Synthesizing a Comprehensive Framework for Lean Software Development

Henrik Jonsson
Technical System Development
Etteplan Oyj
Västerås, Sweden
henrik.jonsson@etteplan.com

Stig Larsson, Sasikumar Punnekkat
School of Innovation, design and engineering
Mälardalen University
Västerås, Sweden
{stig.larsson,sasikumar.punnekkat}@mdh.se

Abstract— **Lean principles, originating from Japanese automotive industry, are anticipated to be useful to improve software development processes. Albeit its popularity there is still no generally accepted, clear and detailed definition of what lean software development actually means. This makes it difficult to perform research on the effects of lean software development and determine its usefulness in various contexts. To fill in that research gap this paper analyzes the state of the art based on twenty key Lean concepts derived from nine seminal sources identified in a systematic literature review. The original explanations of the key concepts have been elaborated further and synthesized into a framework for lean software development consisting of a set of goals, recommended activities and practices. The detailed results for the key concept Value are reported. The proposed framework is expected to serve as a basis for further research and for Lean assessment of organizations.** *(Abstract)*

Keywords-Lean Software Development; Systematic Literature Review; State of the art, Software Process Improvement, (key words)

I. INTRODUCTION

'Lean' principles derived from Toyota car production are now spreading into many other domains, with expectations of radically improved process performance in those domains as well. Software development is not an exception and the interest for how to apply the lean principles to development of software has grown rapidly. However, the very basic question of what lean software development means is still not clearly answered [8]. As long as no unified definition of what lean software development is exists, it is difficult to perform research on it and relate it to agile and other streams of influences in the software process improvement domain. From practioners' point of view there is also valuable to see what Lean can means more concretely for software development. As many organizations are adopting Lean in other areas, either on the manufacturing floor only, or to the enterprise as a whole there is a need for more concrete guidance of what Lean means for the software development departments in particular.

To make it easier to perform research in this area and to provide more concrete guidance to organizations, we have derived a framework for Lean Software Development from multiple recognized sources. The starting point for this framework was a systematic literature review (SLR) identifying nine seminal sources for Lean Software Development and 44 different Lean principle statements (Section IV). Based on their descriptions, 20 key Lean concepts were compiled. Furthermore, specific goals, recommended activities and examples of practices for each key concept were extracted and synthesized in a traceable manner (Section V). Section II relates this study to others work and section III presents the details about the research method applied. Section VI discusses the findings and section VII concludes the paper.

II. BACKGROUND AND RELATED WORK

The term 'lean production' was coined by researchers from Massachusetts Institute of Technology (MIT) assigned to study why Japanese automotive companies, mainly Toyota, had managed to increase their productivity and steadily taking market shares from the American ones [25]. What they found was a production philosophy and set of principles, collectively called Lean. In Japan this is primarily known as 'The Toyota Way' [19]. While many major automotive industries now have integrated large parts of these ideas into production to remain competitive, those ideas have now been spread to other industrial domains. As shown in this paper, the ideas to apply the Lean ideas to software development has roots back to the MIT research group coining the term [16], but it has mostly been popularized by the books by the Poppendiecks [21],[22],[23] as shown in this study.

Previous reviews [3],[14] of applications of Lean principles to software development have summarized the genealogy of lean software development. A recent paper [10] compiled principles from selected sources in order to identify common high level values for lean soft development analogous to the agile values expressed in the Agile manifesto (www.agilemanifesto.org). However, none of these studies was reported to be based on a systematically performed literature review as the one reported in this paper. And none of these has attempted to synthesize a detailed description of what a software development organization should do to be Lean.

Extensive work to describe the general taxonomy of Lean has been performed by MIT in the Lean Aerospace Initiative (LAI) and an assessment tool for guiding organizations to adopt lean on an enterprise level has been developed [18]. This LAI Self Assessment Tool (LESAT) has recently been extended for software development, mainly by adding common references to the agile manifesto and the Poppendiecks' [21] popular interpretation of lean software development [7]. Since the focus is still on enterprise level, the framework does not provide so much concrete guidance for software development organizations.

III. Research method

The overall approach to synthesize a framework for Lean Software Development starting from a systematic literature review is summarized in Figure 1.

