8 Lean Manufacturing, Lean enterprise and Lean Production

Lean software development

Lean software development (LSD) is a translation of lean manufacturing and lean IT principles and practices to the software development domain. Adapted from the Toyota Production System, a pro-lean subculture is emerging from within the Agile community.

Origin

The term lean software development originated in a book by the same name, written by Mary Poppendieck and Tom Poppendieck.The book presents the traditional lean principles in a modified form, as well as a set of 22 tools and compares the tools to agile practices. The Poppendiecks' involvement in the Agile software development community, including talks at several Agile conferences has resulted in such concepts being more widely accepted within the Agile community.

Lean principles

Lean development can be summarized by seven principles, very close in concept to lean manufacturing principles:

1.Eliminate waste

2.Amplify learning

3.Decide as late as possible

4.Deliver as fast as possible

5.Empower the team

6.Build integrity in

7.See the whole

Eliminate waste

Lean philosophy regards everything not adding value to the customer as waste (muda). Such waste may include:

unnecessary code and functionality

delay in the software development process

unclear requirements

insufficient testing (leading to avoidable process repetition)

bureaucracy

slow internal communication

In order to eliminate waste, one should be able to recognize it. If some activity could be bypassed or the result could be achieved without it, it is waste. Partially done coding eventually abandoned during the development process is waste. Extra processes and features not often used by customers are waste. Waiting for other activities, teams, processes is waste. Defects and lower quality are waste. Managerial overhead not producing real value is waste.

A value stream mapping technique is used to identify waste. The second step is to point out sources of waste and to eliminate them. Waste-removal should take place iteratively until even essential-seeming processes and procedures are liquidated.

Amplify learning

Software development is a continuous learning process with the additional challenge of development teams and end product sizes. The best approach for improving a software development environment is to amplify learning. The accumulation of defects should be prevented by running tests as soon as the code is written. Instead of adding more documentation or detailed planning, different ideas could be tried by writing code and building. The process of user requirements gathering could be simplified by presenting screens to the end-users and getting their input.

The learning process is sped up by usage of short iteration cycles – each one coupled with refactoring and integration testing. Increasing feedback via short feedback sessions with customers helps when determining the current phase of development and adjusting efforts for future improvements. During those short sessions both customer representatives and the development team learn more about the domain problem and figure out possible solutions for further development. Thus the customers better understand their needs, based on the existing result of development efforts, and the developers learn how to better satisfy those needs.

Another idea in the communication and learning process with a customer is set-based development – this concentrates on communicating the constraints of the future solution and not the possible solutions, thus promoting the birth of the solution via dialogue with the customer.

Decide as late as possible

As software development is always associated with some uncertainty, better results should be achieved with an options-based approach, delaying decisions as much as possible until they can be made based on facts and not on uncertain assumptions and predictions. The more complex a system is, the more capacity for change should be built into it, thus enabling the delay of important and crucial commitments. The iterative approach promotes this principle – the ability to adapt to changes and correct mistakes, which might be very costly if discovered after the release of the system.

An agile software development approach can move the building of options earlier for customers, thus delaying certain crucial decisions until customers have realized their needs better. This also allows later adaptation to changes and the prevention of costly earlier technology-bounded decisions. This does not mean that no planning should be involved – on the contrary, planning activities should be concentrated on the different options and adapting to the current situation, as well as clarifying confusing situations by establishing patterns for rapid action. Evaluating different options is effective as soon as it is realized that they are not free, but provide the needed flexibility for late decision making.

Deliver as fast as possible

In the era of rapid technology evolution, it is not the biggest that survives, but the fastest. The sooner the end product is delivered without major defects, the sooner feedback can be received, and incorporated into the next iteration. The shorter the iterations, the better the learning and communication within the team. With speed, decisions can be delayed. Speed assures the fulfilling of the customer's present needs and not what they required yesterday. This gives them the opportunity to delay making up their minds about what they really require until they gain better knowledge. Customers value rapid delivery of a quality product.

The just-in-time production ideology could be applied to software development, recognizing its specific requirements and environment. This is achieved by presenting the needed result and letting the team organize itself and divide the tasks for accomplishing the needed result for a specific iteration. At the beginning, the customer provides the needed input. This could be simply presented in small

cards or stories – the developers estimate the time needed for the implementation of each card. Thus the work organization changes into self-pulling system – each morning during a stand-up meeting, each member of the team reviews what has been done yesterday, what is to be done today and tomorrow, and prompts for any inputs needed from colleagues or the customer. This requires transparency of the process, which is also beneficial for team communication. Another key idea in Toyota's Product Development System is set-based design. If a new brake system is needed for a car, for example, three teams may design solutions to the same problem. Each team learns about the problem space and designs a potential solution. As a solution is deemed unreasonable, it is cut. At the end of a period, the surviving designs are compared and one is chosen, perhaps with some modifications based on learning from the others - a great example of deferring commitment until the last possible moment. Software decisions could also benefit from this practice to minimize the risk brought on by big up-front design.

