Lean for Systems Engineering

Niels Malotaux, Project Coach

Last year I presented at the "Lean & Agile for Systems Engineers" seminar in Stockholm organized by INCOSE-Sweden. I was asked to talk about "How is it possible that most organizations still survive while their competitors are applying Lean?" (short answer: "They don't") and "My Practical Approaches for Becoming Lean and Really Agile". I was also asked first to present a "Lean & Agile Primer", in order to set the stage: what are we actually talking about? I wanted to avoid the usual hallelujah stories about Lean and Agile, so I researched to find and present about the "essence" of these ideas, and how these ideas can actually improve systems engineering results. You can download the presentations: www.malotaux.nl/doc.php?id=31 .

Here are some of my findings about Lean. Perhaps I can write another time about my findings about Agile. Short advice: avoid the hype and the tools; try to find the essence and go for that.

Lean

The origin of the word "Lean" in the present context comes from people from MIT, who tried to find the secret behind the Japanese Car Production 'miracle' (read Toyota). The word started the Lean hype with the book "The Machine that Changed the World, the Story of Lean Production" by Womack, Jones and Rook (1990). A reviewer of the book noted: "A lot of the cost of vehicles is based on: bad design, poor management, and an attitude that problems, no matter how small, can be overlooked". Sounds familiar?

Womack writes: The goal is reduction of waste. To achieve this, a company must look at what creates value and eliminate all other activities:

1. Specify *value* from the standpoint of the end customer by product family.
2. Identify all the steps in the value stream for each product family, eliminating whenever possible those steps that do not create value.
3. Make the value-creating steps occur in tight sequence, so the product will flow smoothly toward the customer.
4. As flow is introduced, let customers pull value from the next upstream activity.
5. As value is specified, value streams are identified, wasted steps are removed, and flow and pull are introduced, begin the process again, and continue it until a state of perfection is reached in which perfect value is created with no waste.

Nice sounding words, but what do they really mean?

I decided, instead of reading second-hand material, to go to the Japanese original, Taiichi Ohno's book: "The Toyota Production System - Beyond Large Scale Production" (1978), as well as background information from other Japanese sources. I found "Around 1950 the company (Toyota) nearly went bankrupt and had to lay off one third of its employees." This stimulated Ohno to focus on four specific *aims* (with my comments):

- ***Deliver the highest possible quality and service to the customer***
  Without quality, the customers won't buy, and if the customers don't buy our products, all other issues are irrelevant. Is it coincidental that Deming presented his speech to Japanese top-management in 1950, saying just that?

- ***Develop employees' potential based upon mutual respect and cooperation***
  The employees create the value. All management can do is to *facilitate* and help them to create the value more efficiently. Still, "Death by Overwork" is an official Japanese Government statistic and Toyota only recently decided that overtime should be limited to 360 hours per year, which may mean that people are still expected to work overtime *on average* up to 1.5 hour every day.

- ***Reduce cost through eliminating waste in any given process***
  There are so many things we tend to do which do not contribute to creating value for the customer. It is the task of everybody in the organisation to recognize and actively eliminate waste in whatever we do, and not do.

- ***Build a flexible production site that can respond to changes in the market***
  Because Toyota initially didn't build so many cars, they had to adapt what they learnt from Ford's mass-production (*you can buy any model and any colour, as long it's a model T and it's black*) towards making small-scale and varying production as efficient, or even better.

To quote some lines from Ohno's book:

- When asked what Toyota was currently doing, Ohno replied:
  "All we do is looking at the TimeLine from Order to Cash, removing the non-value-adding wastes"
  This provides a clear focus to continuous improvement, not wasting time and resources nobody is waiting for.

- **“The Toyota Production System began when I *challenged* the old system”**
  Continuously challenging the old system is the basis of improvement. As Einstein and Benjamin Franklin already said: Insanity is doing the same thing over and over again and expecting a different outcome. The old system apparently needed improvement, so it always should be challenged. We should be so humble as to admit that we *always* can improve. For those who think that continuously improving quality of what we do and what we produce does cost more: Crosby already had found that Quality is Free. Even in 1950 Deming had already told the Japanese this. It is my own observation that *quality is cheaper*.

