC 15939 (see Figure 1),
- A full meta-model of the ISO/IEC 15939 measurement information model in Appendix A,
- models of measures are instantiated from the simplified meta-model of measures in Figures 8, 9 and Figures 12 through 20 in Appendix D,
- mapping of measures to lean principles (see Section VII-D),
- guidelines for lean software development research in Section VII-F
- a template for creating models of measures (see Figure 6) and its accompanying source code in Appendix C.

The paper is organized as follows. Section II discusses the related work. Next, Section III describes the research methodology applied. Section IV presents the publication results and Section V identifies as well as lists lean software development measures. The results are then synthesized in Section VI. Finally, the findings of the paper are analysed in Section VII and concluded in Section VIII.

II. RELATED WORK

The term "*lean*" originates from manufacturing where it was interpreted by Liker in four concepts [5]. The concepts were: philosophy, process, partnerships and problem solving.

Later, fourteen principles were distilled from the four concepts by Womack *et al.* [6] in the context of manufacturing. The principles defined by Womack *et al.* can be grouped into four sections which are similar to the four concepts of Liker. They are as follows:

- Long-term philosophy: Focusing on long-term value;
- *Right process will produce right results:* removal of waste and focus on value;
- Value to organization by developing people: human development, supporting the corporate identity and working as a team as well as with suppliers;
- Solving root problems drives organizational learning: continuous improvement based on consensus based decision making where multiple options are considered.

Both Liker [5] and Womack *et al.* [6] used the Japanese company Toyota as the basis for their work. In the field of software engineering, Middleton [S3] was one of the first researchers to publish on the application of lean manufacturing principles. Middleton found that resource allocation, flow and lead time improved through the use of lean. The most well-known interpretation of lean in software development, as provided by Poppendieck and Poppendieck [S2], which is also based on lean manufacturing literature. They proposed lean as a way of shortening feedback loops in order to prevent delays and defects. Based on their writing on lean software development, Poppendieck and Poppendieck defined seven principles which are specifically focused on lean software development [S6]. These can be understood as:

- *Eliminate waste:* is about not building the wrong product or the product wrong and doing work in a continuous flow.
- *Build in quality:* means proofing the process for mistakes using continuous integration and zero tolerance of defects.
- *Create knowledge*: use of quick feedback loops and knowledge gained from failures that increases learning in the organization.
- *Defer commitment:* means making decisions that will not commit the organization until the last responsible moment.
- *Deliver fast:* uses the queuing theory to manage the workflow, which means that speed, quality and low cost should all improve at the same time.
- *Respect people:* by providing them with purpose, challenges and responsibility.
- *Optimize the whole:* through purposefully thinking in sustainable ways that focuses on long term solutions and focusing on the whole in order to avoid sub-optimization.

An approach to lean software development in practice named Kanban was taken up by Anderson [S8]. Kanban consists of a kanban pull system, visualization of work and seeks to create a sustainable flow of work. The kanban pull system represents work items as cards that are added into the work process at an agreed capacity. This limits the work in progress (identified as inventory in this thesis), because when capacity is reached work must be completed before more work can be introduced.

Lean was included in a systematic literature review of agile development [2] as a type of agile methodology. While similar in most principles to agile software development, lean software development is different according to [3] because of its focus on the principle "*see the whole*" and a low-dependency software architecture. The differences of lean are important because by focusing on the whole the sub-optimization of local processes can be avoided.

A. ISO/IEC 15939 Measurement Information Model

Using the ISO/IEC 15939 measurement information model is important as it has been shown that using the model can lead to a 50% faster process for measurement program planning and a 35% smaller measurement system [7].

This thesis uses the following definitions which come from the ISO/IEC 15939 standard:

- A *measure* is the variable value which the result of the measurement is assigned to.
- An *information need* is the insight needed for the management of goals, risk, problems and objectives.
- An *indicator* is the measure typically presented to the user and is the result of an algorithm that combines one or more base and/or derived measures.
- A *derived measure* is the result of the combination of two or more base measures.
- A *base measure* is the result of quantifying a single attribute.
- Attributes are the properties of entities.
- *Entities* consist of either a product, process, project or resource.

Some additional elements of the ISO/IEC 15939 measurement information model that can be found in Appendix A have been excluded, because the literature in lean software development does not provide enough information to satisfy all of the elements. Consequently, only the relations of the elements defined above are represented in the simplified meta-model shown by the author in Figure 1. As a result, the simplified meta-model consists of the relationships between information needs, indicators, derived measures, base measures, attributes and entities.

III. RESEARCH METHODOLOGY

The primary research methodology used in this thesis was systematic mapping [8]. The sources for the systematic mapping were then analysed using thematic analysis [9].

A. Systematic mapping

The research methodology describes the steps taken to find relevant literature to conduct the systematic mapping as well as the methodology for structuring measures. A systematic mapping is "*a defined method to build a classification scheme and structure a software engineering field of interest*" [8]. In contrast to systematic reviews, systematic mapping studies are broader in scope and structure evidence according to a



Fig. 1. A simplified meta-model adapted from the ISO/IEC 15939 measurement information model.

classification scheme. This study aims to structure the area of lean software development measures in order to identify gaps and areas of future research.

Overall, the search process resulted in a total of 27 literature sources. Table I provides an overview of the number of literature sources identified and selected in each step.

TABLE I STUDIES IDENTIFIED AND SELECTED

Selection step	Literature selected
Identified studies from database search	472
Selection by title and abstract (database)	21
Selection after full-text reading (database)	16
Identified literature after backwards snowballing	404
Selection by title (snowballing)	73
Selection by abstract (snowballing)	17
Selection after full-text reading (snowballing)	11
Final set of literature sources	27

1) Aim and research questions: In this study, the overall aim was to structure measures from the lean software development literature. In order to meet this aim, the following research questions were asked:

- RQ1: Which trends can be seen in terms of the number of publications and publication forms over time?
- RQ1.1: How were measures evaluated in terms of research types (see [10])?
- RQ2: Which measures have been described for lean software development?

2) Search strategy: The result of conducting the search was the identification of relevant papers. The search was made based on a search query which was used on databases of scientific literature. The author chose the following four electronic databases: Scopus¹, ACM Digital Library², Elsevier/ICM³ and SpringerLink⁴. After conducting the search for primary sources, the author expanded the search in order to include sources found through backwards snowball sampling.



Fig. 2. Database search strategy

Search query: In order to identify relevant studies and make the process replicable, details of the search strategy are documented in Figure 2. In the beginning, relevant keywords were identified. The keywords used were synonyms for terms used for lean software development, measures and the research questions. Next, a search query was built based on the identified keywords or their various forms, e.g. plurals. The search query was piloted⁵ to become more inclusive and identify more key papers. As a result, the final search query selected was:

TITLE-ABS-KEY((lean OR kanban) AND software AND (process OR project OR management OR development OR engineering) AND (measur* OR metric*))

To reduce the amount of noise (i.e. irrelevant papers), the search query was applied only to titles and abstracts. When duplicates were found they were removed. An overview of the search of electronic databases can be seen in Figure 2.

Backwards snowball sampling: In a second phase of searching, the author examined the references of relevant literature by using backwards snowball sampling. When searching for literature, Kitchenham and Charters [11] recommend using backwards snowball sampling in order to find additional sources not discovered using the database query. Moreover, it can be effective to identify relevant sources with less noise [12]. The snowball sampling process consisted of the collection of all references from the relevant literature found using the database search query.

Selection of literature: By screening papers it allowed for the identification of relevant literature. The screening used inclusion and exclusion criteria to assess the literature based on the relevance to the posed research questions. The inclusion and exclusion criteria were applied to each source through examination of it in greater and greater detail.