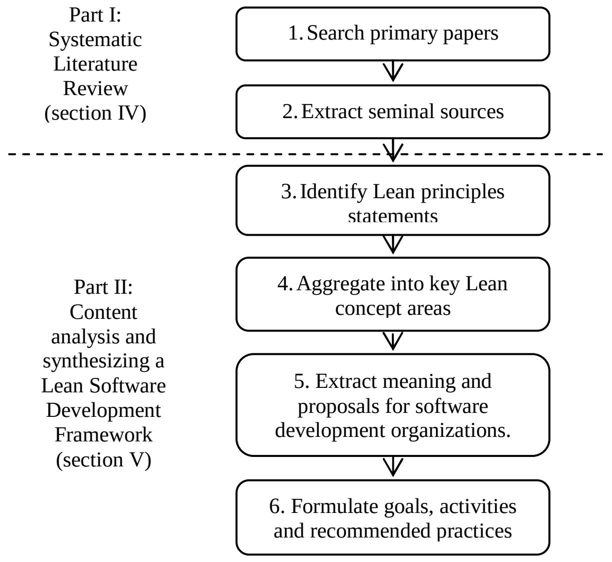
Part I: Systematic Literature Review (section IV)
1. Search primary papers
2. Extract seminal sources
3. Identify Lean principles statements
Part II: Content analysis and synthesizing a Lean Software Development Framework (section V)
4. Aggregate into key Lean concept areas
5. Extract meaning and proposals for software development organizations.
6. Formulate goals, activities and recommended practices

Figure 1. Research workflow

This study began with a systematic literature review with an approach and a protocol according to the guidelines given by Kitchenham [9] (part I). The research question relevant to this part of the study was *which sources the primary papers refer to when explaining Lean*. Initial primary papers were searched (step 1) using the EBSCO Discovery Service including the databases ABI Inform, ACM Digital Library, SpringerLink and ScienceDirect and IEEE Xplorer, October 11, 2012. The exact search terms were the combination of "lean" and "software" in all metadata fields, with subject terms "computer software development", "software engineering", "software process" or "computer science". Only peer reviewed sources in English were included. In an initial screening, papers only mentioning lean in some other context in the abstract or authors field (like Mr. Lean) were excluded. When in doubt, the full paper was reviewed to determine whether it should be included or not. To extract seminal sources, defined as what the authors of the primary papers refers to when explaining lean (step 2, Table I), the full paper text and reference sections in each included paper was read. The total number of references to each source was then calculated. To cross-check the results the number of citations to the identified seminal sources reported by Google Scholar was also obtained (October 25, 2012).

The second part of the study was a content analysis of the identified seminal sources (all books), with the goal to identify important Lean concepts, recommended activities and practices for software development. First, the seminal sources were studied to identify sets of principle statements summarizing Lean (step 3, Table II). The principle statements were analyzed for recurring keywords to identify a common set of key concepts (Table III). In case of doubt, the extended descriptions of the principles by the seminal authors were used to guide the mapping of principles into common concepts. For selected concept areas, each of the seminal authors' description of the concept was systematically scrutinized to identify and extract quotations with page references to descriptions of the actually meaning, motivation and recommended practices for organizations developing software. The table of contents and the index sections guided the search. The seminal authors' views were then compared to find similarities and discrepancies, resulting in a synthesized background description for each selected concept area. Based on that synthesis a number of proposed general activities were formulated for each key concept area. Any associated recommended concrete methods or practices from the seminal authors were also recorded. Finally, overall concept area goals were formulated to summarize the seminal authors view.

IV. Systematic Literature Review Results

As part of a systematic literature study the sources that have been used in the academic computer science was identified. These recognized sources were later used as the basis for building a framework for lean software development as described in section V.

A. *Seminal sources identified*

In the first database lookup 140 hits were found, including six duplicates. After reading through the abstracts (and full paper when in doubt) 30 peer-reviewed journal papers remained. By looking at which references those 30 primary papers made when explaining lean and lean software development, nine seminal sources (all books) were identified. Table I summarizes the number of references to each of the identified seminal source. The seminal sources were divided into those that refer to descriptions mainly concerning 'lean production' (manufacturing), those that describe Lean product development in general (not only software) and those applied to software development in particular. Total number of references to these sources in Google Scholar is also reported. Note that the latter includes references from other sources than peer-reviewed papers as well as references not focusing on software development.