- **“Necessity is the mother of invention: improvements are made on *clear purposes and need”***
  We see the US Lean movement pushing the tools and techniques they found at Toyota. At Toyota, they make improvements on *clear purposes and need*. Their tools and techniques emerged out of necessity and *will be different tomorrow*, because the improvement cycle never stops. This makes me remember the story of what a US mission to Japan found: they visited many successful Japanese companies and saw that every morning the people were doing exercises. Back at home they advised US companies that doing morning exercises would make them more successful. The mission failed to observe unsuccessful Japanese companies, where people also did morning exercises every day.

- **“The TPS has been built on the practice of *asking “Why?” 5 times”***
  Before jumping to implementing the first solution that comes to mind, better first develop the problem, then look for possible solutions and then select the most promising one. Five times “Why?” is usually sufficient to find the root problem from which we then can design the appropriate solution. In practice we see so many projects developing a perfect solution for the wrong problem. Better first *develop the problem!*

- **“The time that provides me with the most *vital information about management* is the time I spent in the plant, *not in the office”***
  The ‘working floor’, where the people are creating the value, is the place where you can observe what can be improved, and how you can help the people who create the value to become more effective and efficient. Sitting behind your office desk you will create theoretical solutions for theoretical problems that do not work as expected in practice. This is what Masaaki Imai describes in his book *“Gemba Kaizen - A Common Sense, Low-Cost Approach to Management”* (1986), Gemba being “the working place” and Kaizen being “continuous improvement”.

- **“Toyota’s top management watched the situation quietly and I admire the attitude they took”**
  What Ohno did was not very Japanese: he squarely told people the truth, instead of trying for them not to ‘lose face’. He got quite some opposition and he admits that without clear Executive Support he wouldn’t have been able to implement his ideas. Sounds familiar?

Where the US version of Lean seems to focus on tools and techniques (which of course sell nicely), the Japanese simply used and keep using the Deming Cycle (Plan-Do-Check-Act) and asking five times “Why?” to find solutions for their own particular issues and circumstances in order to continuously improve. After all, they are not in the business of selling a “method”.

Ohno is a modest man, admitting that he actually got his ideas when reading Henry Ford’s books and looking at US supermarket supply principles. Like Henry Ford, he took old ideas and adapted them for his *clear purposes and need*.

Those who don't learn from history are doomed to repeat it (after Santayana), so let's dig up some more roots of so-called ‘Lean’ ideas:

- **Benjamin Franklin** (1706-1790)
  - Waste nothing, cut off all unnecessary activities, plan before doing, be proactive, assess results and learn continuously to improve

- **Henry Ford** (1863-1947)
  - *My Life and Work* (1922)
    - We have eliminated a great number of wastes (at the farm in Dearborn)
  - *Today and Tomorrow* (1926)
    - Learning from waste, keeping things clean and safe, better treated people produce more

- **Toyoda’s** (Sakichi, Kiichiro, Eiji) (1867-1930, 1894-1952, 1913-)
  - Jidoka: Zero-Defects, stop the production line (1926)
  - Just-in-time - flow - pull

- **W. Edwards Deming** (1900-1993)
  - Shewhart cycle: Design-Produce-Sell-Study-Redesign (Speech to Japanese management, Japan - 1950)
  - Becoming totally focused on quality improvement (idem)
    Management to take *personal responsibility* for quality of the product
  - *Out of the Crisis* (1986) - Quality reduces waste

- **Joseph M. Juran** (1904-2008)
  - *Quality Control Handbook* (1951, taught in Japan - 1954)
  - Total Quality Management - TQM
  - Pareto Principe

- **Philip Crosby** (1926-2001)
  - *Quality is Free* (1980)
  - Zero-defects (1961)

- **Taiichi Ohno** (1912-1990)
  - (Implemented the) *Toyota Production System (Beyond Lange-Scale Production)* (1978, 1988 in English)
  - Absolute elimination of waste - Optimizing the TimeLine from order to cash

- **Masaaki Imai** (1930-)
  - *Kaizen: The Key to Japan's Competitive Success* (1986)
  - *Gemba Kaizen: A Commonsense, Low-Cost Approach to Management* (1997)

Ohno says there are two Pillars of the Toyota Production System:

- **Just in Time** (JIT)
  No inventory
  Inventory is stuff waiting to be used later. Inventory deteriorates and has to be kept, managed, transported, which is all considered waste, because it doesn't create value, it only costs. Ohno admits that the idea is simple, but the implementation is a lot of hard and continuous work. An example of inventory with Systems Engineering is detailing requirements way before they are used. Once they are needed they are deteriorated by improved insight in what the project really is about.