For the relevant papers identified using the database query, the inclusion and exclusion criteria were used to eliminate papers based on the title and abstract followed by full-text reading. For an overview of the process see Figure 2.

For the sources found using snowball sampling, the author first examined the title; then the abstract or table of contents;

¹http://www.scopus.com

²http://dl.acm.org/

³http://www.sciencedirect.com

⁴http://link.springer.com/

⁵Results of the piloting can be found online: www.student.bth.se/~mafj12/ lean_sd_measures_pilot.pdf



Fig. 3. Snowballing strategy

and finally the full text of the paper for relevance. An overview of the process can be seen in Figure 3. In both selection processes, the following inclusion and exclusion criteria were applied:

Inclusion: The literature explicitly describes the software development process used as being lean or Kanban. Moreover, the literature must describe measures.

Exclusion: Literature that falls outside of the software engineering domain. Literature that describes measures which are only used to compare methodologies. Also, literature that mentions lean but does not make it its focus. Finally, literature that focuses on describing measures specifically for lean manufacturing will also not be taken into consideration.

B. Data extraction and analysis

A common analysis technique in systematic reviews is identified by Petersen *et al.* [8] as thematic analysis. Thematic analysis [13] is a common approach to analyse qualitative research. It focuses on identifying patterns, i.e. themes, from the collected data that are relevant to the research question posed. In the thesis, the research questions are focused on measures and publication details in lean software development.

Dixon-Woods *et al.* [9] describes thematic analysis as sharing commonalities with comparative analysis and narrative studies. The author chose to specifically focus on thematic analysis because it was important to be able to summarize the findings based on a number of perspectives of the systematic mapping, i.e. themes. The approach of narrative synthesis was not chosen as it is has the liability of being dependent on the prejudices of the researcher as identified in [14]. Also, content analysis was not used it would have diminished the complexity and context of the thesis echoing [14]. By limiting the researcher's prejudices and at the same time being able to apply the complexity of analysis available using themes the methodology of thematic analysis was chosen.

The methodology of thematic analysis used in the thesis consists of six steps, these are:

- 1) familiarization with the data,
- 2) generating initial codes,
- 3) searching for themes,
- 4) reviewing themes,
- 5) defining and naming themes,
- 6) producing the report.

Familiarization with the data: In this step, the author became familiarized with the data, which was the area of measures and lean software development. Concretely, the author read the ISO/IEC 15939 standard and created a meta-model based on the measurement information model in the standard [4]. Elements from the meta-model (see Figure 1) were used as an initial basis for understanding the literature.

Next, the author went through the relevant literature by identifying information relevant to measures. Identified information consisted of descriptions on the use of measures, measures themselves and how they were used. The process was carried out as inclusively as possible since it was used as the basis for creating initial codes in the next step.

Generating initial codes: The author inductively generated codes based on information which was meaningful in answering the research questions. The codes used in the analysis were based on the structured understanding using the simplified measurement information meta-model from Figure 1.

The resulting codes can be found in Appendix B which were used to structure the collected data. The coding consisted of three sections: publication details, research paper methodology and measure details. Publication details provided information about the (1) title, (2) author, (3) year of publication and (4) which conference, book or journal it came from. Next, the methodology used in the scientific papers was examined. This consisted of whether the article was empirical and what type of research was conducted. By separating the codes into section it eased the collection.

The goal of codifying measures was to place the identified measures in the context of the simplified meta-model (see Figure 1). The coding was done using the following seven steps.

- 1) Identify the name of the measure and classify it as either a base measure, derived measure or indicator.
- 2) Extract a short description of the measure.
- 3) If possible, identify the most relevant information need for the measure.
- 4) Extract the lower level measures, e.g. derived measures and/or base measures.
- 5) For each base measure identify its attribute.
- 6) Identify the relevant entities based on the attributes from the previous step.
- 7) Repeat the steps 1-6 for each measure identified in the literature.

By codifying the measures using the previous seven steps it allowed for the collection of data on measures.

Searching for themes: In the next step, the author looked for repetitive codes that matched. Based on the analysis, repeating and closely related codes were grouped together in order for themes to be able to emerge.

Reviewing themes: Based on the previous step themes emerged that needed to be reviewed as a result of the combination of closely related codes. As a result, relations between the measures began emerging in the context of the meta-model from Figure 1. *Defining and naming themes*: Measures resulting from the previous steps were then defined and named based on the reviewed codes. In some cases, measures which had the same meaning but different names were reconciled. Additionally, other themes arose that from the coding were defined and named.

Producing the report: In order to produce a report that contributed to the field of lean software development, the relation of measures and the seven principles identified by Poppendieck and Poppendieck [S6] was identified. Moreover, measures were mapped to the elements of the meta-model (see Figure 1) and then synthesized as well as analysed. Themes that arose from the coding of the literature were also included in the report.

C. Threats to validity:

Validity threats are important to discuss in order to judge the reliability of the results. The main threats relevant for this study are related to the search and the researcher's bias.

Missing studies in the search: One threat to validity is that important literature relevant for the research questions could have been missed. The database search was complemented by backwards snowball sampling in order to reduce this threat. Furthermore, the author piloted the search query in order to be more inclusive, which reduced the threat that studies could be missed in the search.

Researcher bias: Researcher bias is a threat during the selection of literature and the extraction and interpretation of results. The study selection was done by an individual researcher, which poses a threat. By being as inclusive as possible and not excluding any literature when in doubt at an early stage (reading abstract and title) the threat was reduced.

Reliability of the interpretation: The reliability of the interpretation is important as differences in the researcher could lead to widely different results for measures. In order to reduce the threat, the author used the measurement information model from the ISO/IEC 15939 standard [4] to structure measures using a simplified meta-model (see Figure 1). By standardizing the interpretation, it reduced the threat and made the results more reliable.

IV. PUBLICATIONS RESULTS

The publication results present the results for RQ1 and RQ1.1.

A. RQ1: Publication trends and forms

To answer the first research question, the number of publications over time in different forms (journals, conferences, books) were analysed. Figure 4 shows the number of publications for each year in three forms. Only a few scientific studies with a focus on measures in lean software development have been published before 2010. Only recently (2010 and onwards) has the number of publications increased significantly. After 2006, the number of journal publications increased visibly. This indicates an increased interest in measures related to lean software development.



Fig. 4. Number of publications versus year of publication.

With respect to publication forms, the author found that 15 studies have been published in conferences and 7 in journals. Furthermore, the author identified 5 books reporting on measures in lean software development. Two conferences attracted two papers each, namely the *Digital Avionic Systems Conference* and the *Lean Enterprise Software and Systems Conference (LESS)*. All other journals and conferences, in which the included studies have been published in, occurred only once. Overall, besides the two previously mentioned conferences, no preference for a particular conference or journal was identified for the studied area.

B. RQ1.1: Research types

Wieringa *et al.* [10] distinguishes different research types, these include:

- validation research: techniques are evaluated in lab experiments,
- evaluation research: techniques are implemented and evaluated in industry practice,
- *solution proposals:* a solution is proposed without evaluation,
- philosophical papers: taxonomies and conceptual frameworks,
- opinion papers: expressing personal opinion,
- experience papers: experts report on their practical experience.

Using the listed types, the author coded the primary studies from the systematic mapping.

Table II shows the research types, number of studies, and references to the primary studies. It is visible that the majority originates from expert experience. Also, a large portion of studies have been conducted using research methods in industry. Only a few studies have been conducted in the laboratory, are pure solution proposals or present opinions. With respect to the evaluation research, it is noteworthy that a large portion of studies – six out of nine – have been conducted in Scandinavia. Finally, none of the studies were philosophical.