TABLE I. SEMINAL SOURCES FOR LEAN SOFTWARE DEVELOPMENT IDENTIFIED THROUGH A SYSTEMATIC LITTERATURE REVIEW

Author	**Ref.+ Year**	**References from primary papers**	**Google Scholar references**
Lean Production			
Womack and Jones	[26] 1991	15	8942
	[27] 2003		3694
Liker	[11] 2004	5	1723
Ohno	[19] 1998	2	2588
Lean Software Development			
Poppendieck	[21] 2003		525
	[22] 2006	13	183
	[23] 2010		27
Andersson	[2] 2010	6	58
Middleton and Sutton	[15] 2005	3	33
Coplien and Bjornvig	[5] 2010	1	27
Morgan	[16] 1998	2	4
Lean Software Development			
Morgan and Liker	[17] 2006	1	373

Most of the found primary papers' authors refer to Womack and Jones' books [26],[27] when explaining Lean thinking in general. However, both Womack and Jones, and Liker refer back to Ohno [19] and other Japanese sources who initially invented the Toyota production system.

When it comes to interpretation of lean in software development, the Poppendiecks were the far most cited source, with three books in the area [21],[22],[23]. More recently, Andersson has also become popular, mainly for his work on the software Kanban project management method [2]. Middleton and Sutton are less cited but provide more concrete guidance on how to apply lean concepts in embedded software development. In addition, Middleton et al. also provides some empirical study reports [12],[13],[14] identified as primary papers in the systematic literature part. Coplien and Bjornvig [5] also give some concrete guidance to software, but with main focus on effective functional specifications and software architecture. Morgan's master thesis from 1998 was the oldest reference seminal paper found, but the work of Ayoyama [4], identified as a primary paper, shows that lean principles in software has even older roots from the Japanese industry, even before the term Lean was introduced.

A third category of seminal sources identified by studying the primary papers was the one about lean product development (LPD), not particularly focusing on development of software. Morgan and Liker's book [17] mainly describing the product development environment at Toyota was the only source in this group referred to by the primary papers. This and other LPD sources, for instance Ward [25], were also referred to by a few of the seminal lean software development sources described above. As many of the challenges of product development is not unique to the development of software, software engineering practitioners and researchers can at least partly be helped in their Lean journeys by studying the LPD literature more carefully. This study includes content from Morgan and Liker's book [17].

B. *Lean principles identified*

After identifying the seminal sources as described above, these were examined in more detail in order to extract summarizing key principles for Lean in general and lean software development in particular. This section and Table II summarize those principles and puts them side by side.

Principles from the Lean Production seminal sources

Liker's [10] starting point for explaining the 'The Toyota way' is the 4P's: Philosophy, Process, People/Partners and Problem Solving. The main Philosophy is to work for adding value to customers and society to enable economic and social growth. The Process parts emphasizes that we have to take the right process path to reach our goals efficiently and to have a long-term perspective. The People and Partner part, also known as "respect for humanity", emphasizes that production and development is a learning and challenging human environment, and that we have to work closely with both customers and suppliers, helping both to improve their businesses. Finally, the Problem solving part stresses the importance to continuously solve root problems and continuously learn. Besides these key pillars, Liker summarizes Lean with 14 principles mostly from a production perspective including process, organization as well as technological issues (see Table II).

Womack and Jones [27] condense Lean Thinking into five important concepts, namely Value, Value stream, Flow, Pull and Perfection. In contrast to Liker [11] these are mainly process oriented.

While both Liker's [11] and Womack and Jones' [27] descriptions mostly stem from the production floor, the view of Lean is extended Morgan and Liker [17] who describes the Toyota product development system. Although there are much common overlap, the principles stated by Morgan and Liker [17] extends the view of Lean when it comes to product development, more closely related to software development than pure production. The way of front-loading the development process and considering many options (set-based engineering) are examples of process-related aspects unique to production development. The way of aligning and boosting the development organization using functional expertise groups working in integrated product teams lead by a chief engineer is also unique features to the lean product development discussion.