- **Perfection** (this is not formally a part of Ohno's two pillars, but I think it's implied)
  - Perfection is a condition for JIT to work. If the things that are delivered Just In Time are not in perfect order, they cannot be used, the flow is disrupted and it's not any more JIT.
  - If a defect is found: stop the line, find the cause, fix it immediately.
    Use root-cause-analysis to find the root of the problem, or as they say: there is a process that is producing this defect. Find this process and eliminate it, or replace it with a process that doesn't produce the defect. Continuous improvement of product, project and process.

- **Autonomation**
  - The loom runs unattended until signalling it needs help.
    Originally, the Toyoda's produced autonomous looms. If one wire at the loom breaks, it's of no use of keeping the loom running, because it'll be producing defective and therefore unusable cloth. If the looms run autonomously and signal a defect when it occurs, one operator can attend many looms. The sales of Toyoda's patents on autonomous looms to a company in the UK provided the money to start developing automobiles.

  If we translate Autonomation to development:

  - The development team runs unattended until signalling it needs help (caused by an issue beyond the team's control)
  - Management observes the team and proactively *facilitates* them to become ever more efficient. Management prevents issues delaying the team beyond the team's control - Education, Empowerment and Responsibility of people. This is one of the most important tasks of management. If management doesn't remove waste proactively (mainly *before* it happens!) or even causes waste, management is to be regarded as waste in Lean terms, and therefore should be removed or replaced.
  - If an issue still does occur, management helps to *remove obstacles* quickly, making sure it doesn't happen again.

We see here an enlightened way of management, not policing with command and control, but rather management being subservient to the workers to help them to become more efficient. Quite a different attitude from what we see in contemporary management attitudes in the West. It was, however, borrowed from the original type of management that, before World War II, made the US companies so powerful and prosperous. This knowledge was exported to Japan after the war (see *"CCS Management Course"*) and in the US replaced by what the Hopper brothers in their book "*The Puritan Gift*" call the "*Cult of the (so called) Expert*", showing how this attitude led to the current economic crisis.

Capacity = Work + Waste

The capacity in an organization is used for:

- **Work**, which creates value
- **Non-value adding, but necessary, work**
  Example: managers are waste, unless they organize, facilitate, and optimize the work of the workers, increasing the value produced by the workers more than the management's added cost.

- **Waste**: anything that does not contribute to value creation
  "Because it costs nothing, eliminating waste is one of the easiest ways for an organization to improve its operations."

Identifying Waste

People are not stupid. If it were easy to recognize and eliminate waste, it would have been done already. Apparently, it's counter-intuitive. To help us identifying waste, Ohno mentions seven types of waste, to which an eighth was added later. In the following table we see these wastes, with a translation into what these would mean to development (or systems engineering tasks) and *possible* remedies to do something about it (not necessarily the best remedies, but examples to make you start thinking):

Manufacturing	**Development / SE** (after Poppendieck)	***Possible* Remedies** (my possible suggestions)
Overproduction	Extra features Unused documents	Prioritizing, real requirements, Deciding what not to do
Inventory	Partially done work	Synchronization, Just In Time
Transport	Handoffs	Keeping in one hand/mind: - Responsibility (what to do) - Knowledge (how to do it) - Action (doing it) - Feedback (learning from result)
Processing	Design inefficiency Wishful thinking	Knowledge, experience, reviews Preflection
Waiting	Delays	Process/Organization redesign
Movement	Task Switching	Maximum 2 tasks in parallel
Defects	Defects	Prevention
Ignoring ingenuity of people	Ignoring ingenuity of people	Real management, Empowerment Bottom-up responsibility

5S and 3Mu

When people talk about Lean, you will soon hear about 5S and about the 3 Mu's. These are an aid for people to internalize the essence:

• Seiri	-	Remove unnecessary things	→ waste
• Seiton	-	Arrange remaining things orderly	→ flow
• Seiso	-	Keep things clean	→ uncovers hidden problems
• Seiketsu	-	Keep doing it, standardize	→ know what to improve
• Shitsuke	-	Keep training it	→ resisting inevitable entropy

• Muda	-	Waste	→ minimize waste
• Mura	-	Irregularities	→ optimize flow
• Muri	-	Stress	→ sustainable pace

What is *real* Value ?