Study type	Num. of studies	References
Validation	1	[S9]
Evaluation	9	[S3]–[S5], [S7], [S10]–[S14]
Solution	2	[S15], [S16]
Philosophical	-	_
Opinion	4	[S8], [S17]–[S19]
Experience	11	[S1], [S2], [S6], [S8], [S20]–[S26]

TABLE II Research Types

V. DESCRIBED MEASURES RESULTS

In order to answer RQ2, "Which measures have been described for lean software development?" the measures were structured according to the meta-model (see Figure 1). As a result, the three types of measures (base measures, derived measures and indicators) are presented in this section.

A. Base measures

The base measures were structured according to the entities which they measure.

Measures related to the process and project entities

In relation to the "*Process/Project*" entity (see Table III) there were 12 base measures (BM1 to BM22) identified. Overall, it is noteworthy that the majority of studies are related to measuring time, i.e. the general duration to complete work (BM1), as well as the value added time (BM4) and non-value added time (BM5). In addition, the number of work items (BM6) is a commonly used measure in the literature. Other attributes and their related measures, in comparison, have only a few references.

Duration in time units (BM1) [S2], [S6]–[S8], [S10], [S13]–[S16], [S23], [S25], [S27] is a specified period of time, e.g. an hour, week or day. The measure of the duration of time was identified in twelve sources. Specific applications of duration in lean literature include measuring the working days available [S13], the amount of time it takes to receive feedback from integration and builds [S8] as well as the time spent editing.

Failure load (BM2): Failure load is the general term used by Anderson [S8] to identify the number of work items that must be processed again due to previous low quality. It was identified in four sources. In another context, this is known as maintenance requests, which are the number of requests to fix something in the software product, e.g. the number of new maintenance requests can be measured [S11]. Similarly, trouble reports [S25] are measured and represent issues found in the software product. Both trouble reports and maintenance requests are grouped together since they represent requests to fix something in the product. In a case study, failure load has also been applied when measuring errors requiring rework [S3] when programmers found errors in the work of analysts that needed to be immediately fixed.

Value transition (BM3) [S7], [S11], [S25] measures the point in time for which there is a handoff between activities. The measure was described in three sources. It is especially important, because many sources take the description of the

value transition measure for granted, and move on to more complex measures such as derived measures.

Value added time (BM4) [S2], [S4], [S6], [S8], [S11], [S20], [S25] is the amount of time spent doing activities that add value. Measures of duration which focus on value added activities are e.g. touch time [S8], in process time and core process time [S20]. Value added time can be identified in seven sources. The first two measures touch time and in process time can be considered to be the same because they are the amount of a work item has been assigned without becoming blocked or stuck in a queue. Next, the core process time is also considered to be a synonym of value added time [S20]. It is the amount of time spent working on a value adding work item, excluding the non-value adding activities such as set up or gathering information [S20].

Non-value added time (BM5): Following the value added measures, there is a greater number focused on non-value added time measures [S2], [S4], [S6], [S20], [S25]. Nonvalue added time is also presented as waste time [S2], [S6]. Additionally, these include: request age [S6], transmission time [S20], delay time [S20], set up time [S20] and queue time [S22]. Firstly, the age of requests [S6], which is a measure of how old requests are at a specific point in time. Secondly, the transmission times [S20] which relate to the transfer of information within the information flow. Thirdly, delay time [S20] is defined as the average duration of a single work item in a queue. A more specific example of delay time is queue time, which measures average duration in the backlog [S22]. Through the lean literature examined the measure appears five times. Although there are more measures of duration focusing on non-value added time, both kinds are important for identifying the duration of value and waste (i.e. non-value added time).

Number of work items (BM6) [S2], [S5]–[S8], [S13], [S24]– [S26] is a common lean software development measure found in nine sources and measures the amount of work items, e.g. the amount of requirements [S5], [S7]. The work items measured consists of units of work [S13]; ongoing discarded and completed requirements [S7]; faults [S7]; tests [S7]; and requests [S6]. Work items measure specific types of work items such as in the case of [S7], where the differentiation between ongoing, discarded and completed is made.

Anderson [S8] defines work items as a requirement, feature, user story, use case, production defect, maintenance, refactoring, bug, improvement suggestion, bug, improvement suggestion or blocking issue. Work items are broad by definition and not only limited to these examples according to Anderson.

Bounce backs (BM7) [S2] measure only the number of defects which are repeated after being fixed in a non-satisfactory way. Petersen [S11] also uses the bounce back measure by measuring the number of attempts needed to resolve a maintenance request. In total, bounce backs is described twice in the literature. Bounce back differentiates itself from failure load, because it focuses exclusively on measuring defects which appear again.

Financial costs (BM8) [S2], [S6], [S20] are the costs in-

curred. Non-recurring costs [S20] are one-time costs that must be paid for work to be completed but are not continuous. In contrast, recurring costs [S20] are the costs which need to be paid each time work is completed and can be continuous. Financial costs are described in four different sources from the literature.

Cost of investment (BM9) can be understood as the input value in [S7] as it represents the equipment and tools invested in. The cost of investment is only described in [S7].

Financial revenues (BM10) [S2], [S6] are the incomes which are used to pay costs, and can also be understood as the amount of realized gains from an investment after costs are paid. Interestingly, financial revenues are described in two sources which come from the same authors.

Schedule slippage (BM11) [S13], [S26] is defined in lean software development literature as the duration of time that exceeds the original project plan and is described twice in the literature.

Fault slippage (BM12) [S5] can be defined as when work items need to be fixed later then the relevant planned testing activity. It is described once in literature.

Measures related to the product entity

For the "*Product*" entity seven base measures were identified (BM13 to BM19) and shown in Table IV. Interestingly, the most commonly used measures of the product focus on size of the defect backlog (BM13) and the size of work items in story points (BM14).

Number of defects in the backlog (BM13) is examined in the context of lean software development as the size of the defect backlog [S2], [S10], [S11], [S25]. It is described in four sources and simply measures the total number of defects in the backlog at a given time.

Story points (BM14) [S2], [S23], [S24], [S26] is a concept from SCRUM [15] where the size of the work item is measured. It is described in four sources from the literature. An example of using story points is when estimating the size of a work item. In [S23] only work items smaller than a certain story point size are let into the flow. For example, if a work item could be considered to consist of nine story points it would need to be broken since it is too large and it would not be allowed into the value stream.

Lines of code (BM15) [S16] counts the number of language statements in the source code. It is a very tradition measure which can be observed in only one source.

Cohesion and coupling (BM16) [S9], [S16] are object oriented measures from two sources that capture the internal code quality of the product. Coupling measures the degree to which modules are related to each other, while cohesion examines how strongly related the code is.

Number of passed acceptance tests (BM17) [S16] measures the number of acceptance tests that have been executed successfully with a passing outcome and only was identified in a single source.

Code coverage (BM18) [S27] is the amount of language statements in the software product that are covered by testing and was only identified in the literature once.

Code churn (BM19) [S15] is the number of language statements of code that have been changed, removed or added. It has only been described once in the learn literature.

Measures related to the stakeholder entity

For the "*Stakeholder*" entity three base measures (BM20-BM22) were identified, shown in Table V. Two measures are related to satisfaction, while one focuses on the customer and the other on the employee. Both are perceived assessments through rating. The emotional seismograph is mentioned, however, has not been explained in [S26]. Furthermore, the skill level is assessed, considering both required and available skills.

Perceived customer satisfaction (BM20) is a measure of the level of customer satisfaction with the software product [S6] or a feature [S5].

Employee seismograph (BM21) [S26] is not explained other than it was used to measure and monitor the level of employee satisfaction.

Perceived skill level (BM22) [S13] measures the amount of skills needed or the amount of skills available.