Principles from lean software development seminal authors

Whereas the above principles are for lean production and development in general, this subsection presents and analyses how the seminal sources interpret them for software development. The exact formulation and short explanations of Poppendiecks' principles for Lean Software Development (www.poppendieck.com) seems still to be floating, making it difficult to analyze them definitely. A deeper analysis below (see Table III) showed that Poppendiecks' principles in general much overlap with the Lean production and development principles discussed above when it comes to process-related issues. Organizational and technical issues

TABLE II. LEAN PRINCIPLES IDENTIFIED

Author	Principles
Liker [11]	L1. Base your management decisions on a long-term philosophy, even at the expense of short-term financial goals. L2. Create a continuous process flow to bring problems to the surface. L3. Use "pull" systems to avoid overproduction L4. Level out the workload (heijunka). L5. Build a culture of stopping to fix problems, to get quality right the first time. L6. Standardized tasks and processes are the foundation for continuous improvement and employee empowerment. L7. Use visual control so no problems are hidden. L8. Use only reliable, thoroughly tested technology that serves your people and processes. L9. Grow leaders who thoroughly understand the work, live the philosophy, and teach it to others. L10. Develop exceptional people and teams who follow your company's philosophy. L11. Respect your extended network of partners and suppliers by challenging them and helping them improve. L12. Go and see for yourself to thoroughly understand the situation (genchi genbutsu). L13. Make decisions slowly by consensus, thoroughly considering all options; implement decisions rapidly (nemawashi). L14. Become a learning organization through relentless reflection (hansei) and continuous improvement (kaizen).
Womack and Jones [27]	W1. Value: First identify what really matters to the customer W2. Value Stream: Ensure every activity adds customer value W3. Flow: Eliminate discontinuities in the value stream W4. Pull: Production is initiated by demand W5. Perfection: Retaining integrity via Jidoka and Poka-Yoke
Poppendiecks (www. poppendieck. com)	P1. Optimize the whole (Focus on entire value stream; Deliver a complete product; Think long term) P2. Eliminate waste (Build right thing; Learn; Eliminate trashing) P3. Build quality in (Mistake-proof process; Break architectural dependencies) P4. Learn constantly (Predictable performance is driven by feedback; maintain options, last responsible moment) P5. Deliver fast (Rapid delivery; High quality, low cost are fully compatible; Queuing Theory applies, Managing workflow is easier than managing schedules) P6. Engage everyone (Use semi-autonomous teams; provide challenges and feedback; work for a purpose) P7. Keep getting better (Failure is a learning opportunity; Standards exist to be challenged and improved; use the scientific method)
Andersson [2]	A1. Visualize the workflow A2. Limit work in progress A3. Manage flow A4. Make process policies explicit A5. Improve collaboratively (using models and the scientific method)
Morgan and Liker [17]	M1. Establish customer-defined value to separate value-added from waste. M2. Front-load the product development process to explore thoroughly alternative solutions while there is maximum design space. M3. Create a level product development process flow. M4. Utilize rigorous standardization to reduce variation, and create flexibility and predictable outcomes. M5: Develop a chief engineer system to integrate development from start to finish. M6. Organize to balance functional expertise and cross-functional integration. M7. Develop towering competence in all engineers. M8. Fully integrate suppliers into the product development system. M9. Build in learning and continuous improvement. M10. Build a culture to support excellence and relentless improvement. M11. Adapt technologies to fit your people and process. M12. Align your organization through simple visual communication. M13. Use powerful tools for standardization and organizational learning.

are less covered in the way the principles summarized by the Poppendiecks.

Andersson states five principles for his software Kanban method [2]. Compared to Poppendiecks' principles these are more practice-oriented, mostly focused on the project management issues.

Coplien and Bjornvig [5] summarize the lean principles as the rule of thumb *"everybody, all together, from early on"*. In itself this does not give so much concrete guidance but could help practitioners already trained in Lean thinking to keep aligned with it.

Middleton and Sutton [15] do not try to create an own definition, but discusses the implications of the lean concepts defined by Womack and Jones when applied to software development. This was valuable input the detailed analysis of what Lean means to software development presented below.

