Lean Software Development – What Exactly Are We Talking About?

Oisín Cawley[1], Xiaofeng Wang[2], Ita Richardson[1],

[1]Lero-The Irish Software Engineering Research Centre, University of Limerick, Ireland.

[2]Free University of Bozen/Bolzano, ominikanerplatz 3 piazza Domenicani, I-39100 Bozen/Bolzano, Italy.

[1]{Oisin.Cawley, Ita.Richardson}@lero.ie, [2]Xiaofeng.Wang@unibz.it

Abstract. As the Software Engineering landscape continues to evolve and new paradigms are introduced, there can be a tendency for both industry and academia to enthusiastically embrace new approaches and march forward under whatever banner conventional wisdom has decided to adopt. One such banner is Lean Software Development, a paradigm that continues to see a growth in interest driven by the need for cost reductions within industry. The term lean attracts the attention of business, but precisely how it applies within software development is still being debated. In addition, its relationship to the better understood agile methodologies is also a topic for debate. Having been drawn into this research area ourselves, we present here a review of Lean Software Development and try to distil out for the reader some understanding of this somewhat undefined topic. We conclude with some thoughts on where this subject might go to from here.

Keywords: Software Engineering, Software Development, Lean, Agile.

1 Introduction

We are living in a period of history which is witnessing a drive for cost reductions and efficiencies across almost every business segment in almost every developed country. Traditionally businesses have been able to find efficiencies and cost reductions through automation and organisational restructuring of the more routine and repeatable processes. However, many businesses are now looking at higher skilled functions, such as Information Technology, including Software Development, and asking how such functions might play their part in finding more cost effective ways to operate.

The manufacturing world in particular benefited greatly from the Japanese concepts, and indeed philosophy, which grew out of the Toyota Production System of the 1940s. With evidence of huge productivity gains, the declarative term of "Lean Manufacturing" wasn't coined until 1990 [1]. Driven by a need to become and remain competitive in business, the term 'Lean' is now being applied to functions outside of the manufacturing context.

Only recently has the term Lean Software Development started to become a household name within the Software Engineering vernacular. It is quite common for businesses to have a lean ethos and run lean campaigns to improve processes and reduce costs [2]. However, we should be careful not to assume that by prefixing any activity with the word 'lean', it should automatically generate more cost effective processes and instil employees with a new sense of direction. This, naturally, does not follow and we need to be clear what we actually mean when we refer to something as being lean. We therefore suggest that although Lean Software Development has piqued the interest of many within industry and academia, it lacks a formalised description which fuels the confusion which has grown up around it [3], [4], [5]. This paper aims to begin a process of definition by presenting a view of the origins and subsequent contributions that have been made to LSD as we know it to date.

To this end the remainder of the paper is organized as follows: Section 2 is a retrospective on the origin and fundamentals of Lean. Section 3 is focused on the application of Lean within a software development context. Specific Lean principles pertinent to software development are reviewed. It is followed by a discussion of the relation of Lean Software Development and agile methods, which is a frequently debated topic where Lean practices are concerned. Then our interpretation of Lean Software Development in the light of the agile and lean debate is presented in Section 5. The paper ends with a conclusion section in which the contributions of the paper are highlighted and future research directions proposed.

2 Lean Fundamentals

2.1 The Origin of Lean

The origins and taxonomy of Lean are generally attributed to the now well studied Toyota Production System (TPS) developed in Japan around the 1940s [6]. The TPS is a form of what Womack et al. Coined 'Lean Manufacturing' [1]. It is an approach to manufacturing which revolutionised the automobile industry and allowed the Japanese become one of the dominant forces in the industry worldwide. Interestingly, some of the concepts were actually used prior to this by the Ford Automobile company [7]. The core principles underlying lean manufacturing, are not confined to manufacturing processes either but have been shown to be applicable in many other disciplines too [8].

One of the cornerstones of Toyota's success was their ability to produce high quality cars at the end of the production line with little or no need for rework. This was achieved by the early detection of defects and the immediate focus on eliminating the cause of the defect so that it would never happen again. This basic principle is of course transferable to many other domains/situations. In an interview, sportsman

Brendan Cummins, one of the leading hurling[1] goalkeepers in Ireland, when asked what advice he would give to young players said:

"*Go home and practice until you eliminate the mistakes that led to the defeat. It's the only way to get better.*"

In a Software engineering context the parallel can be drawn with finding and eliminating defects in your code as quickly as possible and learning not to make those same mistakes again. [9] suggests that each bug found by a developer should lead to the following two questions:

– How could I have automatically detected this bug?
– How could I have prevented this bug?