Overall, it is clearly visible that the "*Process/Project*" related measures are emphasized in the lean literature. Among them the base measures related to time (BM1, BM4 and BM5) and the number of work items (BM6) seem to be given a lot of emphasis as seen by the amount of sources that describe them.

B. Derived measures

Derived measures use functions to combine two or more base measures [4]. In Table VI the elicited derived measures are listed. Provided within the table are the relevant base measures which have been identified from the literature in the column *Base measures*. Base measures were identified based on their availability in the literature. Overall, Table VI shows 13 derived measures, which used the number of work items (BM6) and duration of time (BM1) most frequently as base measures.

Throughput (DM1) [S6], [S8], [S10], [S14], [S25] can be defined as the number of work items that are completed over a given time period. A common synonym for throughput is velocity [S5], [S23], [S26] and is primarily used in the context of organizations which have a background in SCRUM [15].

Throughput is also used in specific contexts such as for work items [S22], defects [S10], story points [S24] and maintenance requests [S11]. For example, for the defect inflow and outflow [S10] when examining the throughput of defects into and out of phases.

Furthermore, distinctions are made in the inflow and outflow of work items [S10], [S11]. For example, when examining the outflow of work done in a phase that is then handed over to the subsequent phase [S7].

Cycle time (DM2) [S1], [S2], [S6], [S13], [S18], [S20]– [S23] can be defined as the duration of time per work item. In the literature different kinds of work items are measured by cycle time including features [S6] and story points [S23]. Poppendieck and Poppendieck [S6] describe cycle time as

 TABLE III

 BASE MEASURES FOR THE "PRODUCT" AND "PROCESS" ENTITIES

ID Measure	;	Attribute	Sources
BM1: Duration	in time units	Timestamps	[S2], [S6]–[S8], [S10], [S13]–[S16], [S23], [S25], [S27]
BM2: Failure lo	oad	Work items needing rework	[S3], [S8], [S11], [S25]
BM3: Value tra	nsition	Time of value transition	[S7], [S11], [S25]
BM4: Value ad	ded time	Time spent adding value	[S2], [S4], [S6], [S8], [S11], [S20], [S25]
BM5: Non-valu	e added time	Time spent not adding value	[S2], [S4], [S6], [S20], [S25]
BM6: Number	of work items	Work items	[S2], [S5]–[S8], [S13], [S24]–[S26]
BM7: Bounce b	backs	Reworked items needing rework	[S2], [S11]
BM8: Financial	l costs	Costs	[S2], [S6], [S7], [S20]
BM9: Cost of i	nvestment	Investment costs	[S7]
BM10: Financial	revenues	Revenues	[S2], [S6]
BM11: Schedule	slippage	Time delay in completion	[\$13], [\$26]
BM12: Fault slip	opage	Slip through defects	[S5]

 TABLE IV

 Base measures for the "Product" entity

ID Measure	Attribute	Sources
BM13: Number of defects in the backlog	Defect backlog	[S2], [S10], [S11], [S25]
BM14: Story points	Work item size	[S2], [S23], [S24], [S26]
BM15: Lines of code	Language statements	[S16]
BM16: Cohesion and coupling	Relatedness of source code	[S9], [S16]
BM17: Number of passed acceptance tests	Passed acceptance tests	[S16]
BM18: Code coverage	Language statement test coverage	[S27]
BM19: Code churn	Added, removed or edited language	[\$15]
	statements	

 TABLE V

 Base measures for "Stakeholder" entity

ID Measure	Attribute	Sources
BM20: Perceived customer satisfaction	Customer satisfaction	[S5], [S6]
BM21: Employee seismograph	Employee satisfaction	[S26]
BM22: Perceived skill level	Skill level	[S13]

a versatile measure that exposes defects, complexity, handoffs and code churn. Poppendieck and Poppendieck further describe it as one of the best ways to measure quality and performance of the operation.

A measure which is similar to cycle time is the 'common tempo time'. 'Common tempo time' [S13] uses the net working days available divided by the number of work items required. This provides an estimate of the cycle time needed in order to accomplish the required work items given the working days available.

Lead time (DM3) [S4], [S5], [S8], [S11], [S20]–[S23], [S25] can be measured from the time a work item is requested until the time it reaches the terminal value phase. It has been used in the context of trouble reports [S25], maintenance requests [S11] and tasks [S20]. Alternatively, it can be defined as the time from when the request is made until active work on the work item begins [S20].

Capacity (DM4) [S2], [S5], [S6], [S8], [S14], [S20], [S26] is defined as the amount of work items that can be handled in a given time period. This measure is used in the context of the capacity of the organization [S5], [S26], release [S14] and process [S20]. Capacity is something that is difficult to measure, but can be estimated using historical data or expert knowledge [S5]. It is recommended that capacity is not exceeded [S2], [S5], [S6], [S8], in order to create slack in

the process.

Profit and loss (DM5) [S2], [S6], [S8] is a financial measure which presents the revenues and expenses during a period of time. Poppendieck and Poppendieck [S2], [S6] recommend using profit and loss as a measure of product success.

Inventory (DM6) [S5], [S7], [S20] can be measured as the number of work items in a value phase at a specific point in time. The set of work items in a value phase can alternatively be referred to as a queue, hence the synonyms queue size [S22] and queue [S14].

The work items contained in the inventory vary. Examples of individual inventories include change request, faults, requirements and test cases [S5] in addition to maintenance requests [S11].

Time efficiency (DM7) is presented by Poppendieck and Poppendieck [S2], [S6] as the percent of value adding time as a percent of the total time. Alternatively, the time efficiency can be calculated using a number of ratios, e.g. the in process time divided by the cycle time [S20].

Value efficiency (DM8) is a measure that takes the difference of the output and input values divided by a specified time window [S7]. The value efficiency takes into consideration the amount of investment needed in order to achieve value in a limited span of time.

Net promoter score (DM9) [S6] calculates a score after

taking into consideration whether users are promoters, passive or detractors in relation to their satisfaction with the product.

Initial quality (DM10) [S8] measures the amount of defect slippage as a percent of the work-in-progress and throughput. Similarly, failure rate [S20] is the percent of work items that will be defective and need to be reworked.

Return on investment (DM11) [S2], [S6] is a the result of calculating the efficiency of an investment.

Design debt (DM12) [S16] uses basic object oriented metrics to measure whether the design debt is high or low for each individual module. A high design debt can indicate poor design and a need for refactoring.

Rework rate (DM13) [S20] is the amount of work items with defects that are outputs.

The derived measures that are most prominently found in the literature were throughput (DM1), cycle time (DM2), lead time (DM3), capacity (DM4) and inventory (DM6). Of note is that the prominently identified measures share common base measures. These prominent derived measures along with all derived measures identified are described below.

C. Indicators

Indicators provide evaluations of specified attributes with respect to information needs [4]. Indicators are also often visualized graphically and used in organizational dashboards to support decision makers. The author identified 14 indicators in the literature, shown in Table VII. The table also shows the potentially relevant measures that the indicator can be applied to based on its use in literature. The studies where specific indicators have been used are indicated in the column *Source*. Additionally, descriptions of the use of the indicators is provided in the remaining subsections text.

Cumulative flow diagram (11) [S5], [S7], [S8], [S11], [S22], [S24] is a stacked graph that shows the number of work items versus the time. It consists of as many lines as there are activities in the value stream. It is organized so the uppermost line represents the trend of the initial activity and the lowest line represents the trend of the terminal activity.

Regression (12) of lead time [S8], [S11], [S22], [S25] and cycle time [S6], [S13], [S23] shows the measures over a specified period of time. The regression uses historical data to show trends, e.g. whether it is increasing or decreasing.