V. SYNTHESIS OF A LEAN SOFTWARE DEVELOPMENT FRAMEWORK

Based on the findings from the literature review results presented above, a framework for lean software development was constructed. At the highest level Lean concepts were identified from keywords used in the principle statements formulated by the seminal authors. Their view of each concept and the implications for software development organizations were then systematically analyzed and synthesized into a number of activities to perform and overall goals to strive for in order to be Lean according to these sources.

TABLE III. LEAN KEY CONCEPTS SYNTHESIZED FROM SEMINAL SOURCES

Concept	Liker	Morgan & Liker	Womack &Jones	Poppendiecks	Andersson
Main Philosophy					
Long-term decisions	L1			P1	
System thinking				P1	
Continuous Improvement	L14	M9	W5	P7	A5
Process					
Value		M1	W1	P2	
Value stream			W2	P1	
Flow	L2	M3	W3	P5	A2-3
Pull	L3		W4		
Mistake-proof process	L5		W5	P3	
Waste	L4	M1		P2	
Set-based engineering	L13	M2		P4	
Standardized work	L6	M4		P7	A4
People					
Go see (genchi genbutsu)	L12				
Mentorship leadership	L10	M10			
Supplier integration	L11	M8			
Chief engineer		M5		P6	
Integrated functional expertice		M6			
T-competence	L10	M7		P6	
Technology					
Visualization	L7	M12			A1
Adapt tools to humans	L8	M11			
Powerful tools		M13			

A. Synthesis of Lean concepts

By analyzing the principles from the above sources 20 key concepts for Lean were identified and mapped back to principles of the individual seminal sources (Table III). From this analysis it is clear that no author covers all the aspects of Lean in their own set of specified principles. While the lean software development seminal authors cover the process-oriented concepts well, their set of principles has lower coverage when it comes to organizational and technical features to support the process. It is however important to note that these results are solely based on how the main Lean principles were stated by the seminal authors. Further discussions related to the missing concepts were occasionally found when analyzing the seminal papers further as described below.

B. Implications for software development organizations

After identifying Lean key concepts as described above the views of the different seminal authors about each of the concept were analyzed in more details in a structured manner, with the goal to create a comprehensive framework for Lean Software Development traceable back to seminal sources. First all seminal sources were searched for definitions of the concept (what is it?). Based on that, a general summary of the concept was formulated. Secondly, statements about what a software development organization should think of and do to be lean were extracted from the authors' more detailed description of the phenomena. Recommended practices and methods mentions were also extracted. All this information was recorded in a spreadsheet, one per key Lean concept, with references to source and page numbers. By comparing and aggregating the various seminal sources' views, a number of activities and goals were formulated for each concept area.

The result of this work is exemplified below. Due to space limitations, only the details for the Value concept area are reported. Value was chosen because it is an essential starting point according to Womack and Jones' principles (W1) [27] .

C. Summary for the Value concept area

Several aspects of the value term as used in Lean are discussed by the seminal authors, but a strong common theme as that it is the value for the ultimate customer that must be in the focus all the time. Several seminal authors claim that this focus is easily lost in a development organization, when (1) starting to focus too much on project scope (and not product) time and cost issues [22], (2) too soon turning to discussing financial numbers with upper management [17] or (3) engineers' own interest in new, fancy technology, occasionally developed to overly perfection, takes precedents over the overall goal of creating a good balance of customer needs in the product [27]. To avoid the latter it is important to “keep the customer values in front of the technical people making detailed designed decisions” [21].

Middleton and Sutton summarize value as the collection of all the wants and needs of your customer [15], and point out that value does not just encompass functionality and usability, but also non-functional properties and pricing. Similarly, Morgan and Liker [17] emphasize that both quantitative and qualitative aspects of customer value, including performance, cost and quality, must be addressed. In terms of waste, a well-designed product shall help the ultimate customer to avoid waste in their own value stream [15]. Besides providing exactly the right functionality and nothing more, a Lean software development organization must work hard to minimizing the failure demand (for instance bugs) that annoys and disturbs the customer and creates waste both in the customer’s business and in the developing organization [23]. In other words, a well-designed product shall provide a hassle-free customer experience [27].