Empower the team

There has been a traditional belief in most businesses about the decision-making in the organization – the managers tell the workers how to do their own job. In a "Work-Out technique", the roles are turned – the managers are taught how to listen to the developers, so they can explain better what actions might be taken, as well as provide suggestions for improvements. The lean approach favors the aphorism "find good people and let them do their own job," encouraging progress, catching errors, and removing impediments, but not micro-managing.

Another mistaken belief has been the consideration of people as resources. People might be resources from the point of view of a statistical data sheet, but in software development, as well as any organizational business, people do need something more than just the list of tasks and the assurance that they will not be disturbed during the completion of the tasks. People need motivation and a higher purpose to work for – purpose within the reachable reality, with the assurance that the team might choose its own commitments. The developers should be given access to the customer; the team leader should provide support and help in difficult situations, as well as ensure that skepticism does not ruin the team’s spirit.

Build integrity in

The customer needs to have an overall experience of the System – this is the so-called perceived integrity: how it is being advertised, delivered, deployed, accessed, how intuitive its use is, price and how well it solves problems.

Conceptual integrity means that the system's separate components work well together as a whole with balance between flexibility, maintainability, efficiency, and responsiveness. This could be achieved by understanding the problem domain and solving it at the same time, not sequentially. The needed information is received in small batch pieces – not in one vast chunk with preferable face-to-face communication and not any written documentation. The information flow should be constant in both directions – from customer to developers and back, thus avoiding the large stressful amount of information after long development in isolation.

One of the healthy ways towards integral architecture is refactoring. As more features are added to the original code base, the harder it becomes to add further improvements. Refactoring is about keeping simplicity, clarity, minimum amount of features in the code. Repetitions in the code are signs for bad code designs and should be avoided. The complete and automated building process should be accompanied by a complete and automated suite of developer and customer tests, having the same versioning, synchronization and semantics as the current state of the System. At the end the integrity should be verified with thorough testing, thus ensuring the System does what the customer expects it to. Automated tests are also considered part of the production process, and therefore if they do not add value they should be considered waste. Automated testing should not be a goal, but rather a means to an end, specifically the reduction of defects.

See the whole

Software systems nowadays are not simply the sum of their parts, but also the product of their interactions. Defects in software tend to accumulate during the development process – by decomposing the big tasks into smaller tasks, and by standardizing different stages of development, the root causes of defects should be found and eliminated. The larger the system, the more organizations that are involved in its development and the more parts are developed by different teams, the greater the importance of having well defined relationships between different vendors, in order to produce a system with smoothly interacting components. During a longer period of development, a stronger subcontractor network is far more beneficial than short-term profit optimizing, which does not enable win-win relationships.

Lean thinking has to be understood well by all members of a project, before implementing in a concrete, real-life situation. "Think big, act small, fail fast; learn rapidly" – these slogans summarize the importance of understanding the field and

the suitability of implementing lean principles along the whole software development process. Only when all of the lean principles are implemented together, combined with strong "common sense" with respect to the working environment, is there a basis for success in software development.

Lean software practices

Lean software development practices, or what the Poppendiecks call "tools" are expressed slightly differently from their equivalents in Agile software development, but there are parallels. Examples of such practices include:

Seeing waste

Value stream mapping

Set-based development

Pull systems

Queuing theory

Motivation

Measurements

Some of the tools map quite easily to Agile methods. Lean Workcells, for example are expressed in Agile methods as cross-functional teams.

Lean manufacturing, **lean enterprise**, or **lean production**, often simply, "**Lean**", is a production practice that considers the expenditure of resources for any goal other than the creation of value for the end customer to be wasteful, and thus a target for elimination. Working from the perspective of the customer who consumes a product or service, "value" is defined as any action or process that a customer would be willing to pay for.

Essentially, lean is centered on *preserving value with less work*. Lean manufacturing is a management philosophy derived mostly from the Toyota Production System (TPS) (hence the term Toyotism is also prevalent) and identified as "Lean" only in the 1990s. TPS is renowned for its focus on reduction of the original Toyota *seven wastes* to improve overall customer value, but there are varying perspectives on how this is best achieved. The steady growth of

Toyota, from a small company to the world's largest automaker, has focused attention on how it has achieved this success.