Moreover, other kinds of regression include the linear regression function [S7] or the R-squared values [S24] of the lines representing the activities in the cumulative flow diagram. For example, when using the linear regression function, if the slope of the subsequent value phase is lower than the previous phase than it can indicate a bottleneck. Using the R-squared values, the activities with the lowest slope could indicate a bottleneck.

Box-plot (13) [S11] can be used to visualize the variability of lead times between priorities and phases by graphing quartiles of e.g. the lead time for different priorities.

Burndown chart (14) [S2], [S22], [S26] shows the trend of the amount of story points remaining versus the duration of time.

Control chart (15) uses statistical control to monitor whether observed values are in line with expectations. This is for example by using standard deviations as control limits based on historical values. One example of using a control chart is with weighted maintenance requests (MR) [S11]. Moreover, it is also applied to individual inventories in [S5] to monitor and control that the inventory does not exceed or drop below expected values.

Utilization (16) [S5], [S6], [S20], [S26] measures how much of the organizations capacity is used. It is accomplished by comparing the load (i.e. inventory) to the capacity of the organization. Utilization can be used to assess the overall state of a software development organization [S5] by understanding if the organization is overloaded.

Design debt advice function (I7) [S16] combines the design debt for all modules of the software to indicate the design debt of the overall product, which can indicate whether corrective action should be taken in the form of refactoring.

Variance (18) indicates inconsistency in the flow of work items through an activity. In [S7] it is calculated using the estimation error from the linear regression function of the activity trend line from the cumulative flow diagram (I1).

Due date performance (19) [S8], [S25] is the percentage of work items which are delivered in the time expected. In the same way, the lead time success rate [S25] has been described as the number of trouble reports that are answered in time in relation to all trouble reports.

Relative comparison (110) is the comparison of the relative size of elements to one another. For example, the inventory of each activity is measured by comparing the relative percent and size of inventories in each activity [S25]. Another example can be found in the comparison of future value stream maps to current value stream maps [S4].

Cost of delay (111) [S8], [S24] is the difference in the profit and loss achieved between two different points in time.

Cost model (112) [S7] consists of the three components investment, work done and waste that are added together in order to measure the resulting cost.

Overall state (113) [S11] indicates the result of combining the individual inventory levels which are either high or low based on whether they are higher or lower than capacity. This indicator takes into account two or more inventories in order to indicate the overall state of lean software development process.

Histogram (114) shows the distribution of the lead time [S8] or potentially any other measure. Anderson [S8] prefers using the histogram as an indicator for the lead time of individual work items, because it shows the outlying measurement results and opportunities for improvement more clearly.

VI. SYNTHESIS

A. Measures in lean software development

Indicators and derived measures are most frequently based on the number of work items (BM6) and the duration in time (BM1) base measures. By structuring measures using the meta-model the dependent measures for indicators and derived measures can be seen by modelling their relatedness. When the

TABLE VI Derived measures

ID M	leasure	Base measures	Source
DM1: Th	hroughput	Uses no. of work items (BM6) and duration in time units (BM1)	[S6], [S8], [S10], [S14], [S25]
DM2: Cy	ycle time	Uses no. of work items (BM6), duration in time units (BM1), value transition (BM3)	[S1], [S2], [S6], [S13], [S18], [S20]–[S23]
DM3: Le	ead time	Duration in time units (BM1), value transition (BM3)	[S4], [S5], [S8], [S11], [S20]–[S23], [S25]
DM4: Ca	apacity	Uses no. of work items (BM6) in time of BM4 and BM5	[S2], [S5], [S6], [S8], [S14], [S20], [S26]
DM5: Pro	ofit and loss	Costs (BM9) and revenues (BM10)	[S2], [S6], [S8], [S17]
DM6: Inv	ventory	Uses no. of work items (BM6), value transitions (BM3)	[S5], [S7], [S11], [S14], [S20], [S22], [S25]
DM7: Ti	me efficiency	Uses value added time (BM4) and non-value added time (BM5)	[S2], [S6], [S20]
DM8: Va	alue efficiency	Investment (BM9), financial revenues (BM10) and duration (BM1)	[S7]
DM9: Ne	et promoter score	Uses perceived customer satisfaction (BM20)	[S6]
DM10: Ini	itial quality	Uses no. of work items (BM6) and fault slippage (BM12)	[S8], [S20]
DM11: Re	eturn on investment	Uses costs (BM9) and revenues (BM10)	[S2], [S6]
DM12: De	esign debt	Object oriented measures such as BM16	[S16]
DM13: Re	ework rate	Uses no. of work items (BM6) with defects	[S11], [S20]



Fig. 5. Model of the related measures.

TABLE VII	
INDICATORS	

ID	Indicator	Relevant measures	Source
I1:	Cumulative flow diagram	Duration (BM1), value transition (BM3) and work items (BM6)	[S5], [S7], [S8], [S11], [S22], [S24]
I2:	Regression	Potentially any measure (e.g. DM2 and DM3) over a duration (BM1)	[S6]–[S8], [S11], [S13], [S22]–[S25]
I3:	Box-plot	Measures on ratio and interval scale (e.g. DM3)	[S11]
I4:	Burndown chart	Duration of time (BM1), no. of handoffs (BM3), no. of work items (BM6),	[S2], [S22], [S26]
		throughput (DM1)	
I5:	Control chart	Measures on ratio and interval scale (e.g. DM4, DM6 and DM13)	[S5], [S11]
I6:	Utilization	Inventory (DM6) and capacity (DM4)	[S5], [S6], [S20], [S26]
I7:	Design debt advice func.	Design debt (DM12)	[S16]
I8:	Variance	Duration of time (BM1), no. of handoffs (BM3), no. of work items (BM6)	[S7]
I9:	Due date performance	Lead time (DM3), duration of time (BM1)	[S8], [S25]
I10:	Relative comparison	Inventory (DM6) and time efficiency (DM7)	[\$25]
I11:	Cost of delay	Profit and loss (DM5) and duration of time (BM1)	[\$2]
I12:	Cost model	Inventory of investment, work done and waste (DM6)	[S7]
I13:	Overall state	Inventory (DM6) and capacity (DM4)	[85]
I14:	Histogram	Potentially any measure (e.g. DM3)	[S8]

relatedness of measures is visualized as a model (see Figure 5), the dependent measures related to the indicators and derived measures are emphasized by the frequency of associations. In addition to BM1 and BM6, the base measures consisting of the activity transitions (BM3), financial revenues (BM10) and financial costs (BM8) are also frequently used.

B. Models of measures

The purpose of the ISO/IEC 15939 standard is to define, "(...) a suitable set of measures that address specific information needs" [4]. Using the ISO/IEC 15939 standard a simplified meta-model described in Section II-A was proposed. The meta-model (see Figure 1) was then used to instantiate a set of models of the measures. Each model of measures is defined so an organization can understand what measures, entities and attributes are needed in order to answer a specific information need. This benefits organizations, because they are able to reuse knowledge which has been shared in research.

In this thesis, the meta-model is used to instantiate models of measures in Appendix D. All of the models use the same visual ranking of elements as seen in Figure 6. The author provided Figure 6 as an instantiation of the meta-model which should be used as a recommendation for how to structure models of measures. The software used to create the models was Graphviz⁶, which is an open source software solution that can structure the measures as graphs. The author has included the source code in Appendix C that is needed to create Figure 6. The source code has been provided due to the fact that it should be used as the basis for the future description and

⁶Software for structuring graphs: http://www.graphviz.org



Fig. 6. Template for the model of measures.

structuring of measures.

In order to communicate the measures used in lean software development literature, the author has created models which are presented in Appendix D. Figure 6 presents how the measures are laid out based on the proposed meta-model from Figure 1. At the top of the model there is an information need, which is associated with its subordinate indicator. The indicator's subordinate measures include derived measures as well as base measures. Derived measures use base measures and base measures are associated with attributes of entities. Variations are possible in the model as long as it conforms to the meta-model in Figure 1.