However, lean is not only concerned with defect identification and eradication but equally concentrates on the surrounding processes, and getting work to 'flow' smoothly through the entire process. While this is easier to visualise in a manufacturing setting where product can be seen to flow down the assembly line [10], this can also be applied in a product development environment [11].

2.2 Lean Concepts

Lean is more a way of thinking about, or a mental approach taken, to a particular process or set of processes. Lean is about achieving more with less [12], or producing in one-third the time, at one-third the cost, and with one-third the defect rate [13]. The primary focus and guiding principle is the identification and elimination of process waste in order to focus on creating customer value [14], [15], [10], [16]. An important aspect is that value is determined by the customer not the producer, and so any cost incurred in the process needs to be in support of activities which add value to the customer (Fig. 1).

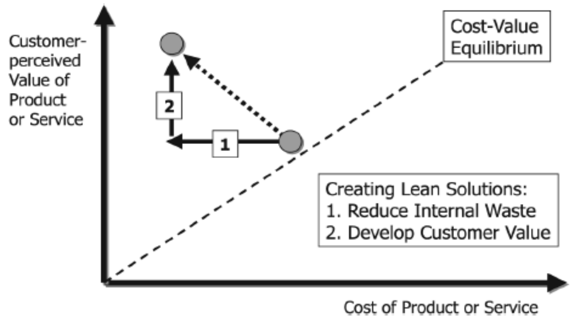
Customer-perceived Value of Product or Service
Cost-Value Equilibrium
2
1
Creating Lean Solutions:
1. Reduce Internal Waste
2. Develop Customer Value
Cost of Product or Service

Fig. 1. Relationship between Value, Cost and Waste (Hines et al., 2004)

There are five main modern lean concepts [8]:

1. Value: It is defined by the customer and it is paramount to have a clear understanding of what that is;

[1] http://www.gaa.ie/about-the-gaa/our-games/hurling/

2. Value Stream: A map that identifies every step in the process and categorises each step in terms of the value it adds;
3. Flow: It is important that the production process flows continuously;
4. Pull: Customer orders pull product, ensuring nothing is built before it is needed;
5. Perfection: You strive for perfection in your process by continuously identifying and removing waste.

However, a 6th concept is a recurring and important one:

6. Respect for People [17],[18] [19].

This is highlighted by [17] where they describe the key facets of the Toyota Production System. This last principle is very significant and a key component of Toyota's second basic concept: "To make full use of workers' capabilities". In addition, [20] see it as a key component in building what they call a 'lean enterprise', and it has also been incorporated into the International Council on Systems Engineering's model for Lean Enablers for Systems Engineering [18]:

> "*The "People" principle promotes the best human relations at work based on respect for people: trust, honesty, respect, empowerment, teamwork, stability, motivation, drive for excellence, and healthy hiring and promotion policies*" [19].

Petersen and Wohlin summarise lean manufacturing under two important headings, the removal of waste and continuous flow [21]. It is instructive to take a brief look at both of these.

The Removal Of Waste

Lean thinking classifies work into 3 categories [8]:

- Value-adding activities
- Required non-value adding activities
- Non-value adding activities

By mapping out a work process using a value stream map, process steps which do not contribute to creating value can be identified, thus allowing for a concentrated effort on reducing or eliminating these steps. The concept of waste can therefore be quite broad but Taiichi Ohno – the founder of the Toyota production process - defined 7 types of waste [14], and an eighth has been added by [6]. These are:

- The waste of over-production
- The waste of time on hand (waiting)
- The waste of transportation
- The waste of over-processing or incorrect processing
- The waste of stock on hand (excess inventory)
- The waste of movement
- The waste of making defective products
- The waste of unused employee creativity

It is the identification of these different types of waste which make a lean approach so powerful, since it does not suggest that any particular part of the process should be targeted, but rather waste in any form and in any place should be sought out.

Continuous Flow

Within a manufacturing context the control of excess inventory is a constant focus for a number of reasons, such as storage and transportation costs, longer lead times, and the masking of quality problems [22], [6]. The reduction of inventory is given special attention in a lean context [1, 8]. The Toyota company's approach was in fact not to manage excess inventory but to seek to eliminate it [6]. Especially within the wider supply chain this becomes an increasingly difficult problem to manage [20].

Lean aims to achieve a smooth and continuous flow of inventory through the production process [23]. By linking together disjointed operations, thereby increasing teamwork and feedback, quality problems are identified earlier [6]. A consequence of this linking up is that it becomes possible to better schedule the generation and delivery of inventory to downstream processes in a just-in-time fashion [14], [6].

