Jim Coplien reveals the new building blocks for Lean software *p. 4*

Lean MAGAZINE

ISSUE #5 | JANUARY 2010

Scania's new architecture role for a Lean software organization *p. 9.*

SPECIAL ISSUE ON LEAN SOFTWARE ARCHITECTURE

24 pages on Agile Development!

How to build an effective team in a critical project *p. 14*   Scrum news *p. 22*

Only a foolish man buildeth his house upon the sand

In the early days of Agile Development there were some that believed that all tedious stuff from older methodology wasn't needed anymore, and some initial Agile failures can probably be credited to negligence of creation of a solid system architecture. There were even some rumors that enthusiastic TDD-folks (Test Driven Development) were convinced that a test-driven approach was all that was needed and a good system architecture would emerge from the test structure in some magic way.

But even if the discipline of system architecture is nowadays generally accepted as a vital part of any software project, it has changed in character. New concepts have been developed in recent years – both on the technical and on the organizational side. Jim "Cope" Coplien, the father of organizational patterns, opens this issue with an article addressing architecture in Agile and Lean development, as well as introducing a new architecture concept, called DCI (Data-Context-Interaction).

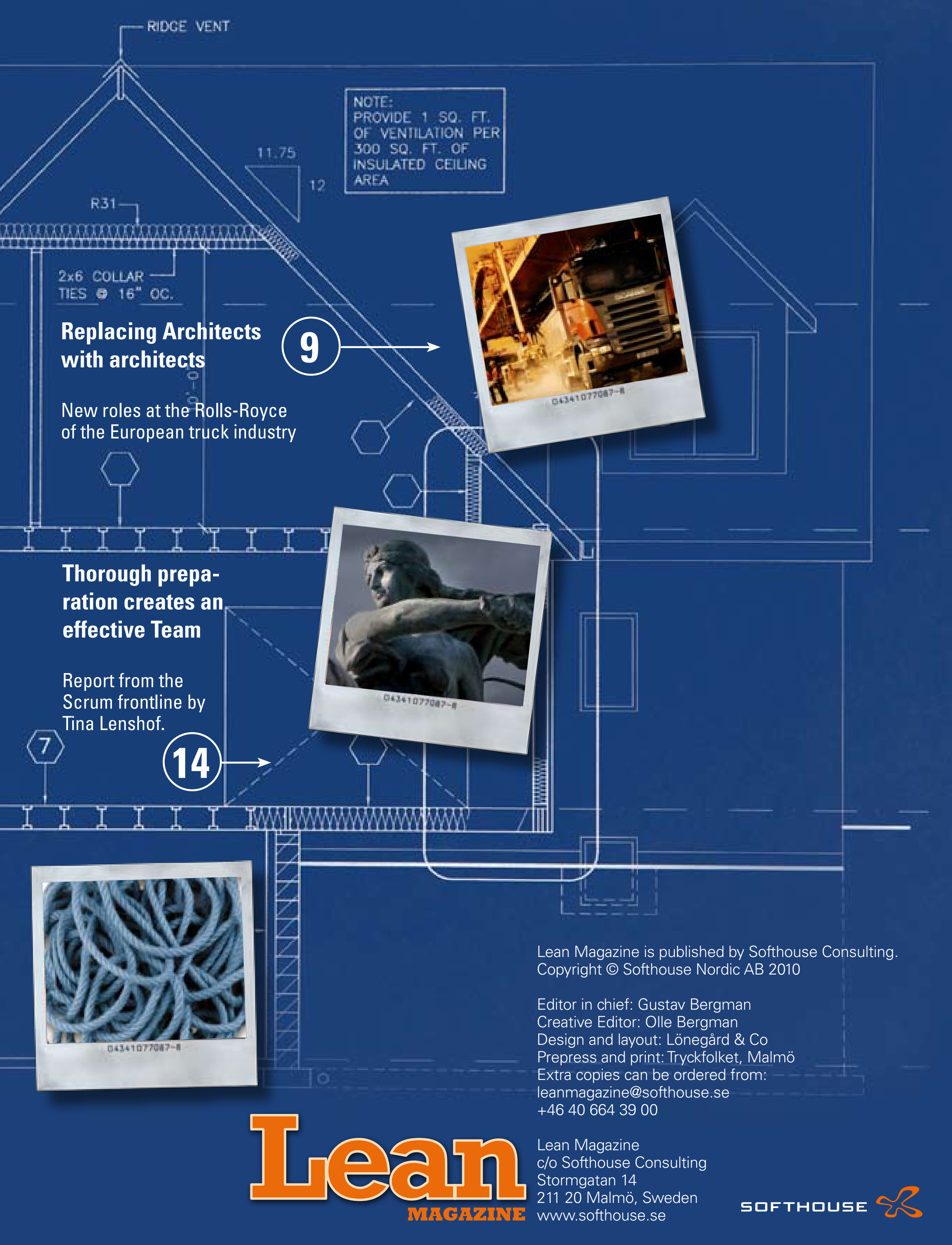
The organizational aspect of architecture is treated by Staffan Persson, System Architect at Scania, who presents the architect's new role in Scania's organization.