To answer the information need of where bottlenecks are in the process (see Figure 7), the information need of, "*where is there a bottleneck in the process?*" is presented at the top of the model. Below it there are the measures consisting of indicators (I), derived measures (DM) and base measures (BM). Finally, below the base measures are their corresponding attribute and entity. In each diagram, multiple measures are shown if they have been identified in the literature. For example, inventory is used only in [S14] to identify bottlenecks, but is included in the model. As seen by using models of measures it is possible to address information needs using a structured approach which is based on evidence from the lean software development literature.

Research on lean software development commonly identifies bottlenecks in the process using only the regression of the throughput [S7], [S8], [S11], [S24]. This can be seen in Figure 7. However, the model is extended in [S14] to use the regression of individual work item queues, i.e. the inventory. Models of measures which occur through numerous sources benefit by being modelled, because it makes them more usable. As a result, the author has modelled the information needed for bottleneck identification.

An additional example of a model of measures can be extracted when examining the lead time performance. This is important for organizations who are lean since they want to deliver quickly. Also, service level agreements enforce strict deadlines so it is important for organizations to know how good their lead time performance is. Literature on lean software development uses four indicators to measure the lead time performance. These are: regression [S8], [S22]; box-plot [S11]; histogram [S8] and due date performance [S8], [S25]



Fig. 7. Model of measures for bottleneck identification.



Fig. 8. Model of measures for lead time performance.

(see Figure 8). Each indicator takes a different perspective on lead time performance. For example, using a histogram helps to identify outlying lead times needing improvement in contrast to regression which focuses on the average lead time trend over time.

Models package measures so they are useful to organizations and show possibilities for how they can be used in new ways. For example, based on an organizations need, Figure 7 can be used to quickly understand which measures are needed to determine where the bottlenecks are the in the process. And different measures are simultaneously seen which show the possibilities for fulfilling the information need. Alternatively, if an organization is planning to improve its lead time then Figure 8 identifies a number of indicators.

In [16] it is noted that using the measurement information model sped up the development of measurement systems by 50% and reduced the size of the measurement systems by 35%. As a result, by using the models of measures in Appendix D based on the simplified ISO/IEC 15939 measurement information model organizations can more effectively and quickly use the appropriate measures based on their information needs.

VII. ANALYSIS

A. Increased attention on lean measures

The trend analysis has shown that the number of studies on lean software development measures has drastically increased in 2010 (see Section IV). Pernstål *et al.* [17] conducted a systematic review on lean software development in the time period of 1990-2010. There are eleven studies which are in common between Pernstål *et al.*'s literature review and this thesis on lean measures. In the time before and excluding 2010 this study had 11 sources focusing on measures, while Pernstål *et al.*'s has 26 sources for the same time period from a broader set of lean software development related literature. Even though this thesis focuses on lean software development measures only, a total of 27 sources were identified of which the majority come from 2010 and onwards. This clearly indicates that the interest in lean software engineering measures has drastically increased from 2010 onwards. The trend identified emphasizes the practical and research relevance of studying lean software measures.

B. Lean or Agile measures?

The mapping of a set of relevant lean software development measures is ambitious. A contributing factor is that the field of measure in lean has clearly been influenced by other fields. One example of this is the burndown chart, which focuses on time-boxed iterations. Instead of using time-boxed intervals, lean focuses on releasing minimum marketable feature sets and establishing a release cadence [S8].

A driving factor behind the use of burndown charts is that many organizations combine lean with other agile practices. This is due to the fact that the combination of lean and agile represents a significant body of the lean literature [18]. A derived measure commonly used with burndown charts provides further evidence of this phenomenon. In a number of cases velocity [S23], [S26]; which is terminology commonly used with the burndown chart. is used instead of throughput.

There are further issues that arise where the literature seems to contradict itself. The concept of "stopping the line" [S6] is found in lean when defects are discovered. It prevents the build-up of inventory after a defect is found - allowing for it to be quickly corrected in production. Poppendieck and Poppendieck [S6] describe an example of a company, which does not need to track defects, because they are immediately fixed upon discovery. The author believes that the focus on defect backlogs is therefore not in alignment with lean principles. This can be seen in a number of cases, for example when forecasting the amount of defects by using the defect backlog [S10]. The author believes that lean measures of defects should focus on preventing the build up of any sort of inventory of defects by "stopping the line" of production of software and preventing the rework of defects, e.g. by measuring the failure load (BM2).

C. Need for the evaluation of measures

Only nine studies evaluated lean measures in industry as shown in Section IV. Evaluation studies have primarily come from Scandinavia and have focused on a small set of companies (in particular Ericsson). Hence, the results obtained have limitations in generalizability. Future evaluation needs to understand why and how measures are used in practice by examining different contexts. For instance, whether there is a difference between large and small software companies that use lean practices.

Furthermore, one type of measure, i.e. indicators, needs further evaluation. This thesis has identified that they can be applied with a wide range of underlying measures, but have been until now only applied to a few examples (see Table VII). Future research is needed to examine the benefits for using specific indicators, and the reasons why. One example of an evaluation in literature is by Anderson [S8] who writes that histograms for lead time are preferable in situations where individual outlier values are of interest, while regression presents the overall lead time trend over time. Research should provide evidence that the currently used indicators are the most suitable and explore new applications of indicators which could provide more relevant information based on the foundation of this mapping.

D. Focus in relation to lean principles

Lean software development measures were mapped to the lean principles based on the seven lean principles listed in Section II. The mapping was conducted using the context in which the measures were presented from the literature. Because the principles are important for lean software development, the mapping provides information about where there are gaps in the principles currently. Of the seven principles, four were well-covered while three lacked relevant measures as presented by the literature (see Table VIII). The principles which have been identified as gaps in the literature include create knowledge, defer commitment and respect people. The principle to create knowledge was supported by only one base measure, while defer commitment had no measures. Additionally, the principle of respect people had only two supporting base measures. Future research needs to examine these principles carefully as they present opportunities for the definition of new measures given their importance to lean software development.

The principle to *create knowledge* means quick feedback loops and failures that increase learning in the organization. The only measure for the principle was for the perceived skill level of workers, which was used to determine the knowledge in the organization. Although not directly related to creating knowledge, the measure focuses on identifying knowledge in order to understand the capacity for work. Further measures that focus on measuring the learnings from failures and feedback loops could help to form future research for the principle of *create knowledge*.

The principle of *defer commitment* recommends to make decisions that will not commit the organization until the last possible moment. This allows for decision making to occur based on facts as a pose to predictions earlier in the process. Since there are no measures in the literature which the author identified, adjoining fields of research should be examined for insight. For example, the confidence of determining the size and effort of requirements can be estimated using an ambiguity estimator as in [19]. The ambiguity estimator is one example of a measure which could be used in order to inform the need to defer commitment.

The principle to *respect people* is fulfilled when people are provided with a purpose, challenged and given responsibility. There is only one example of a measure from the literature that seeks to measure this principle. It is found in [S26] as



Fig. 9. Throughput (DM1), Cycle time (DM2) and Inventory (DM6).

an emotional seismograph which is used to understand how satisfied employees are. However, concrete information about the measure is not available, so it would be difficult for another organization to apply the same measure. Further research could look to develop measures for how much responsibility workers are given in order to understand the principle of *respect people*.

Software engineering's need to cope with the human factor and variability in development make the principles to *respect people*, *create knowledge* and *defer commitment* important. Future research can begin by examining these three principles which have not been more extensively covered by measures in the literature.