8.1 Overview

Lean principles are derived from the Japanese manufacturing industry. The term was first coined by John Krafcik in his 1988 article, "Triumph of the Lean Production System," based on his master's thesis at the MIT Sloan School of Management. Krafcik had been a quality engineer in the Toyota-GM NUMMI joint venture in California before coming to MIT for MBA studies. Krafcik's research was continued by the International Motor Vehicle Program (IMVP) at MIT, which produced the international best-seller book co-authored by Jim Womack, Daniel Jones, and Daniel Roos called *The Machine That Changed the World*. A complete historical account of the IMVP and how the term "lean" was coined is given by Holweg (2007).

For many, Lean is the set of "tools" that assist in the identification and steady elimination of waste (*muda*). As waste is eliminated quality improves while production time and cost are reduced. A non exhaustive list of such tools would include: SMED, Value Stream Mapping, Five S, *Kanban* (pull systems), *poka-yoke* (error-proofing), Total Productive Maintenance, elimination of time batching, mixed model processing, Rank Order Clustering, single point scheduling, redesigning working cells, multi-process handling and control charts (for checking mura).

There is a second approach to Lean Manufacturing, which is promoted by Toyota, in which the focus is upon improving the "flow" or smoothness of work, thereby steadily eliminating *mura* ("unevenness") through the system and not upon 'waste reduction' per se. Techniques to improve flow include production leveling, "pull" production (by means of *kanban*) and the *Heijunka box*. This is a fundamentally different approach from most improvement methodologies, which may partially account for its lack of popularity.

The difference between these two approaches is not the goal itself, but rather the prime approach to achieving it. The implementation of smooth flow exposes quality problems that already existed, and thus waste reduction naturally happens as a consequence. The advantage claimed for this approach is that it naturally takes a system-wide perspective, whereas a waste focus sometimes wrongly assumes this perspective.

Both Lean and TPS can be seen as a loosely connected set of potentially competing principles whose goal is cost reduction by the elimination of waste. These principles include: Pull processing, Perfect first-time quality, Waste minimization, Continuous improvement, Flexibility, Building and maintaining a long term relationship with suppliers, Autonomation, Load leveling and Production flow and Visual control. The disconnected nature of some of these principles perhaps springs from the fact that the TPS has grown pragmatically since 1948 as it responded to the problems it saw within its own production facilities. Thus what one sees today is the result of a 'need' driven learning to improve where each step has built on previous ideas and not something based upon a theoretical framework.

Toyota's view is that the main method of Lean is not the tools, but the reduction of three types of waste: *muda* ("non-value-adding work"), *muri* ("overburden"), and *mura* ("unevenness"), to expose problems systematically and to use the tools where the ideal cannot be achieved. From this perspective, the tools are workarounds adapted to different situations, which explains any apparent incoherence of the principles above.

8.2 Origins

Also known as the flexible mass production, the TPS has two pillar concepts: Just-in-time (JIT) or "flow", and "autonomation" (smart automation). Adherents of the Toyota approach would say that the smooth flowing delivery of value achieves all the other improvements as side-effects. If production flows perfectly (meaning it is both "pull" and with no interruptions) then there is no inventory; if customer valued features are the only ones produced, then product design is simplified and effort is only expended on features the customer values. The other of the two TPS pillars is the very human aspect of autonomation, whereby automation is achieved with a human touch. In this instance, the "human touch" means to automate so that the machines/systems are designed to aid humans in focusing on what the humans do best. This aims, for example, to give the machines enough intelligence to recognize when they are working abnormally and flag this for human attention. Thus, in this case, humans would not have to monitor normal production and only have to focus on abnormal, or fault, conditions.

Lean implementation is therefore focused on getting the right things to the right place at the right time in the right quantity to achieve perfect work flow, while minimizing waste and being flexible and able to change. These concepts of flexibility and change are principally required to allow production leveling (Heijunka), using tools like SMED, but have their analogues in other processes

such as research and development (R&D). The flexibility and ability to change are within bounds and not open-ended, and therefore often not expensive capability requirements. More importantly, all of these concepts have to be understood, appreciated, and embraced by the actual employees who build the products and therefore own the processes that deliver the value. The cultural and managerial aspects of Lean are possibly more important than the actual tools or methodologies of production itself. There are many examples of Lean tool implementation without sustained benefit, and these are often blamed on weak understanding of Lean throughout the whole organization.