D. Activities for the Value concept area

Further analysis of the extracted quotations from the seminal sources resulted in the formulation of a number of activities proposed by the seminal authors to be performed by a software development organization in order to be Lean. Associated concrete practices or methods recommended were also captured. Although these can provide some more guidance, they must be evaluated based on the particular organizational context and how they fit together with upper-stream and lower-stream activities. In the spirit of Lean’s continuous improvement thinking, other existing modified or novel practices should be considered at all times. The activities themselves should also critically be considered in the continuous improvement work.

The resulting activities for the Value concept area are presented above, starting with a synthesized activity statement including a motivation followed by a description summarizing and giving further references to the seminal sources.

V1. To create the right product and avoid rework, all customers need to be identified taking into consideration customer’s customer and any party that can affect the sales and the business benefit of the product.

The very first step is to identify all important customer stakeholders. Understanding who the customers are is central in Lean thinking and to understand value, but not all seminal authors elaborate on how customers are defined and identified. According to the Poppendiecks, customers are anyone who pays for, uses, supports or derives value from the product [23]. Product managers and product owners are not customers [23], but are the ones mainly responsible of transferring customer’s need to the developing organization. Sometimes the direct customer is within the same enterprise, for instance for software supporting the production line. In such cases and other, it is anyway useful to identify and understand the needs of your customer’s customers [15]. Merely identifying the end users is not sufficient. Instead all parties that can influence the ability to sale the product, including government regulators, customer’s management and general public must be taken into consideration [15]. For product development, production and service personnel are also important stakeholders in this context. Often, but not always, the actual production (deployment) effort for software is comparably easy. Anyhow it is important to identify those functions that manage installation and maintenance of the software product, which affects the overall business benefit. One recommended practice to identify customers is *Brainstorming* taking into consideration all those that can affect the purchasing decision [16].

V2. To be able to identify actual customer needs, the engineering lead and other representatives from the development organization need to visit the (typical) customer home turfs to meet, study and interview the customer.

A crucial step is to early get access to ‘gemba’, that is a home place for the end user where the product is intended to be used in the end. Ideally, a chief engineer [17] (also known as ’champion’ [21]) should do this early, then formulate a vision statement and channel this into an effective value stream and to the rest of the organization. Preferably several representatives from the software development organization, not only top management, shall visit this place and observe the customers in their real-world context. This shall be complemented by customer meetings and interviews, explicitly excluding sales personnel [15].

V3. To be able to create a product with well-balanced feature set, all customers’ wants and needs, including performance and cost, need to be systematically captured, analyzed and categorized.

Within this activity implicit and unrecognized needs shall be captured for each customer. To find real customer problems to be solved the *Five Why* method, central in Lean, can be useful [15] and analyzing the customer’s own value stream [21]. Besides the requirement to capture functional wants and needs, important non-functional wishes must be identified, including usability and what the customer are willing to pay [15].

Several seminal authors [15], [21] refer to the *Kano model* to categorize customer needs into must-be, performance and attractive features. When there are differences in customers and their environment the organization must identify general kinds of activities all customers are performing [15]. Identifying things that the customer specifically wants to avoid can also be useful input the requirements work and testing.

V4. To maximize the business profit-to-risk ratio, customer values and the implementation of these need to be prioritized, balancing different customer needs with the need for risk reduction.

Simple prioritization into *shall* or *should* requirements is not sufficient according to the seminal Lean authors. More systematic and dynamic schemes to prioritize between different customers and values should be considered. Examples of such methods are *Affinity Diagramming, Three-pile method*, self-rated importance questionnaires and Scrum-style product backlogs with user stories [15] . The

Poppendiecks [21] emphasizes the importance of resolving needs into *when* particular features are needed to maximize the overall business performance (profit). Middleton and Sutton adds that determining the right order must be a balance between technical risk reduction and providing sufficient hard features and emotional values for each release [15]. In this process it can also be useful to think ahead on how the customer will react, change behavior and wants after a specific feature has been deployed to them [15].

V5. To be able to develop a distinguishable and profitable product, the organization needs to analyze competitors' products' features and their actual usage.

The value resolution process should include analysis of competitors target customers, values and solution in order to create a distinguishable and needed product [17]. Visiting and studying customers who use competing product is recommended [15].

V6. To enable an effective development value stream, requirements need to be captured in a customer-centered format, easy to verify and possible to easily map to implementation and verification.