E. Improved representation of shared dependent measures

When codifying measures, there emerged a pattern of measures that shared dependent measures. In some cases, measures that rely on the same dependent measures are equivalent to each other and can be used without additional effort.

An example of this is the relation of throughput, cycle time or inventory as presented in [S6] which is identified as Little's law (see Equation 1).

$$Cycle time (DM2) = \frac{Inventory (DM6)}{Throughput (DM1)}$$
(1)

From the equation it is clear that the measures are mathematically related. Additionally, it can be seen from the model of the measures (see Figure 9) that the measures rely on similar base measures. If an organization was already measuring the throughput and inventory, then it would be possible for them to measure the cycle time as well without additional effort.

The measures are able to be used when calculating each other, because of their reliance on the same base measures. Since any two of the three measures can be used to calculate the third mathematically, these measures each contain dependencies useful for determining each other. Currently, using visual inspection it is possible to noticeably represent this in a model of measures such as Figure 9. This could be also expanded to apply to the full measurement information metamodel presented in Appendix A if more detail was available.

The benefit of making the relations of measures visible is that it provides the means for organizations to see what other measures are possible to use given the currently used measures.

F. Guidelines for measures in software process improvement

Since measures provide feedback of improvement and are therefore key to software process improvement there will continue to be a need to use measures in research. In the literature the author was not able to identify a consistent way in which measures were described. Within the field of lean software development numerous examples of how measures have been defined exist.

Staron *et al.* [S12] has represented measures using the Unified Modeling Language⁷-like notation which provides richer information about the measure. An equation was also given that clearly defined how the measure could be applied.

Staron and Meding [S14] in their pape clearly define the measures used in a separate subsection focused on the definitions of measures. They then provide an example of how the measures can be implemented.

Petersen and Wohlin [S5] used QIP [20] and followed a six step process which included setting quantifiable goals and measures. The goals identified could be considered information needs. The measures were then applied and explained later in the paper using an example, figures, equations and plain text. Petersen and Wohlin later defined measures mathematically in [S7].

Many papers such as [S15], [S25], [S26] only use measures as a supporting argument for the conclusions of the paper. As a result, measures are not the only focus of the paper, so it is important that they be described succinctly and correctly in order to allow space for the papers other contributions. As a result, this thesis makes a contribution for future research on lean software development measures by proposing a set of minimal guidelines, these are:

- measures shall be well-defined by either
 - a citation to a rigorous definition,
 - or the definition of the measure and an explanation of how the dependent measures are combined.
- use of the simplified meta-model of measures (see Figure 1) to represent measures (see template in Figure 6) shall be used.

These guidelines are helpful for researchers who write about lean software development and are focused on measures. The guidelines are purposely minimal, because using the full meta-model adapted from the ISO/IEC 15939 measurement information model (see Appendix A) is too information rich. The author is not aware of any research focused on lean software development which uses the full ISO/IEC 15939 measurement information model. Therefore, the simplified meta-model of measures can be used instead in order to define measures well.

The simplified version of the full meta-model in Figure 1 still contains crucial information, i.e. well-specified information needs, attributes, entities and all measures are a part of the meta-model in Figure 1.

A more rigorous approach to measures will be enabled by following the guidelines listed. The benefits will be that measures will be more applicable to practitioners since they will be more easily used and compared. It also will contribute

⁷Unified Modeling Language specification: http://www.omg.org/spec/UML/

TABLE VIII MAPPING OF PRINCIPLES TO MEASURES

Principle	Base measures	Derived measures	Indicators
Eliminate waste	BM4, BM5, BM6, BM8, BM9	DM7, DM11, DM9, BM20	15, 16, 110, 112
Build quality in	BM2, BM12, BM13, BM14, BM15, BM16, BM17, BM18	DM9, DM10, DM12, DM13	I7
Create knowledge	BM22	-	-
Defer commitment	-	_	-
Deliver fast	BM1, BM3, BM7, BM11, BM19	DM1, DM2, DM3	I2, I3, I4, I9, I14
Respect people	BM21	_	-
Optimize the whole	BM10, BM20	DM4, DM8	I1, I8, I11, I13

to research as the relationships and use of measures will become more accessible to collect well-defined measures from.

VIII. CONCLUSIONS

The author carried out a systematic mapping study to structure the field of lean software development measures by using a classification scheme (i.e. the simplified meta-model). The simplified meta-model (see Figure 1) was proposed by the author as a scaled down version of the meta-model adapted from the ISO/IEC 15939 measurement information model in Appendix A. The proposed meta-model was then used to codify measures (see Section V) and enables models of the measures to be created for Figures 8, 9 and Figures 12 through 20 in Appendix D.

The identification of equivalent measures from dependent measure is enabled by the modelling of measure's associations, which was proposed and demonstrated in the thesis. In addition, models of measures were instantiated for information needs of bottleneck identification (see Figure 7) and lead time performance (see Figure 8).

In order that future research on lean software will be rigorous, the author proposed guidelines to define or cite definitions of measures and instantiate models of measures using the meta-model from Figure 1.

RQ1: Which trends can be seen in terms of the number of publications and publication forms over time? The author identified a significant increase in literature on lean software measures in 2010 and onwards, indicating its relevance for research and practice.

RQ1.1: How were measures evaluated in terms of research types? The most frequent way of reporting lean measures were through evaluation and experience reports. There were only a few opinion papers, solution proposals, and validation research papers presented. Nine studies were conducted using evaluation in industry. However, these originated primarily from Scandinavia focusing on only a few companies, hence further studies in different contexts are needed.

RQ2: Which measures and indicators have been proposed? State-of-the-art measures were distilled from the lean software development literature. The thesis identified 22 base measures, 13 derived measures, and 14 indicators from the literature. The measures are presented in Tables III-VII. The most frequently described indicators from literature were the cumulative flow diagram (I1) and regression (I2). From the derived measures the most common measures were throughput (DM1), cycle time (DM2), lead time (DM3), capacity (DM4) and inventory

(DM6). Finally, the base measures that were most frequently described or used were duration in time (BM1), value added time (BM4), non-value added time (BM5) and number of work items (BM6).

By mapping the measures to lean principles (see Table VIII), the author found that there were no measures related to the principle of *defer commitment* and only one measure for *respect people* and *create knowledge* each. These are important principles for which there were only a few measures identified. Therefore, lean software development researchers should look to identify measures that satisfy these principles.

Future research should focus on identifying measures where there are gaps in the lean principles. Moreover, indicators need further comparison and analysis in order to understand their benefits for different information needs. Finally, a survey should be conducted using the thesis as the basis for understanding how measures are used in software development organizations using lean.

REFERENCES

- [1] J. M. Morgan and J. K. Liker, *The Toyota product development system*. Productivity press New York, 2006.
- [2] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information & Software Technology*, vol. 50, no. 9-10, pp. 833–859, 2008.
- [3] K. Petersen, "Is lean agile and agile lean," A comparison between two software development paradigms, Modern software engineering concepts and practices: advanced approaches, IGI Global, pp. 19–46, 2011.
- [4] "IEEE standard adoption of ISO/IEC 15939:2007 systems and software engineering measurement process," *IEEE Std 15939-2008*, pp. C1–40, 2009.
- [5] J. K. Liker, The Toyota Way. Esensi, 2004.
- [6] J. P. Womack, D. T. Jones, and D. Roos, *The machine that changed the world: The story of lean production–Toyota's secret weapon in the global car wars that is now revolutionizing world industry.* Simon and Schuster, 2007.
- [7] M. Staron, W. Meding, G. Karlsson, and C. Nilsson, "Developing measurement systems: an industrial case study," *Journal of Software Maintenance*, vol. 23, no. 2, pp. 89–107, 2011.
- [8] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in 12th International Conference on Evaluation and Assessment in Software Engineering, vol. 17, 2008, p. 1.
- [9] M. Dixon-Woods, S. Agarwal, D. Jones, B. Young, and A. Sutton, "Synthesising qualitative and quantitative evidence: a review of possible methods," *Journal of health services research & policy*, vol. 10, no. 1, pp. 45–53B, 2005.
- [10] R. Wieringa, N. A. M. Maiden, N. R. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," *Requir. Eng.*, vol. 11, no. 1, pp. 102–107, 2006.
- [11] B. A. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," 2007.
- [12] S. Jalali and C. Wohlin, "Systematic literature studies: database searches vs. backward snowballing," in *ESEM*, 2012, pp. 29–38.