Together with our feature topic we have the usual mix of new and interesting pieces on Lean and Agile. Tina Lenshof from Softhouse talks about Scrum Teambuilding, Krister Kauppi from Elicit writes on the relationship between Lean and Agile, and the concluding essay on the last page is this time about Lean and Agile elements in board games by Softhouse's game aficionado Ola Sundin.

Good reading to everyone,

Gustav Bergman,
Editor in chief, Lean Magazine

Photo: André de Loisted



Lean Magazine is published by Softhouse Consulting.
Copyright © Softhouse Nordic AB 2010

Editor in chief: Gustav Bergman
Creative Editor: Olle Bergman
Design and layout: Lönegård & Co
Prepress and print: Tryckfolket, Malmö
Extra copies can be ordered from:
leanmagazine@softhouse.se
+46 40 664 39 00

Lean Magazine
c/o Softhouse Consulting
Stormgatan 14
211 20 Malmö, Sweden
www.softhouse.se

Lean MAGAZINE

SOFTHOUSE

Photo: istockphoto.com

Jim Coplien:

LEAN & AGILE AND THE MATTER OF ARCHITECTURE

By Jim Coplien, Gertrude & Cope

Software Architecture was often neglected in the early years of the agile movement. However in recent years most developers have learnt to appreciate its importance. In this article, Jim Coplien – the author of Wiley's upcoming book "Lean Software Architecture" (see page 22) – gives an overview of architecture's role in the Lean and Agile movements, and tells us about new interesting concepts that are emerging.

The pendulum of change

Mary Poppendieck described in a 2008 talk at Øresund Agile how the pendulums of practice swing back and forth over the years. I've seen this in my 40 years in the industry, and software architecture has always been one of these pendulums. I can't quite find the metaphor that fine-tunes Mary's vision to describe how the pendulum slams from one opinion to the other, and back again. The metaphor should invoke a vision of moving deliberately through levels of learning. Perhaps our entire industry is Agile at its very foundations, reacting eagerly to changes it induces itself. Instead, perhaps we should be responding responsibly to the changes in our environment.

Changing fashions of software architecture

Software architecture made it into the software vernacular after a talk between Jerry Weinberg and Fred Brooks at IBM where Jerry encouraged Fred to follow through with his metaphor. The pendulum had a firm beginning at center-right. Architecture stayed in vogue for large projects until the early 1990s when it became unfashionable. Why? It had started to become overdone: sometimes a detached exercise for its own sake, driven out of fear of uncertainty and change, creeping ever more strongly to the right. Then, the pendulum slammed to the left. But ignoring architecture also proved problematic, and now the pendulum is moving back the other way. Bob Martin says, 'One of the more persistent myths of agile development is that up-front architecture and design are bad … Pardon me, but that's Horse Sh--.'

What is lean software architecture?