Customer values shall be captured in a customer-oriented language, using their own terminology, explaining the goals to be achieved and the reason why, avoiding talking about the solution space (how?). Ideally the specification of value should be in the language of the customer, easy to understand by both the customer and the developer, be unambiguous, has evident completeness and separate what from how [15]. From an ideal requirement specification it should also be possible to define a clear mapping to how and where to implement the feature and how to verify it in the end. To facilitate the latter a test-driven approach (i.e. writing requirements as tests or at least writing test cases before implementation, preferably automated) can be a step forward in line with Lean thinking [21]. For simple products and when a customer representative is present, *User Stories* may be sufficient to capture the values in a condensed and effective way, at least when there is no need for more formal specifications according to legal and/or maintenance requirements. For more complex systems expressing the behavior systematically in the language of the user, *use case diagrams and scenarios* are practices stated to be well in line with Lean thinking [5],[21]. Another method claimed to appropriate for lean software development is the *Software Cost Reduction (SCR)* method, at least for real-time systems [15].

Several of the seminal authors [5],[15],[21] stress the importance of using domain modeling to both model and understand the customer environments and to use that to create a software architecture that reflects this model as closely as possible, creating maintainable product with high user-perceived integrity [21].

E. Formulating a goal for the Value concept area

The result of the review of seminal sources view of the Value concept presented above the following overall goal was formulated for this area:

"The software development organization shall work hard to, and have a systematic way of, identifying and prioritizing customers and their needs, and also use effective means to let those permeate the development team and the design of the software."

with the motivation

"Strong focus on creating value for the customer and society is central to create a profitable company by fulfilling important needs of the customer and to be able to separate waste from value-adding activities."

Similar high level goals, recommended activities and practices have been formulated for the other concepts in order to create a comprehensive framework for lean software development.

VI. Discussion

The literature review identified only a few (30) primary papers about lean software development indicating that this is a relatively new research area. A limitation of this study was that not all conference papers were included in Discovery indexing service used. In this way there is a risk that some seminal paper referred to in the scientific literature could have been missed. Some useful literature could also have been missed due to the fact that no author of peer-reviewed papers has yet referred to it.

Although care has been taken to map key concepts and recommended activities based on careful encoding, there is always a room for misinterpretations. Nevertheless, the framework is it built up in a very extensible and traceable manner to easily accommodate corrections and extensions based on new insights and experiences. In line with Lean thinking, this framework serves merely as a basis for standardization, and is expected to continuously change as new insights are gained.

When comparing the goal and activity statements obtained in this framework one by one with currently known 'best practices' for software development expressed in for instance capability models such as Capability Maturity Model Integration [24] or just as 'common sense', Lean Software Development may not seem to provide much new insights to the software engineering area. However, it is the focus on improvement of the whole sociotechnical organizational system and the value streams within it, not individual process areas, which is the major important Lean concern [17]. In this way, the Lean Software Development framework developed has a potential to complement the view of other models. It is however important to stress that the purpose of the framework is guide researchers and practitioners to what lean software development actually is. Until evidence of use is provided it is not possible to claim that this will lead to actual improvements of some aspect of software development.

VII. Conclusions and Further work

This paper summarized the results from a literature survey identifying nine seminal sources and 20 key concepts for Lean software development. Based on those it was demonstrated how these results could be transformed to a comprehensive framework. Through the example Value, we demonstrate the potential that such framework has to guide researchers and practitioners in how to apply Lean thinking to software development. The natural next step is to develop this into a Lean assessment instrument to be validated and developed iteratively in pilot studies in various contexts.

The framework may also help to contrast Lean and agile software development. The terms Lean and agile are often used synonymously, but by doing this study we have noticed differences. These differences should be further investigated.

Acknowledgment

This research is supported by the Knowledge Foundation (KKS) through ITS-EASY, an Industrial Research School in Embedded Software and Systems, affiliated with Mälardalen University, Sweden.