One technique used to achieve this smooth flow is termed Kanban. A Kanban system is:

"*A production control system for just-in-time production and making full use of workers' capabilities.*" [17].

The core objective of the Kanban system is to minimise the amount of Work-In-Progress (WIP), or inventory. This is achieved by inventory being "pulled" through the system as it is needed, as opposed to "pushing" it through. Only when a downstream process is ready and needs to do some more work does it pull inventory from an upstream process. The signalling between upstream and downstream processes is typically done via some sort of coloured card (the Kanban) which physically travels between processes [24]. The aim is to keep the process flowing at an even but continuous rate. This is achieved by controlling the number of Kanban cards which are in circulation within the process. Reducing the number of cards reduces the amount of inventory in circulation, and is therefore used to control WIP. The other important aspect of Kanban is that it includes a visual display of the entire process so everyone has visibility and can see if and where issues are starting to appear.

3 'Leaning' Software Development

An important question to ask is how do lean manufacturing practices - typically repeating identical tasks and producing the same product as output - relate to product development activities which always produce something different? Don Reinertsen, an expert in product development management, has applied lean manufacturing practices to product development [25] and details some significant differences he sees between the two [11]. Consequently he suggests that blindly trying to drive up efficiencies and drive down variability without considering the economic consequences is fundamentally wrong.

Looking specifically at lean from a Software Development perspective, Raman attempted to "*... see whether the basic Lean principles ... can be applied to software development*" [26]. He concluded, that with practices such as rapid prototyping, quality function deployment, continuous integration, object oriented and component-based development:

"*The question whether Lean Software Development is Feasible can easily be answered with "yes"* " [26].

Similarly, [23] suggest that:

"*Results of lean product development are more interesting for software engineering than the pure manufacturing part as the success of software development highly depends on an integrative view*".

They conclude that lean principles may be beneficial in a software development context but that:

"*Further evaluation of lean principles is needed to understand how they affect the performance of the software process*" [21].

3.1 Waste in Software Development

A common core focus of all lean software development proponents is delivering value by identifying and eliminating waste, but the interpretation of waste and how to address it can vary. For example, while [27] and [28] identify specific wastes which should be addressed immediately, [11] suggests that only when the proposed waste has been converted into economic terms does it become useful in deciding whether it is waste and what to do about it. Additionally [29] talks about addressing what he calls the waste of negative iterations. Referring to iterative design, he says a negative iteration is one which could be eliminated without any loss in value. Table 1 shows an interpretation of waste based on [6, 27, 28].

Table 1 Waste in Lean Software Development

Lean Manufacturing	Lean Software Development
Over-Production	Extra Features/Code
Time On Hand (waiting)	Delays
Transportation	Task Switching
Over-Processing or Incorrect Processing	Extra Processes
Stock On Hand (excess inventory)	Partially Done Work
Movement	Movement
Making Defective Products	Defects
Unused Employee Creativity	Unused Employee Creativity

3.2 Lean Software Development Principles

The core intent of lean can be summarised as follows:

"*All we are doing is looking at the timeline from the moment a customer gives us an order to the point when we collect the cash. And we are reducing that timeline by removing the nonvalue-added wastes*" [14].

By applying this approach through the application of lean principles [6] within the context of software development, we can see how many of the modern software development techniques support them. The contemporary understanding of lean software development is largely driven by practitioners writings [27], [30], [28] and [31], however the broad nature of lean means that lean software development has much in common with related domains such as lean product development [11] and lean systems engineering [19]. In [32] Robert Charette developed 12 principles of Lean Development:

1. Satisfying the customer is the highest priority
2. Always provide the best value for money
3. Success depends on active customer participation
4. Every LD project is a team effort
5. Everything is changeable
6. Domain, not point, solutions
7. Complete, don't construct
8. An 80 percent solution today, instead of 100 percent solution tomorrow
9. Minimalism is essential
10. Needs determine technology
11. Product growth is feature growth, not size growth
12. Never push LD beyond its limits

Table 2 lists some of the sets of lean principles proposed in the literature more specific to a software development context and well know within the agile community. While these sets differ slightly in the lean principles they advocate, they all share some core lean concepts such as waste elimination. Poppendieck and Poppendieck tell us that waste is anything which does not add value, Anderson tells us it is important to be able to visualise such process waste within the workflow, and Reinertsen agrees but suggests that we must weigh up the economic cost of eliminating a particular type of waste. Lean principles and practices continue to evolve but a synthesis of what we can call lean practices has been started by [5].