Lean architecture comes from applying the principles of the Toyota Production System to software architecture. "Lean" means to get rid of waste (like unnecessary documentation), inconsistency (like mismatched interfaces), and irregularity spaced development work in production. Lean understands that you do deliberate analysis and planning before going into production, using techniques like set-based design that explore every viable alternative. The word "Lean" applies to both the assembly line and to the car being built, but also describes the processes behind them. Lean is both about the thing and the process, reminiscent of what good generative patterns are. Lean architecture is both about an architecture with no fat, and about the consistency and reduction of waste in the process surrounding its creation and use.

A place for everything

Lean means discipline in maintenance, too. Yes, the Toyota Way goes beyond just the Toyota Production System (TPS) into Total Production Maintenance (TPM). One common aspect of TPS and TPM is that everything has its place. In TPS, there is a technique called poka-yoke or "fool-proofing" that ensures that pieces are put together correctly. It is like the concept of a design jig in craftsmanship. In software, architectural partitioning and interfaces guide feature programmers to the code for a specific domain, clarifying the code's place in the context of the entire system. In TPM, the tool board has tool outlines for each service tool, so each has its place. These organizations, relationships, and loci are carefully planned up front.

Forming the shape of the system

In Lean software architecture, we use Domain-Driven Design (DDD) to come up with the system form. The result of this process is the shape of the system. In the same sense that the essence of a Toyota steering wheel is captured in the plastic injection mould used to build it, so the essence of the system is captured in its architecture. We can tailor the steering wheel in many ways, just as we can tailor an abstract base class with many derived classes suitable to their respective markets.

Lean architecture delivers APIs: usually abstract base classes, with argument declarations and other code annotations that describe the relationships between them. It doesn't include details of data structure or method definition. It is architecture in the true historic sense of the word as a kind of pure form that delays structure.

Jim Coplien

Jim ("Cope") Coplien (Gertrud&Cope, Denmark), Ph.D., CST, CSM, CSP, is the father of Organizational Patterns, is a co-founder of the Software Pattern discipline, a pioneer in practical object-oriented design, and a widely consulted authority, author, and trainer in software design and organizational development.

Photo: Christopher Roos

The structure, we deliver just-in-time, prompted by the need to support a use case. This just-in-time notion is another key Lean tenet.

Is a lean architecture agile?

Now back to the other buzzword: Agile. Should software be both Lean and Agile? If we look at the words carefully, Lean applies to the system form and how it relates to the domain structure of the business and of technology. It is a complicated structure created by a complicated process. However, it needn't be complex. If something is complicated, I can take it apart and put it back together again, as an auto mechanic can do with a car. Complex things, on the other hand, are more than the sum of their parts. Software is both complicated and complex. Most of software's complexity comes not from form, but from the domain of time and software behavior. Use cases are what make software complex, partly because of the high rate of change within the system during a use case, and partly because a human being is usually involved. We tend to be complex creatures.

The architect Stewart Brandt notes that architectures have shear layers: layers of different rates of change in a house. The stone foundations or load-bearing walls may be modified once a century. Other walls that serve to partition space may come and go on the scale of a few decades. Windows and doors may come and go every decade or so; the carpeting a bit more frequently, and the internal décor on the scale of the seasons. A good software system has a Lean architecture that captures the rather stable complexity of its application and solution domains, and the complex mapping between then. On top of that is the shear layer of features that respond day-by-day or month-by-month to customer requests.

Therefore, Lean architecture has another side, which is its Agile application. In the same sense that a Toyota engineer develops a car so you can drive it through a complex race course in dynamic driving conditions, so a Lean architecture supports Agile adaptation of the system to the market.

Individuals and interaction, and usable code

Agile is about change. But Agile is also about individuals and interactions, and about software that works. A software system that works integrates seamlessly with the people who use it. That means that its structure should correspond to the mental model of the end users. End users interact with systems on the basis of their mental model of the objects on the other side of the screen. If the program objects don't map those in the end user's head, confusion results – and that violates the Agile provision for interactions with individuals. The programmer is also an individual, one who wants to separate the slow-changing foundations from DDD from the rapidly changing Use Cases. But the end user doesn't have this dichotomy! How do we resolve this?