When Terminal 5 was built at Heathrow (London Airport), it was a huge technical success. It was quite an achievement, to build such a multi-billion pound project *on time* and *on budget*. However, the people for whom the terminal was actually built (without passengers there is no need for a terminal!) aren't interested in the technical details of a terminal.

They only want to:

- check-in their luggage as *easily* as possible, and
- get their luggage back as *quickly* as possible in *acceptable* condition *at their destination.*

They didn't.

One of the problems is to determine what the value of the project (or our work in general) really is about. By the way, a project doesn't even deliver value itself. It only can deliver the *conditions for the users* of the system to create the value.

This is about what we call *Real Requirements*[1]. Requirements engineering is quite underdeveloped in most projects. Why? I think it's lack of proper education.

[1] See e.g. Ralph R. Young, Effective Requirements Practices (Addison-Wesley, 2001) for a thorough discussion of 'real requirements".

Example of identifying waste (Value Stream Mapping)

Assume that the users of our system encounter a problem that costs them extra time (*Problem occurring*: negative value or waste to the users), see Figure 1:

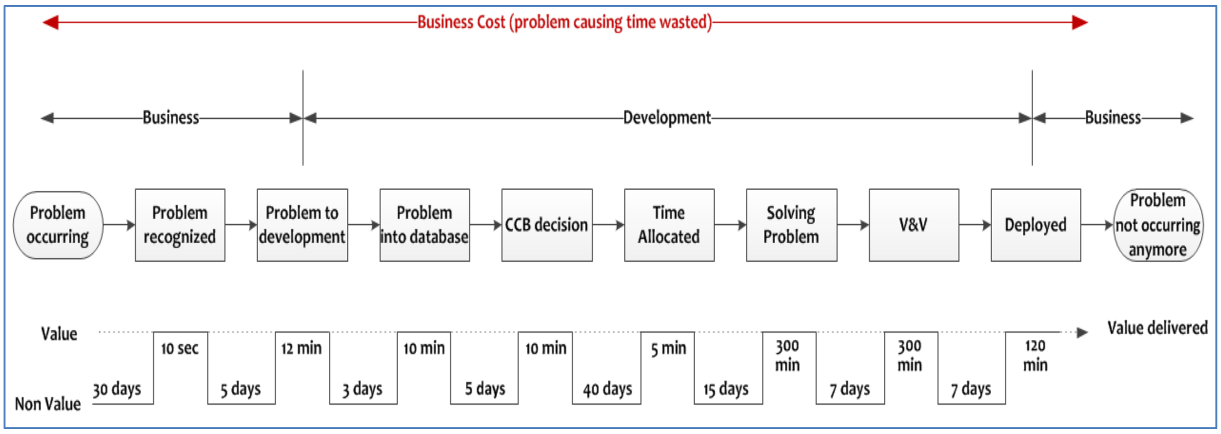

Figure 1: Value Stream Mapping Example

Then it may take some time:

- before some of the users start complaining (*Problem recognized*)
- before the problem is reported to those who can do something about it (*Problem to development*)
- before development, (receiving it e.g. by email) puts it in their issue -database (*Problem into database*)
- before the Change Control Board decides whether and how to handle the issue (*CCB decision*)
- before handling the issue is scheduled (*Time allocated*)
- before the problem gets solved
- before the solution is verified and validated (*V&V*)
- before the solution is deployed
- before the problem doesn't hinder the users anymore.

In this process, there are a lot of value-adding activities, all (seemingly) necessary to create the value of the solution. It's obvious that between these activities, there is a lot of waiting time.

From this example we can see the following:

- The total time the users keep having the problem is 114 days.
- The value-adding time is 1.6 days and the non-value adding time is 112 days.
- If, using our system, twice a day 100 users experience this problem, spending an extra 10 minutes every time, the wasted business cost of the users waiting for the solution is (with some assumptions):
  2 times per day x 100 people x 10 minutes extra x 112 days x $400 per day = $187k.
- The net cost of producing the value of the solution is (with some more assumptions):
  3 people x 1.6 days x $1000 per day = $5k.