Table 2 Lean Principles Relevant to Software Development

Lean Software Development Principles (Poppendieck and Poppendieck, 2003)	The Principles of Product Development Flow (Reinertsen, 2009)	The Kanban Principles (Anderson, 2010)
• Eliminate waste • Build quality in • Create knowledge • Defer commitment • Deliver fast • Respect people • Optimise the whole	• Use an economic view • Manage queues • Exploit variability • Reduce batch size • Apply WIP (Work in Progress) constraints • Control flow under uncertainty • Use fast feedback • Decentralise control	• Visualize the workflow • Limit WIP • Manage Flow • Make Process Policies Explicit • Improve Collaboratively (using models & the scientific method)

4 Lean or Agile

We look now specifically at the confusion which has emerged about the differences/similarities between Lean Software Development and Agile Software Development. According to Robert Charette - originator of "Lean Development" [13] – LSD is a key component in building a change tolerant business [32]. The key difference he sees between lean and agile is that agile is a bottom up approach while lean is a top down approach.

The toolkit of lean SD practices developed by [27] contains many practices already well established within the agile community. This has both helped the agile community to embrace lean, but also added to the confusion as to what exactly the difference between lean and agile is. Consequently the boundary between lean SD and agile SD is something that is currently being debated [33], [34], [5]. [35] performed a comparison between lean concepts and generic agile practices, with specific focus on the Scrum methodology, drawing parallels between them, but concluding that lean thinking can help a company analyse its software development process irrespective of the development methodology in use. Petersen [4] performed a more detailed comparison between two development paradigms (lean and agile) and concludes that: "*(1) Agile and lean agree on the goals they want to achieve; (2) Lean is agile in the sense that the principles of lean reflect the principles of agile, while lean is unique in stressing the end-to-end perspective more; (3) Lean has adopted many practices known in the agile context, while stressing the importance of using practices that are related to the end-to end flow*". He also concludes that "*agile uses practices that are not found in lean*" but does not state the reciprocal i.e. that lean utilises practices not found in agile, as reported by [5].

Waste elimination, within a lean manufacturing perspective, means removing steps in your process that do not directly contribute to adding customer value. From a software development perspective, one interpretation of waste is the identification and elimination of defects in the software, and so effort is expended to ensure defects are found and removed as quickly as possible. Another interpretation is writing too much code such as developing features which were not requested (over production), something which [28] suggests that Behaviour Driven Development (BDD)[2] can help address. Similarly, if defects are allowed to propagate through the system and are identified late in the project, then a considerably larger effort is required to remove them than if they were caught early. This wasted effort may be eliminated by following a test-first approach such as the agile practice of test driven development (TDD) [36]. TDD proposes that even before any code is written, a test case is written for the code. Only when the test passes does the developer move on to the next coding task. This significantly reduces the possibility of defects going undetected.

[2] BDD aims to help focus development on the delivery of prioritised, verifiable business value by providing a common vocabulary that spans the divide between Business and Technology.

4.1 Agile and Lean Practices

Many agile practices have been mapped to form a toolset of lean SD practices by [37], [30] and a guide on implementation was written by [28]. Table 3 shows a cross section of these practices.

Table 3 Agile Practices within Lean Software Development

- Run software tests as soon as code is written
- Write code instead of more documentation or detailed planning
- Propose user interfaces and get feedback instead of more detailed requirements
- Test the top 3 tools instead of trying to pick the right one first time
- Develop in short iterations
- Features that are too big for 1 iteration need to be broken down
- The highest priority feature should be developed first
- High risk items should be addressed earlier rather than later
- Make progress visible
- Automate software builds and build tests
- Spanning (a portion of the application is built to completion which spans all modules and dependencies)
- Set-based development (multiple initial options developed in parallel)
- Object oriented design
- Component-based development
- Avoid extra features
- Fast feedback loops
- Empower the team through self-organisation
- Pull systems make the work self-directing
- Make work packages small to assist with work flow

Some lean practices however do not have any specific reference within agile practices. For example, in 1988, Harold Thimbleby published an article entitled "Delaying Commitment" [38], in which he advocates delaying software decisions as long as possible. Although this could be termed lazy evaluation, he argued that there is much benefit in keeping your options open and as flexible as possible to be able to quickly cater for any changes that crop up further along in the project. This later became one of the lean SD practices as described by [37]. Although the agile manifesto similarly values "Responding to change", the specific practice of delaying commitment has not been seen within the corresponding methodologies. Another example is the practice of Poka-Yoke [39], where processes are defined that make it impossible for mistakes to happen, and where that is not achievable, to design the processes in such a way that defects are easily detected and corrected. For clarification, Table 4 shows a compilation of practices which have been categorised as having only lean origins [5].