Agile and Lean require new kinds of building blocks

The answer lies in the difference between classes, objects, and roles. Classes are units of source code and what the programmer writes. Objects are the units of program execution, and are part of the end-user cognitive model. Roles are the units of end-user model of action: a user understands an object in terms of the roles that it plays rather than in terms of the object itself. If we investigate the use case for a money transfer between accounts we will encounter roles like Source Account and Destination Account. Those aren't objects – my Salary

>>

DCI

DCI (Data-Context-Interaction) is a so called Software Pattern for object oriented programming which has been developed by Trygve Reenskaug in recent years. It advocates a dynamic behavior of the software objects built on its role in each particular context, and de-emphasizes the ties between data and behavior which is central to the currently most popular pattern MVC (Model-View-Controller), developed by Reenskaug at Xerox PARC in the late seventies.

» Account is one of my accounts and my Savings Account is another, and either one may play either of roles Source Account or Destination Account. We want programmers to be able to deal with these separately because they change at different rates for different reasons, but we want the object that reflects the end-user model to exhibit all the behaviors of this role it is playing. In programming terms, that means being able to glue together the domain class and the role into a single class whose objects meet end user expectations.

The new building blocks: the DCI Architecture

Trygve Reenskaug's DCI architecture (Data, Context, and Interaction) is a way to organize this role-to-object mapping while properly balancing the concerns of the end users with those of the programmers. DCI starts with groupings of business functionality called Contexts (the "C" in DCI). A Context roughly corresponds to a use case, and a new Context object is instantiated at the start of each scenario. Contexts get their work done through Interactions (the "I" in DCI) between roles. An algorithm is a series of actions, where each action applies to some role like a Source Account or Destination Account. We can code up complete, generic algorithms in terms of methods on these roles. At the beginning of each scenario, the Context injects its roles into domain objects that do the work. These domain objects (the Data – the "D" in DCI) are the Models of MVC, or the basic building blocks from DDD.

The Context directly captures the scenarios of the end user mental model in terms of the roles by which end users conceptualize them, supporting the Agile agenda of customer collaboration. Programmers can reason about these algorithms directly, rather than hoping that the right behavior will emerge as a consequence of the interactions between objects. That supports the Agile agenda of working code– and goes further to usable code based on the end user model rather than on software engineering formalisms. The domain objects capture long-term stable system form that help the programmer contain change in the long term, supporting the Agile agenda of responding to change. De-coupling the algorithms from the data further supports responding to change. Furthermore, DCI reaches deep into such Lean principles as overall consistency, reduction of documentation, and just-in-time delivery. ■

Photo: clipart.com

REPLACING **ARCHITECTS** WITH ARCHITECTS

The centralized design and decision making associated with traditional software architects has, most righteously, been questioned by the agile community. It goes against the very core of Lean and Agile – empowerment. As a consequence, the role of the architect has been eliminated, but not just its traditional manifestation. This has in turn led to the devaluation of both architecture and, even more so, of software architects.

By Staffan Persson, System Architect Scania

The question should not be if we should get rid of the centralized design and planning, because that we should. Instead we ought to ask: what should we replace it with? The agile lack of response obviously provides no guidance.

The misconception of self-organizing teams

The underestimation of architects and architecture is in addition caused by another agile misconception; that of self-organizing teams. I usually refer to it as the agile myth of Apollo 13. Apollo 13 mission control, lead by Gean Kranz, successfully brought the flight crew members back to earth after an explosion which caused a loss of electrical power and failure of oxygen tanks. The mission control team solved numerous problems, acting as a self-organizing team rather »

Photo: Scania

Photo: istockphoto.com

"The Software Design boss"

Software Architecture is, generally speaking, assumed to have a very particular meaning. Wikipedia explains "The software architecture […] is the structure or structures of the system, which comprise software components […] and the relationships between them."

The architect, then "subdivides a complex application […] into smaller, more manageable pieces". The software architect is regarded as some form of software design boss, making all the important design decisions, communicating through software blueprints.