While solving the problem, we are actually wasting almost $187k, instead of spending 'only' $5k. Having mapped the "value stream", we can now look for minimizing the delays between the value adding stages. We may challenge the necessity and efficiency of the value adding stages themselves. And of course we also do a root-cause analysis to determine why we produced the problem in the first place. Still, we must be careful not to optimize everything at once, because if we try to perfect too much in too short a time, we will probably fail.

So-called "Process Improvers" usually start "improving" the value adding activities *within* the boxes (see figure1). Now we can see that, in this case, they should better start with removing the waste *between* the boxes.

Value Stream Mapping in Development or Testing

The previous example is used for *production*: repeated actions, which we can observe and then improve on. Some people maintain that this cannot be applied to development. A lot of what we do in development as well as in testing, however, is more of the same, with a lot of waiting and inefficient synchronizing. Here we can use the same regular value stream mapping principles as described. Some development activities (mainly *within* the boxes of figure 1) are, however, specific to this particular development, and analysis after the fact has not much use, because once we find out that we could have prevented the waste, the time has already been spent and the waste is already produced. Instead of relying only on *reflection* on what we should have done better, we'd better use *preflection*: *imagining* which elements of what we are *going to do* will produce waste, *before doing it*, and then *not doing it*. This is the only way to avoid this type of waste and this is what we routinely do with the Evolutionary Planning approach (see www.malotaux.nl/?id=evoplanning). An example of how waste is prevented *before the time is spent* can be found at www.malotaux.nl/?id=designdelivery

We use ‘magic sentences’ like:

- “Why are we doing this?”
- “Is it really necessary?”
- “Is it really necessary *now*?”
- “Who’s waiting for it?”
- “How much time would you need?” – “How much time do you have?” – “How much time should you use?” – “How much time will you give yourself?”

Planning in this way, we routinely save 30% of the time while producing better results in all other respects. Because it’s so easy, people learn to save time within just several weeks. Why don’t they already do this, or why does it take even several weeks? Because what people have to do is partly counter-intuitive. That’s why it hasn’t happened spontaneously already. A coach helps them by saying “Can you do this 3 weeks for me? Trust me, it has always worked, at least until now!” If then, within 2 weeks people find it starts working for them, new intuition is born. From then, it will go automatically and the coach can start focusing on other issues.

But I’m only a Systems Engineer! What has this to do with me?

What has this to do with me? After all, I’m a systems engineer (developer, tester, or whatever else) and not a project manager. Well, the project manager is *responsible* for the project to deliver a very effective and efficient result. However, the workers in the project and especially the systems engineers *determine* the time they spend, and the *efficiency* of the solutions they provide for the users. This makes systems engineers even more responsible for being aware of all time elements:

- in the product (system) they are conceiving
- in the project, how they come to good solutions, how they organize their work
- in the processes used to develop the product
- in the processes their product (system) uses to support the users letting the value crystallize.

Some snippets

- **Womack**
  Most managers think their greatest contribution to the business is doing work-arounds on broken processes, rather than doing the hard work to get the process right so that it never breaks down.

- **Imai,** in Gemba Kaizen - A Commonsense Low-Cost Approach to Management
  90 per cent of all corporate problems can be solved using common sense and improving quality while reducing cost through the elimination of waste.

- **Tom Harada (worked under Ohno)**
  Plan-Do-Check-Act cycle was by far the most important thing we did in hindsight.
  Not to make this article too long, please read the page on Plan-Do-Check-Act on my website:
  www.malotaux.nl/?id=pdca
  Plan-Do-Check-Act is the basis for continuous Root-Cause-Analysis and for continuous improvement.

- **A project manager**
  Root-Cause-Analysis on every defect found? We don’t have time for that!
  Apparently people have no time to do things First Time Right and they have all the time when things go wrong. The development budget is squeezed and the *panic budget* is unlimited. This is one of the reasons why projects take way more time than necessary. Why don’t people learn? Because it’s counter-intuitive!