Table 4 Lean (only) Software Development Practices

- Jidoka (build quality in) [6]
- Poka-Yoke (Defect detection and prevention) [9]
- Kano analysis to link voice of the customer to requirements [40], [41]
- Quality Function Deployment [41]
- Value Stream Mapping [8], [6], [27], [42]
- Transparency [8]
- Make project status highly visible
- Visualise all work items
- Limit WIP (Work in Process), [43], [31]
- Workload levelling "Heijunka" [6], [40]
- Addressing bottlenecks [6], [44, 45], [40], [27]
- Deferring decision making [38], [27]
- Moving variability downstream [27]
- Reducing slack [22]
- Measure and manage [46]
- Employ Queuing Theory [25], [44, 45] but measure the right things [11]
- Employing Pull systems
- Kanban, Limitied WIP, CONWIP (Sugimori et al., 1977; Bradley, 2007; Kniberg and Skarin, 2010), [31]
- Batch control processing [47], [11]
- Value stream Kaikaku [8]
- Relentless reflection "Hansei" and continuous improvement "Kaizen" [6], [28], [48]
- Avoiding too much local optimisation [27]
- Hiding individual performance [27]
- Root cause analysis
- The 5 whys? [16]
- Promoting a 'safe' environment. Instil a "stop the line" mentality [27], [16]
- Developing appropriate incentives/rewards [49]
- Pragmatic governance (enable first, manage/control second) [49]

A more scientific approach is taken by [11] and [31], where focus is placed on measurements derived from areas such as queuing theory, variability and transaction cost analysis and using those measurements to shape and adjust the development processes.

4.2 Kanban

The most recent addition to the lean/agile methodology affray has been that of Kanban software development, and it takes its name from the Kanban scheduling system developed at Toyota [50], [51], [31] and explained in section 1.2 above. The Kanban approach is applied within the software development domain by means of a

Kanban board. Fig. 2 shows an example of a typical software Kanban board, reconstructed from photos of boards used at Yahoo [52].

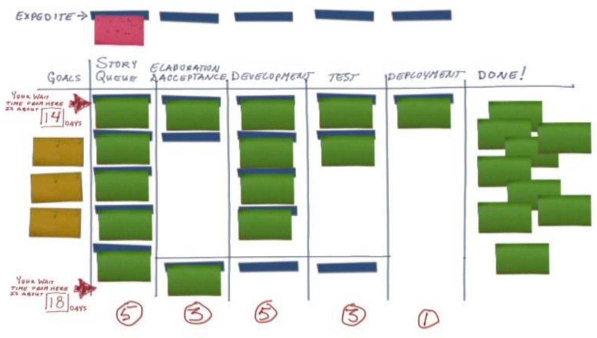

Fig. 2. A Kanban Board for Software Development [52]

Development tasks are written on sticky notes and move (or flow) [51] from left to right across the board. High level objectives are listed to the left, and the next column, similar to agile, is a queue of prioritised stories which progress through the various development stages as identified by the column headings.

Although the process shares similarities to agile approaches, such as a prioritised feature list, Kanban's primary concern is to limit work in progress (WIP). However, there is a second significant difference between it and agile methodologies. The concept of a time-boxed (fixed duration) iteration is no longer used. Instead, the Kanban board is used to set clearly visible limits to the number of tasks allowed in any of the columns. These limits are identified as circled numbers in Fig. 2. There is no fixed number for each WIP limit but by measuring the lead-time of individual tasks, the WIP limits and process itself can be optimised [53]. While an agile approach (agile kanban) would start a clean board for each iteration [50], kanban software development pursues the concept of continuous flow or what [50] refers to as "Sustaining Kanban".

The adoption of kanban SD can be difficult for those already familiar with an agile approach due to the lack of iterations. However, the inability of agile methodologies to easily scale up, is something that is pushing teams to look to alternatives. One emerging approach is to combine agile and kanban into some form of hybrid methodology. An example of such an approach is the methodology termed Scrumban [54], where the more structured and tightly coupled activities as defined by the SCRUM methodology are merged with the pull based workflow approach of kanban.

5 Lean Software Development – An Integrated View

Drawing upon the literature reviewed in the previous sections, we propose that LSD can be viewed as the merging of lean concepts with modern software development practices, such as those found in agile, and other general best practices for software development (Fig. 3). General best practices are those which are common to any disciplined SD approach, such as structured code, in-code commenting, object oriented, and source code control.