» than according to the traditional Command n' Control management paradigm. This "successful failure" is often used as an argument for the superiority of self-organizing teams. And yes, so far, it's all true.

But the conclusion within the agile community has often been: Just turn everything into self-organizing teams, and the rest will just magically sort it self out. The mistake here, is the assumption that self-organization is something that you just decide or choose. But, it's not. Self-organized teams are not created, they occur. And self-organization can be nothing more then anarchy, unless we create the right conditions, and provide the right leadership.

Creating the proper conditions

A real self-organized team will only occur when three prerequisites are met: a purpose (a common goal), ability (craftsmanship, or domain expertise) and a supportive culture (a culture that promotes involvement, responsibility and team spirit).

All these components are apparent in the Apollo 13 example: astronauts lost in space (defining a common team purpose), ability (highly skilled and experienced staff), a supportive culture (through management and team member mindset).

We have to create the environment and conditions that allow individuals and teams to succeed. And here lies the purpose of architecture, and of architects. It is not to control and structure, it is to empower. The purpose of architects is to provide leadership, a leadership necessary and complementary to traditional management.

Underestimated by the agile community

These aspects are not completely neglected by the agile community, but their importance is greatly underestimated. To me, this is one important reason to go "Lean" rather than "Agile"; in Lean, with its "management by means" philosophy and focus on mindset, attention is inevitably shifted from form and formulas, to people and underlying principles.

This view of the architect becomes especially important with the evolutionary approach to development given by the continuous improvement paradigm of Lean. As Gordon Pask puts it: "The role of architect […] is not so much to design a building or a city as to catalyse them: act that they may evolve". This summons up the "new" role of architects in a good way: being the enabler of evolution, act to create room for opportunity.

Architects have traditionally been taking responsibility away from developers. In Lean product development, architecting is a process in the opposite direction, handing power to developers. This process is not accidental, it has to be enforced.

Visualizing cross-functional dependencies

At Scania, as cross-functional architects, our objective is to remove cross-functional barriers, and facilitate cross-functional work. Our job is not primarily to make designs. It is to understand when design is needed, and ensure design is being made. To do this, we need to make cross-functional dependencies apparent, make them comprehensible and visualized, so that individual developers can understand their detailing of design from the perspective of the whole. The architect, in this perspective, is someone who understands the consequences of change, understands and explains dependencies, and involves the right people in defining the solution. For example, architects facilitate cross-functional workshops allowing architecture to be defined rather then defining it themselves.

'The purpose of architects is to provide leadership, a leadership complementary to the traditional management'

One consequence of this is that we need products and systems that can be grasped by "the many developers" (to paraphrase the IKEA founder Ingvar Kamprad). This, if anything, is a job for the cross-functional architects; how can we make the over-all system layout simple enough for every one to understand it, so that everyone can alter it? That, for sure, doesn't happen by chance. The architecture describes the system or product at hand, defines it. To a large extent, it sets the way we think about our product; it defines how we approach it. This can empower, but it can also be a straitjacket. In the end, the architecture defines our mindset.

Customer focus

How we divide the whole into parts control how efficiently we can organize and work in parallel. Interfaces within the product define interfaces within the organization. The purpose of a component sets the purpose of the team developing the component. Here, architecture can be used to do good. For example, by deconstructing the product primarily in customer oriented functions, rather than tiers and reusable components, we can put each development team in control of a complete development-flow, beginning and ending with the customer.

After the agile movement eliminated the architects, turning to Lean, the first thing we probably ought to do is to replace those architects with – architects. Just a very different kind. ■

Staffan Persson

Staffan Persson is system architect for software and electronics for Scania's vehicles. His main driving force is to apply Lean principles in software and product development.

UNTANGLING lean

'But since [the Agile Manifesto was created] I have been doing a lot of reading about Lean, and I can't see that we in the Agile community have added much of anything to what Toyota was already doing [...] It is sobering to realize they have been doing for decades stuff we think we invented.'