Lean aims to make the work simple enough to understand, do and manage. To achieve these three goals at once there is a belief held by some that Toyota's mentoring process,(loosely called *Senpai* and *Kohai*, which is Japanese for senior and junior), is one of the best ways to foster Lean Thinking up and down the organizational structure. This is the process undertaken by Toyota as it helps its suppliers improve their own production. The closest equivalent to Toyota's mentoring process is the concept of "*Lean Sensei*," which encourages companies, organizations, and teams to seek outside, third-party experts, who can provide unbiased advice and coaching, (see Womack et al., *Lean Thinking*, 1998).

In 1999, Spear and Bowen identified four rules which characterize the "Toyota DNA":

Rule 1: All work shall be highly specified as to content, sequence, timing, and outcome.

Rule 2: Every customer-supplier connection must be direct, and there must be an unambiguous yes or no way to send requests and receive responses.

Rule 3: The pathway for every product and service must be simple and direct.

Rule 4: Any improvement must be made in accordance with the scientific method, under the guidance of a teacher, at the lowest possible level in the organization.

There have been recent attempts to link Lean to Service Management, perhaps one of the most recent and spectacular of which was London Heathrow Airport's Terminal 5. This particular case provides a graphic example of how care should be taken in translating successful practices from one context (production) to another (services), expecting the same results. In this case the public perception is more of a spectacular failure, than a spectacular success, resulting in potentially an unfair tainting of the lean manufacturing philosophies.

8.3 A brief history of waste reduction thinking

The avoidance of waste has a long history. In fact many of the concepts now seen as key to lean have been discovered and rediscovered over the years by others in their search to reduce waste. Lean manufacturing builds on their experiences, including learning from their mistakes.

8.4 Pre-20th century

Most of the basic goals of lean manufacturing are common sense, and documented examples can be seen as early as Benjamin Franklin. *Poor Richard's Almanac* says of wasted time, "He that idly loses 5s. worth of time, loses 5s., and might as prudently throw 5s. into the river." He added that avoiding unnecessary costs could be more profitable than increasing sales: "A penny saved is two pence clear. A pin a-day is a groat a-year. Save and have."

Again Franklin's *The Way to Wealth* says the following about carrying unnecessary inventory. "You call them goods; but, if you do not take care, they will prove evils to some of you. You expect they will be sold cheap, and, perhaps, they may [be bought] for less than they cost; but, if you have no occasion for them, they must be dear to you. Remember what Poor Richard says, 'Buy what thou hast no need of, and ere long thou shalt sell thy necessaries.' In another place he says, 'Many have been ruined by buying good penny worths'." Henry Ford cited Franklin as a major influence on his own business practices, which included Just-in-time manufacturing.

The concept of waste being built into jobs and then taken for granted was noticed by motion efficiency expert Frank Gilbreth, who saw that masons bent over to pick up bricks from the ground. The bricklayer was therefore lowering and raising his entire upper body to pick up a 2.3 kg (5 lb.) brick, and this inefficiency had been built into the job through long practice. Introduction of a non-stooping scaffold, which delivered the bricks at waist level, allowed masons to work about three times as quickly, and with less effort.

8.4 20th century

Frederick Winslow Taylor, the father of scientific management, introduced what are now called standardization and best practice deployment. In his *Principles of Scientific Management*, (1911), Taylor said: "And whenever a workman proposes an improvement, it should be the policy of the management to make a careful analysis of the new method, and if necessary conduct a series of experiments to

determine accurately the relative merit of the new suggestion and of the old standard. And whenever the new method is found to be markedly superior to the old, it should be adopted as the standard for the whole establishment."

Taylor also warned explicitly against cutting piece rates (or, by implication, cutting wages or discharging workers) when efficiency improvements reduce the need for raw labor: "…after a workman has had the price per piece of the work he is doing lowered two or three times as a result of his having worked harder and increased his output, he is likely entirely to lose sight of his employer's side of the case and become imbued with a grim determination to have no more cuts if soldiering [marking time, just doing what he is told] can prevent it."

Shigeo Shingo, the best-known exponent of single minute exchange of die and error-proofing or poka-yoke, cites *Principles of Scientific Management* as his inspiration.