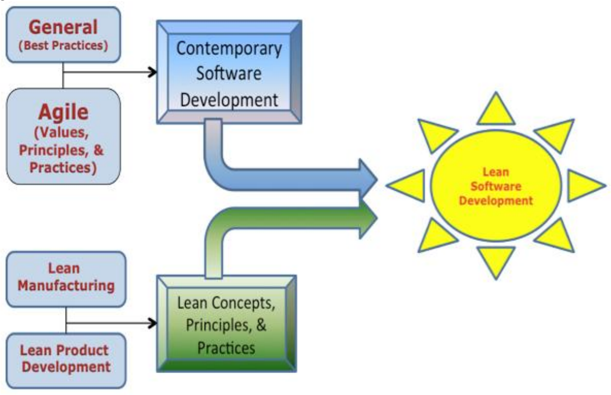
General
(Best Practices)
Agile
(Values, Principles, & Practices)
Contemporary Software Development
Lean Software Development
Lean Manufacturing
Lean Product Development
Lean Concepts, Principles, & Practices

Fig. 3. Foundations of Lean Software Development

To position LSD in a broader context, since lean is a philosophy which can be applied at an organisational level, and agile is typically focused at a more practical level, agile methods can be seen as supportive practices of a lean software development philosophy as depicted in (Fig. 4).

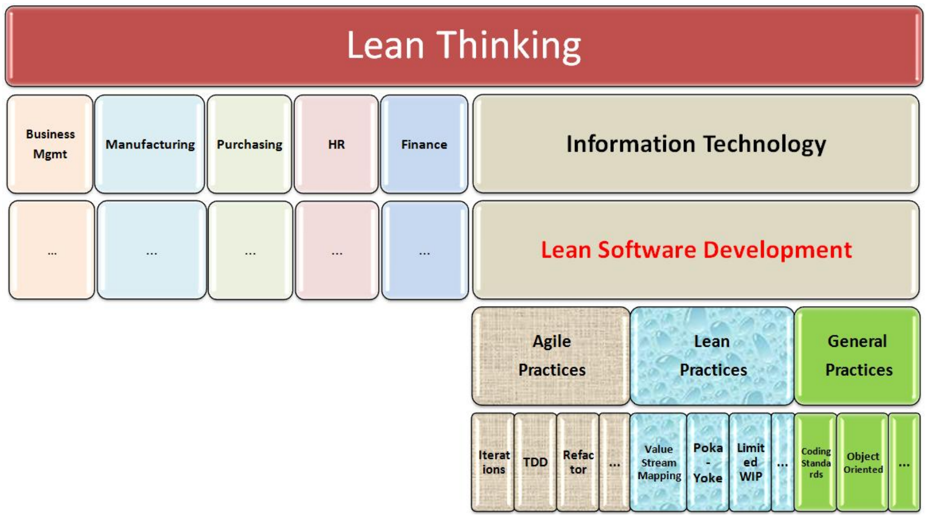

Fig. 4. The positioning of lean and agile SD practices within the organisation

In Fig. 4 we have positioned LSD within a wider umbrella of ‘Lean Thinking’ which spans the entire organisation. This is important to realise, since getting the entire value-stream working together towards a common goal in a continuously flowing fashion is the real aim of a ‘Lean Enterprise’ [8]. Trying to optimise one piece of the process, for example, writing the software code, may lead to sub-optimising the wider process [45]. [3] for example, when examining their internal processes leading to software delivery, found that the actual coding accounted for only 10% of the time it took to deliver a project into production. This indicates that 90% of the time is devoted to activities outside the actual coding and overflowing into other groups, departments, and or even companies.

6 Where to Now?

We have shown that there is a wealth of information available on the topic of Lean Software Development, and presented an integrative perspective of the topic. We hope the reader has an appreciation of the potential that we believe LSD holds for industry as evidenced by the volumes of contributions already within the domain. However, similar to what has happened with Agile SD adoption, once again the Software Engineering community finds itself with something which has evolved from industry and conventional wisdom. The academic world is struggling to pin down the phenomenon with any level of empirical foundation: "*Further evaluation of lean principles is needed to understand how they affect the performance of the software process*" [21].

Due to the subjective nature and qualitative aspects of being 'lean', we suggest that as a research community we need to try and bring some form of unified understanding of this paradigm or methodology to the field. Its nascent nature and the continued use of the term LSD in different contexts, makes it difficult to even begin such a process. We therefore refrain from proposing a possible framework to adopt or adapt in order to achieve this, but rather call on the Software Engineering community to begin a series of workshops which would build towards such an output.