Alistair Cockburn, one of the founders of "The Agile Manifesto".

By Krister Kauppi, coach and consultant at Elicit

Lean and Agile both create good conditions for handling frequent problems in many IT projects – but in rather different ways. As the concepts are often confused, Krister Kauppi will help us untangle them.

Step 1: What is Lean?

Lean Software Development is Lean principles applied to software development, as originally described by Mary and Tom Poppendieck (2003).

Lean has its origins in the Japanese company Toyota. The aftermath of the Second World War was tough on the Japanese economy and in order to survive Toyota had to become highly effective with their scarce resources. By visiting a num-

Krister Kauppi

Krister is a software development coach and consultant at Elicit. His big passion is the search for productivity in both business and personal life.

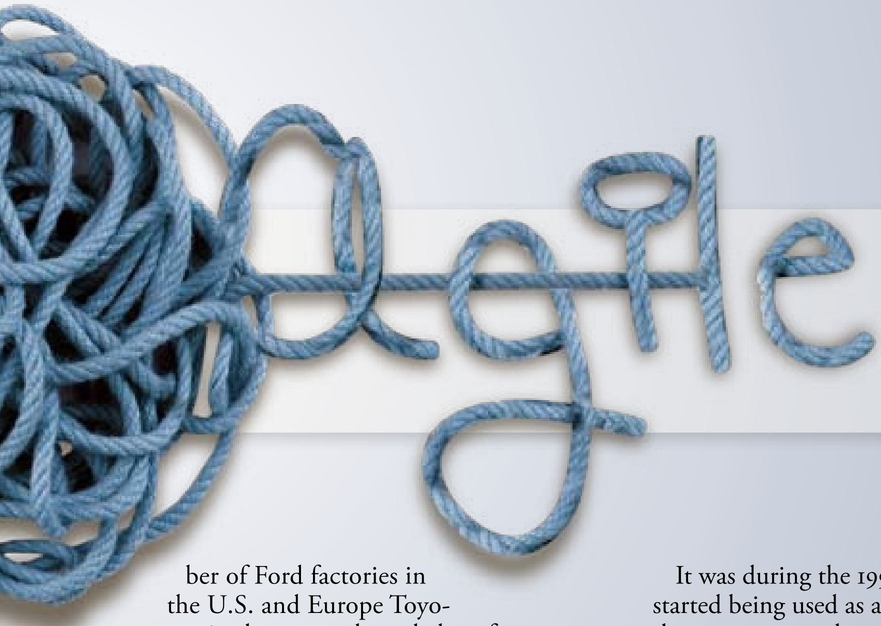
Lean vs. Agile – similarities and differences

Similarities

- Based on values and principles
- Not theoretical products but based on practical experience
- People-oriented
- Focus is on high business value
- Fast deliveries
- Continuous improvement

Differences

- Lean is based on a struggle for Toyota's survival. Agile, however, is based on a reaction to inefficient methods that were used in many IT projects.
- Lean is more about productivity and streamlining the workflow. Agile is more about adaptability.
- Lean is more scalable and has proven successful in processes other than just software development.
- Lean in software development has a wider range than Agile. Lean spans from the customer's vision of the system, to project approval, staffing, collecting of user requests, development, maintenance, support, training, informing etc. Agile on the other hand is more focused on the project phase.



'You don't do Agile or Lean you do Agile and Lean. The only question is how explicitly you use ideas that draw directly from Lean manufacturing.'
Martin Fowler, also a founder of "The Agile Manifesto"

ber of Ford factories in the U.S. and Europe Toyota received a greater knowledge of mass production of automobiles. However Toyota lacked the resources needed to fully apply Fords manufacturing process. Toyota streamlined this process to enable fast and cost-effective deliveries, satisfying the demands of a small and multi-faceted Japanese market.