American industrialists recognized the threat of cheap offshore labor to American workers during the 1910s, and explicitly stated the goal of what is now called lean manufacturing as a countermeasure. Henry Towne, past President of the American Society of Mechanical Engineers, wrote in the Foreword to Frederick Winslow Taylor's *Shop Management* (1911), "We are justly proud of the high wage rates which prevail throughout our country, and jealous of any interference with them by the products of the cheaper labor of other countries. To maintain this condition, to strengthen our control of home markets, and, above all, to broaden our opportunities in foreign markets where we must compete with the products of other industrial nations, we should welcome and encourage every influence tending to increase the efficiency of our productive processes."

8.5 Types of waste

While the elimination of waste may seem like a simple and clear subject it is noticeable that waste is often very conservatively identified. This then hugely reduces the potential of such an aim. The elimination of waste is the goal of Lean, and Toyota defined three broad types of waste: *muda*, *muri* and *mura*; it should be noted that for many Lean implementations this list shrinks to the first waste type only with corresponding benefits decrease. To illustrate the state of this thinking Shigeo Shingo observed that only the last turn of a bolt tightens it—the rest is just movement. This ever finer clarification of waste is key to establishing distinctions between value-adding activity, waste and non-value-adding work. Non-value adding work is waste that must be done under the present work conditions. One

key is to measure, or estimate, the size of these wastes, to demonstrate the effect of the changes achieved and therefore the movement toward the goal.

The "flow" (or smoothness) based approach aims to achieve JIT, by removing the variation caused by work scheduling and thereby provide a driver, rationale or target and priorities for implementation, using a variety of techniques. The effort to achieve JIT exposes many quality problems that are hidden by buffer stocks; by forcing smooth flow of only value-adding steps, these problems become visible and must be dealt with explicitly.

Muri is all the unreasonable work that management imposes on workers and machines because of poor organization, such as carrying heavy weights, moving things around, dangerous tasks, even working significantly faster than usual. It is pushing a person or a machine beyond its natural limits. This may simply be asking a greater level of performance from a process than it can handle without taking shortcuts and informally modifying decision criteria. Unreasonable work is almost always a cause of multiple variations.

To link these three concepts is simple in TPS and thus Lean. Firstly, *muri* focuses on the preparation and planning of the process, or what work can be avoided proactively by design. Next, *mura* then focuses on how the work design is implemented and the elimination of fluctuation at the scheduling or operations level, such as quality and volume. *Muda* is then discovered after the process is in place and is dealt with reactively. It is seen through variation in output. It is the role of management to examine the *muda*, in the processes and eliminate the deeper causes by considering the connections to the *muri* and *mura* of the system. The *muda* and *mura* inconsistencies must be fed back to the *muri*, or planning, stage for the next project.

A typical example of the interplay of these wastes is the corporate behaviour of "making the numbers" as the end of a reporting period approaches. Demand is raised to 'make plan,' increasing (*mura*), when the "numbers" are low, which causes production to try to squeeze extra capacity from the process, which causes routines and standards to be modified or stretched. This stretch and improvisation leads to *muri*-style waste, which leads to downtime, mistakes and back flows, and waiting, thus the muda of waiting, correction and movement.

The original seven *muda* are:

- Transport (moving products that are not actually required to perform the processing)

- Inventory (all components, work in process and finished product not being processed)
- Motion (people or equipment moving or walking more than is required to perform the processing)
- Waiting (waiting for the next production step)
- Overproduction (production ahead of demand)
- Over Processing (resulting from poor tool or product design creating activity)
- Defects (the effort involved in inspecting for and fixing defects)

Later an eighth waste was defined by Womack et al. (2003); it was described as manufacturing goods or services that do not meet customer demand or specifications. Many others have added the "waste of unused human talent" to the original seven wastes. These wastes were not originally a part of the seven deadly wastes defined by Taiichi Ohno in TPS, but were found to be useful additions in practice. In 1999 Geoffrey Mika in his book, "Kaizen Event Implementation Manual" added three more forms of waste that are now universally accepted; The waste associated with working to the wrong metrics or no metrics, the waste associated with not utilizing a complete worker by not allowing them to contribute ideas and suggestions and be part of Participative Management, and lastly the waste attributable to improper use of computers; not having the proper software, training on use and time spent surfing, playing games or just wasting time.For a complete listing of the "old" and "new" wastes see Bicheno and Holweg (2009)

Some of these definitions may seem rather idealistic, but this tough definition is seen as important and they drove the success of TPS. The clear identification of non-value-adding work, as distinct from wasted work, is critical to identifying the assumptions behind the current work process and to challenging them in due course. Breakthroughs in SMED and other process changing techniques rely upon clear identification of where untapped opportunities may lie if the processing assumptions are challenged.