Acknowledgments. This research is supported by the Science Foundation Ireland (SFI) Stokes Lectureship Programme, grant number 07/SK/I1299, the SFI Principal Investigator Programme, grant number 08/IN.1/I2030 (the funding of this project was awarded by Science Foundation Ireland under a co-funding initiative by the Irish Government and European Regional Development Fund), and supported in part by Lero - the Irish Software Engineering Research Centre (http://www.lero.ie) grant 10/CE/I1855.

References.

1. Womack, J.P., D.T. Jones, and D. Roos, *The Machine That Changed The World: How lean production revolutionized the global car wars*1990: Simon & Schuster Ltd. 352.
2. Cawley, O., I. Richardson, and X. Wang, *Medical Device Software Development - A Perspective from a Lean Manufacturing Plant*, in *11th International Conference on Software Process Improvement and Capability*

Determination, R. O'Connor, et al., Editors. 2011, Springer: Dublin, Ireland. p. 84-96.
3. Parnell-Klabo, E., *Introducing Lean Principles with Agile Practices at a Fortune 500 Company*, in *Proceedings of the conference on AGILE 2006*2006, IEEE Computer Society. p. 232-242.
4. Petersen, K., *Is Lean Agile and Agile Lean?: A Comparison between Two Software Development Paradigms*, in *Modern Software Engineering Concepts and Practices: Advanced Approaches*2011, IGI Global. p. 19-46.
5. Wang, X., 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, 2012. **85**(6): p. 1287-1299.
6. Liker, J., *The Toyota Way* 2003: McGraw-Hill.
7. LeanManufacturingConcepts. *History of lean manufacturing*. 2008 8-Aug-2011]; Available from: http://www.leanmanufacturingconcepts.com/HistoryOfLeanManufacturing.htm.
8. Womack, J.P. and D.T. Jones, *Lean Thinking : Banish Waste and Create Wealth in Your Corporation* 1996: Simon & Schuster.
9. Robinson, H., *Using Poka-Yoke Techniques for Early Defect Detection* in *Sixth International Conference on Software Testing Analysis and Review* 1997.
10. Cumbo, D., D.E. Kline, and M.S. Bumgardner, *Benchmarking performance measurement and lean manufacturing in the rough mill.* Forest Products, 2006. **56**(6): p. 25-30.
11. Reinertsen, D.G., *The Principles of Product Development Flow: Second Generation Lean Product Development* 2009: Celeritas Publishing.
12. Christopher, M. and D.R. Towill, *Supply chain migration from lean and functional to agile and customised.* Supply Chain Management, 2000. **5**(4): p. 206-213.
13. Charette, R., *Challenging the Fundamental Notions of Software Development*, 2007.
14. Ohno, T., *Toyota Production System: Beyond Large-Scale Production* 1988: Productivity Press. 152.
15. Hines, P., M. Holweg, and N. Rich, *Learning to evolve: A review of contemporary lean thinking.* International Journal of Operations & Production Management, 2004. **24**(10): p. 994-1011.
16. Womack, J.P., D.T. Jones, and D. Roos, *The Machine That Changed The World: How lean production revolutionized the global car wars*2007: Simon & Schuster Ltd. 352.
17. Sugimori, Y., et al., *Toyota production system and Kanban system Materialization of just-in-time and respect-for-human system.* International Journal of Production Research, 1977. **15**(6): p. 553 - 564.
18. INCOSE. *International Council on Systems Engineering*. 1990 8-Aug-2011]; Available from: www.incose.org.