- [13] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.
- [14] D. Cruzes and T. Dybå, "Synthesizing evidence in software engineering research," in *ESEM*, G. Succi, M. Morisio, and N. Nagappan, Eds. ACM, 2010.
- [15] L. Rising and N. S. Janoff, "The scrum software development process for small teams," *IEEE Software*, vol. 17, no. 4, pp. 26–32, 2000.
- [16] M. Staron, W. Meding, and C. Nilsson, "A framework for developing measurement systems and its industrial evaluation," *Information & Software Technology*, vol. 51, no. 4, pp. 721–737, 2009.
- [17] J. Pernstal, T. Gorschek, and R. Feldt, "The lean gap: A review of lean approaches to large-scale software systems development," *Journal* of Systems and Software, 2013.
- [18] X. Wang, K. Conboy, and O. Cawley, "Leagile software development: An experience report analysis of the application of lean approaches in agile software development," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1287–1299, 2010. [Online]. Available: http: //www.sciencedirect.com/science/article/pii/S0164121212000404
- [19] D. C. Gause and G. M. Weinberg, *Exploring requirements: quality before design*. Dorset House Pub. New York, 1989.
- [20] V. R. Basili, "The experience factory and its relationship to other quality approaches," Advances in Computers, vol. 41, pp. 65–82, 1995.

Systematic mapping references

- [S1] M. Poppendieck and M. A. Cusumano, "Lean software development: A tutorial," *IEEE Software*, vol. 29, no. 5, pp. 26–32, 2012.
- [S2] M. Poppendieck, *Lean software development: an agile toolkit.* Addison-Wesley Professional, 2003.
- [S3] P. Middleton, "Lean software development: Two case studies," Software Quality Journal, vol. 9, no. 4, pp. 241–252, 2001.
- [S4] S. Mujtaba, R. Feldt, and K. Petersen, "Waste and lead time reduction in a software product customization process with value stream maps," in *Australian Software Engineering Conference*, 2010, pp. 139–148.
- [S5] K. Petersen and C. Wohlin, "Software process improvement through the lean measurement (spi-leam) method," *Journal of Systems and Software*, vol. 83, no. 7, pp. 1275–1287, 2010.
- [S6] M. Poppendieck and T. Poppendieck, Implementing Lean Software Development: From Concept to Cash. Addison-Wesley Professional, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1196372
- [S7] K. Petersen and C. Wohlin, "Measuring the flow in lean software development," *Softw., Pract. Exper.*, vol. 41, no. 9, pp. 975–996, 2011.
- [S8] D. J. Anderson, *Kanban*. Blue Hole Press, 2010.
 [S9] M. P. Ware, F. G. Wilkie, and M. Shapcott, "The application of product measures in directing software maintenance activity," *Journal*
- of Software Maintenance, vol. 19, no. 2, pp. 133–154, 2007.
 [S10] M. Staron, W. Meding, and B. Söderqvist, "A method for forecasting defect backlog in large streamline software development projects and its industrial evaluation," *Information & Software Technology*, vol. 52, no. 10, pp. 1069–1079, 2010.
- [S11] K. Petersen, "A palette of lean indicators to detect waste in software maintenance: A case study," in XP 2012, 2012, pp. 108–122.
- [S12] M. Staron, W. Meding, and M. Caiman, "Improving completeness of measurement systems for monitoring software development workflows," in *Software Quality Days 2013*, 2013, pp. 230–243.
- [S13] P. Middleton, P. S. Taylor, A. Flaxel, and A. Cookson, "Lean principles and techniques for improving the quality and productivity of software development projects: a case study," *International Journal of Productivity and Quality Management*, vol. 2, no. 4, pp. 387–403, 2007.
- [S14] M. Staron and W. Meding, "Monitoring bottlenecks in agile and lean software development projects - a method and its industrial use," in *PROFES 2011*, 2011, pp. 3–16.
- [S15] A. Janes and G. Succi, "To pull or not to pull," in OOPSLA Companion 2009, 2009, pp. 889–894.
- [S16] J. Heidenberg and I. Porres, "Metrics functions for kanban guards," in ECBS 2010, 2010, pp. 306–310.
- [S17] S. Raman, "Lean software development: Is it feasible?" in *Digital Avionics Systems Conference*, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE, vol. 1, 1998, p. C131. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=741480
- [S18] J. M. Sutton, "Lean software for the lean aircraft," in *Digital Avionics Systems Conference*, 1996., 15th AIAA/IEEE, 1996, p. 4954. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=559134
- [S19] K. Vilkki, "When agile is not enough," in LESS, 2010, pp. 44-47.

- [S20] H. McManus, "Product development value stream mapping (PDVSM) manual," *Lean Aerosp Initiative*, 2005.
- [S21] M. Taipale, "Huitale a story of a finnish lean startup," in LESS, 2010, pp. 111–114.
- [S22] H. Kniberg, Kanban and Scrum-making the most of both. Lulu. com, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1841732
- [S23] R. Polk, "Agile and kanban in coordination," in AGILE 2011, 2011, pp. 263–268.
- [S24] B. Swaminathan and K. Jain, "Implementing the lean concepts of continuous improvement and flow on an agile software development project: An industrial case study," in AGILE India (AGILE INDIA), 2012. IEEE, 2012, pp. 10–19.
- [S25] M. Seikola, H.-M. Loisa, and A. Jagos, "Kanban implementation in a telecom product maintenance," in *EUROMICRO-SEAA 2011*, 2011, pp. 321–329.
- [S26] J. Prochazka, M. Kokott, M. Chmelar, and J. Krchnak, "Keeping the spin - from idea to cash in 6 weeks: Success story of agile/lean transformation," in *ICGSE 2011*, 2011, pp. 124–130.
- [S27] B. Vodde, "Measuring continuous integration capability," CrossTalk, 2008.

APPENDIX A

ISO/IEC 15939 MEASUREMENT INFORMATION MODEL



Fig. 10. ISO/IEC 15939 measurement information model

APPENDIX B THEMATIC CODING

TABLE IX

Code	Description
А	Publication details
Al	What is the title?
A2	Who is the author(s)?
A3	What is the year of publication?
A4	Is the publication a conference, journal or book?
В	Research paper methodology
B1	Is the research paper empirical?
B2	What are the research methods used?
B3	What is the type of research (see [10]) conducted?
С	Measure details
C1	What is the name of the measure?
C2	What is the short description of the measure?
C3	What information need does the measure answer?
C4	What are the relevant lower level measures needed for the
	measurement (i.e. derived measure and/or base measure)?
C5	What are the attributes the of the measures?
C6	What are the entities of the previously identified attributes?

APPENDIX C

Source code for the template for the model of measures (see Figure 6)

Appendix D Models of measures















Fig. 11. Model of measures for cost savings and work off-loading.



Fig. 16. Model of measures for the distribution of work items and priorities.



Fig. 19. Model of measures for the time efficiency.



Fig. 17. Model of measures for estimating the pace of work.





Fig. 20. Model of measures for the organizations utilization.

Fig. 18. Model of measures for the level of defect output.