References

[1] P. Abrahamsson, N. Oza and M. T. Siponen, “Agile software development methods: A comparative review”, 2010, in “Agile software development: Current research and future directions”, Springer, doi: 10.1007/978-3-642-12575-1

[2] D.J. Andersson, ”Kanban”, 2010, Blue Hole Press

[3] D.J. Andersson, “Lean software development”, Microsoft Development Network, 2012, http://msdn.microsoft.com/en-us/library/vstudio/hh533841.aspx, accessed April 10, 2013

[4] M. Aoyama, “Beyond software factories: concurrent-development process and an evolution of software process technology in Japan.”, Information and Software Technology, 1996, vol. 38, pp. 133–143

[5] J. Coplien and G. Bjornvig, “Lean architecture: for agile software development”, 2010, John Wiley & Sons Ltd

[6] C., Ebert, C., P. Abrahamsson and N. Oza, “Lean software development”. IEEE Software, 2012, vol. 29(5), pp. 22-25, doi: 10.1109/MS.2012.116

[7] T. Karvonen, P. Rodriguez, P. Kuvaja, K. Mikkonen, and M. Oivo, “Adapting the lean enterprise self-assessment tool for the software development domain”, Sep. *2012, 38th Euromicro Conference on Software Engineering and Advanced Applications*, pp. 266–273, doi: 10.1109/SEAA.2012.51

[8] P. Kettunen, “A tentative framework for lean software enterprise research and development”, Lean Enterprise Software and Systems (LESS2010), Lecture Notes in Business Information Processing, 2010, vol. 65, pp. 60-71, Springer Berlin, doi: 10.1007/978-3-642-16416-3_11

[9] B. Kitchenham, “Procedures for performing systematic reviews.”, 2004, Keele University Technical Report TR/SE-0401

[10] M. Lane, B. Fitzgerald and P. Agerfalk, “Identifying lean software development values”, European Conference on Information Systems (ECIS), 2012.

[11] J.K. Liker, “The Toyota way: 14 management principles from the world’s greatest manufacturer”, 2004, McGraw-Hill

[12] P. Middleton, “Lean software process”. Journal of Computer Information Systems, 2001. vol. 42, no. 1, p. 21

[13] P. Middleton, A. Flaxel and A. Cookson, “Lean software management case study: Timberline Inc.”, Extreme Programming And Agile Processes In Software Engineering, Proceedings, 2005, vol. 3556, pp. 1-9

[14] P. Middleton and D. Joyce. “Lean Software Management: BBC Worldwide Case Study”, IEEE Transactions on Engineering Management, 2012, vol. 59, no. 1, pp. 20-32, doi: 10.1109/TEM.2010.2081675

[15] P. Middleton and T. Sutton, “Lean software strategies: Proven techniques for managers and developers”, 2005, Productivity Press

[16] T. Morgan, "Lean manufacturing techniques applied to software development", 1998, Massachusetts Institute of Technology, MSc. Thesis

[17] J.M. Morgan and J.K. Liker, “The Toyota product development system”, 2006, Productivity Press.

[18] D. Nightingale and J. Mize, “Development of a lean enterprise transformation maturity model,” *Information, Knowledge, Systems Management*, vol. 3, pp. 15–30, 2002.

[19] T. Ohno, “Toyota production system: Beyond large scale production”, 1998, Productivity Press.

[20] K. Petersen, “Is lean agile and agile lean?: a comparison between two software development paradigms” in Modern Software Engineering Concepts and Practices: Advanced Approaches, IGI Global, 2010, pp. 19-46, doi: 10.4018/978-1-60960-215-4.ch002

[21] M. Poppendieck and T. Poppendieck, “Lean software development: An agile toolkit”, 2003, Addison-Wesley

[22] M. Poppendieck and T. Poppendieck, “Implementing lean software development: From concept to cash”, 2006, Addison-Wesley

[23] M. Poppendieck and T. Poppendieck, “Leading lean software development”, 2010, Addison-Wesley

[24] CMMI Product Team, “CMMI for development, version 1.3,” Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-2010-TR-033, 2010.

[25] A.C. Ward, “Lean product and process development”, 2007,The Lean Enterprise Institute Inc

[26] J.P. Womack and D.T. Jones, D. Roos, “The machine that changed the world”, 2007, 2nd ed., Simon & Schuster UK.

[27] J.P. Womack and D.T. Jones, ”Lean thinking: Banish waste and create wealth in your corporation”, 2003, Free Press, New York.