19. Oppenheim, B.W., E.M. Murman, and D.A. Secor, *Lean Enablers for Systems Engineering.* Systems Engineering, 2011. **14**(1): p. 29-55.
20. Womack, J.P. and D.T. Jones, *From Lean Production to the Lean Enterprise. (cover story).* Harvard Business Review, 1994. **72**(2): p. 93-103.
21. Petersen, K. and C. Wohlin, *Measuring the flow in lean software development.* Software: Practice and Experience, 2011. **41**(9): p. 975-996.
22. Middleton, P., *Lean Software Development: Two Case Studies.* Software Quality Journal, 2001. **9**(4): p. 241-252.
23. Petersen, K. and C. Wohlin, *Software process improvement through the Lean Measurement (SPI-LEAM) method.* Journal of Systems and Software, 2010. **83**(7): p. 1275-1287.
24. Manufacturing, I.f. *Kanbans*. 2011 8-Aug-2011]; Available from: http://www.ifm.eng.cam.ac.uk/dstools/process/kanban.html.
25. Reinertsen, D., *Managing the Design Factory : The Product Developer's Toolkit* 1997: The Free Press.
26. Raman, S. *Lean software development: is it feasible?* in *Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE.* 1998.
27. Poppendieck, M. and T. Poppendieck, *Lean Software Development: An Agile Toolkit* 2003: Addison-Wesley Professional
28. Hibbs, C., S.C. Jewett, and M. Sullivan, *The Art of Lean Software Development*2009: O'Reilly Media. 128.
29. Ballard, G., *Positive vs negative iteration in design*, in *The 8th Conference of the International Group for Lean Construction*2000: Brighton, U.K.
30. Poppendieck, M. and T. Poppendieck, *Implementing Lean Software Development From Concept to Cash*2006: Addison-Wesley Professional
31. Anderson, D.J., *Kanban*2010: Blue Hole Press.
32. Highsmith, J., *Agile Software Development Ecosystems*2002: Addison Wesley.
33. Fowler, M. *AgileVersusLean*. 2008 11-Aug-2011]; Available from: http://martinfowler.com/bliki/AgileVersusLean.html.
34. Wang, X., *The Combination of Agile and Lean in Software Development: An Experience Report Analysis*, in *Agile 2011*2011: Salt Lake City, Utah, USA.
35. Consulting, O. *Benefits of Lean and Agile Compared*. 2007 [cited 2010 12/1/2010]; Available from: http://www.oakleigh.co.uk/page/3341/White-Papers/Whitepaper-Articles/Benefits-of-Lean-and-Agile-Compared.
36. Beck, K., *Test Driven Development: By Example* 2002: Addison-Wesley Professional. 240.
37. Poppendieck, M. and T. Poppendieck, *Lean Software Development: An Agile Toolkit.* Agile Software Development, ed. A. Cockburn and J. Highsmith2003: Addison-Wesley Professional.
38. Thimbleby, H., *Delaying commitment [programming strategy].* Software, IEEE, 1988. **5**(3): p. 78-86.
39. Grout, J.R. and B.T. Downs *A Brief Tutorial on Mistake-proofing, Poka-Yoke, and ZQC*. 2012.

40. Middleton, P., A. Flaxel, and A. Cookson, *Lean Software Management Case Study: Timberline Inc*, in *Extreme Programming and Agile Processes in Software Engineering*2005. p. 1-9.
41. Raffo, D., et al. *Integrating Lean principles with value based software engineering*. in *Technology Management for Global Economic Growth (PICMET), 2010 Proceedings of PICMET '10:*. 2010.
42. Mujtaba, S., R. Feldt, and K. Petersen. *Waste and Lead Time Reduction in a Software Product Customization Process with Value Stream Maps*. in *Software Engineering Conference (ASWEC), 2010 21st Australian*. 2010.
43. Ladas, C., *Scrumban*. Lean Software Engineering-Essays on the Continuous Delivery of High Quality Information Systems. Vol. 2011. 2009: Modus Cooperandi Press
44. Goldratt, E.M., *The Goal: A Process of Ongoing Improvement* 1992: North River Press.
45. Goldratt, E.M., *Critical Chain*1997: Aldershot : Gower. 248.
46. Anderson, D.J. and R. Garber *A Kanban System for Sustaining Engineering on Software Systems*. 2007.
47. Bradley, R. *Push to Pull: How Lean Concepts Improve a Data Migration*. in *AGILE 2007*. 2007.
48. Joyce, M. and B. Schechter, *THE LEAN ENTERPRISE—A MANAGEMENT PHILOSOPHY AT LOCKHEED MARTIN.* Defense Advanced Research Journal, 2004(August-November 2004).
49. Ambler, S.W. and P. Kroll. *Best practices for lean development governance*. 2007 [cited 2010 22nd January]; Available from: http://www.ibm.com/developerworks/rational/library/jun07/kroll/.
50. Hiranabe, K. *Kanban Applied to Software Development: from Agile to Lean*. 2008 [cited 2010 12/1/2010]; Available from: http://www.infoq.com/articles/hiranabe-lean-agile-kanban.
51. Birkeland, J.O. *From a timebox tangle to a more fexible fow*. in *Xp 2010-11th International Conference on Agile Software Development*. 2010. Trondheim, Norway: Springer Lecture Notes in Business Information Processing (LNBIP).
52. Patton, J. *Kanban Development Oversimplified*. 2009.
53. Kniberg, H. and M. Skarin, *Kanban and Scrum-Making the most of both*, 2010, InfoQ.
54. Ladas, C. *Scrumban*. Lean Software Engineering-Essays on the Continuous Delivery of High Quality Information Systems 2008 [cited 2011 28/9/2011]; Available from: www.leansoftwareengineering.com/ksse/scrum-ban/.