Conclusion

Some essential ideas behind Lean:

- Delivering the right stuff, the right way, at the right time, as efficiently as possible
- Understanding what real value means
- Quickly and easily adapting to *all* Stakeholders (however beware: only the Customer pays, the other Stakeholders don’t mind the cost!)
- (especially for software) Total system focus - software is only an aid – it only provides value *when it is used successfully*
- Continuous elimination of Waste
  - Doing what contributes the most value
  - Not doing what does not contribute value
  - Prevention rather than repair - relentless problem solving - root cause analysis
  - Perfection - *Quality is cheaper*
- Predictability: Continuously being able to tell what will be done when (to take appropriate action)
- Delivering in small steps to real Stakeholders doing real things
  Minimizing the waste of incorrect perceptions, assumptions and implementations, optimizing productivity of Stakeholders – not just *demos*
- Continuously optimizing what we do, how we do it, and how we organize things using PDCA

- Empowerment - everybody feeling responsible for the Result (remember the *Goal of a Project* see www.malotaux.nl/?id=goalofaproject)
- Assertiveness - actively removing impediments, no need for excuses
- Understanding that it's not about tools: a lot is attitude and craft (you cannot 'implement' Lean)
- Management *facilitating* and *coaching* the workers to do the right things the right way at the right time
- Management to be personally responsible for continuous *improvement* (not just change)
- JIT, Autonomation, Kanban, Value Stream Mapping, Flow, 5S, 3Mu, etc. All good to know techniques, however, beware of technique-fetishism!
- Involving the whole organization

Looking at what I've learnt during the past decades in my work to help projects, greatly improving their performance, it's all Lean: not doing what is not necessary, not spending more than necessary, and continuous pursuit of perfection. I call it *Quality on Time*. My father, as a business consultant, did similar things in the 60's, optimizing the flow and performance of e.g. bicycle production, without ever having heard of Toyota. That's not surprising, because after all it's just common sense, as he used to say. However, "common sense apparently is not so common", is a sentence I kept hearing in the corridors of the INCOSE International Symposium in Chicago. A lot of the things that we have to do to make projects successful and on time are counter-intuitive, which makes it almost impossible for this to happen spontaneously. Without active and continuous coaching by management, it will not happen. If management doesn't know yet how to do this, they can ask external coaches for assistance.

Reading

- Philip B. Crosby, *Quality is Free*, 1979, *Quality is Still Free*, 1996. ISBN 0070145326
- W. Edwards Deming
  - *Out of the Crisis*, 1986. ISBN 0911379010
  - *Speech for Japanese Management*, 1950, www.jsdstat.com/Statblog/wp-includes/Hakone.pdf
- K. and W. Hopper, *The Puritan Gift*, 2007. ISBN 9781845119867
- Masaaki Imai, *Gemba Kaizen: A Commonsense, Low-Cost Approach to Management*, 1997. ISBN 0070314462
- Taiichi Ohno, *Toyota Production System: Beyond Large-Scale Production*, English version 1988. ISBN 0915299143
- Henry Ford, *My Life and My Work*, 1922. ISBN 161743020X
- Henry Ford, *Today and Tomorrow*, 1926. Reprint 1988, 2003. ISBN 0915299364
- Benjamin Franklin, *Autobiography*, various publishers. ISBN 0486290735
- Homer M. Sarasohn and Charles A. Protzman, *The Fundamentals of Industrial Management - CCS Management Course*, 1949. http://www.cmis.csiro.au/opm/publications/PDF/ccs_manual_complete.pdf
- Charles Womack et al., *The Machine That Changed the World*, 1990. ISBN 0743299795
- Website INCOSE Lean-SE Working group (192 Lean Enablers): http://cse.lmu.edu/about/graduateeducation/systemsengineering/INCOSE.htm

Niels Malotaux is an independent Project Coach and expert in optimizing project performance, helping organizations around the world to optimize the predictability of their projects. He has some 35 years experience in designing hardware and software systems, at Delft University, in the Dutch Army, at Philips Electronics and 20 years leading his own systems design company. Since 1998 he is devoting his expertise to helping projects to deliver Quality On Time: delivering what the customer needs, when he needs it, to enable customer success. Niels effectively teaches Evolutionary Project Management (Evo) Methods, Requirements Engineering, and Review and Inspection techniques. Since 2001, he has taught and coached well over 100 projects in 25+ organizations in the Netherlands, Belgium, China, Germany, Ireland, India, Israel, Japan, Romania, South Africa and the US, which has led to a wealth of experience in which approaches work better in practice, and which work less well. He is a frequent speaker at conferences, see www.malotaux.nl/?id=conferences