Toyota achieved their goals by reducing capital binding inventory, production defects, overproduction and other kinds of waste that did not add value for the customer. This became the basis for the manufacturing process which was named Toyota Production System (TPS) and because of its high effectiveness it helped Toyota handle the oil crisis in the 1970s better than other companies around the world. As a result the interest in how Toyota managed the company grew and in the book 'The Machine That Changed the World' (1990) TPS was highlighted as a manufacturing process for resource-effective production. It was in this book that Lean was first used as a term describing Toyota's successful way of working.

Today Lean is adapted to most industries and regardless of which industry Lean is applied to, its core is to eliminate waste by continuous improvement and thereby effectively deliver what the customer really wants.

Step 2: What is Agile?

Agile is not a method in itself but a category for various agile methods.

It was during the 1990s that these methods started being used as a reaction to other methods that were process heavy and inefficient in handling uncertainty and change. The category was created in 2001 when a group of leading profiles from different agile methods met with the intent of finding a common ground for the agile methods. The outcome of the meeting led to "The Agile Manifesto", which addresses the values and principles on which Agile is based.

The core of Agile is to develop software more effectively and people-oriented in a changing and complex world. In most agile methods this is accomplished by frequent customer feedback, short iterations, flexible and emerging requirements, with the goal to achieve working software with high business value. Lean Software Development is an agile method because it is aligned with the Agile values and principles. Other commonly known agile methods are Scrum, Extreme Programming and Adaptive Software Development.

Step 3: Conclusion

The facts box to the left sums up the similarities and differences between Lean and Agile. Even if there are differences between Lean and Agile we can conclude that they are closely tied and complete each other. What connects them the most is that they are both based on radical improvement of traditional methods to achieve highly productive software development. This kind of productivity is ultimately about (more often) doing the right things the right way. ■

Thorough preparation creates AN EFFECTIVE TE

– this is how the coach motivates his or her team to achieve amazing results with real joy

The story of a group of software developers who formed a team, travelled to Denmark and found joy working on an agile software project.

In the spring of 2009 Softhouse was contacted by a Danish client looking for a development team with skills in web technology. The client develops electronic infrastucture for Denmark's public authorities. The assignment – in keeping with the times – was to make three mobile telephone services publicly available. One of the developers who took the train across the Öresund Bridge was Tina Lenshof, an M.Sc. in Electrical Engineering educated in Lund.

'I'd only recently arrived at the company, so this felt like a real challenge. Until then we'd been spread out as consultants at a number »

Photo: istockphoto.com

AM

‘Our coach helped us find joy in working in a team’

The Gefyon Fountain (Gefionspringvandet) is the largest monument in Copenhagen. It depicts an ancient legend about Gefyon, a goddess who turned her sons into oxens in order to carve out a piece of Sweden to form the Danish island Zealand.

'Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.'

'The best architectures, requirements and designs emerge from self-organizing teams.'

'At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.'

'The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.'

of different companies, and only a few in the team had actually worked together before. As far as I knew, we had quite different technical backgrounds, and not many in the group had thorough experience of agile working practices.'

Rock-solid team-building

Before the project got going, the team was brought together on home ground in Malmö by its coach Ola Sundin. The first task was to build up the team members' trust in one another. Exercises included everyone presenting themselves in a variety of ways. This helped people both getting to know each other and finding their own place in the group.

Just before the first sprint in the scrum project, everyone did a personality test. Put simply, each person had to choose among words to describe themselves. The test was designed in such a way as to emphasize each participant's positive attributes while still making clear – gently – what they were less good at.

'When you can recognize your deficiencies and admit them to others, you're well on the way to winning their confidence,' says Tina Lenshof.

The group also discussed various personality types and looked at the whole team's results. This allowed them to see which attributes were represented in the team and which ones they needed to cover between them.

'Ola Sundin, our coach, explained how a team develops and goes through different phases. He was careful to explain that it is natural for conflicts to arise when you're on the way to becoming a high-achieving team. He pointed out how important it is for team members to be open towards each other – and towards him in his role as coach – if conflicts arise.'

Daily contact with the client

Motivation needs more than a list of requirements, so Ola Sundin got every team member to formulate one or more personal goals, such as "to become a specialist within a technical field" or "to strengthen my leadership qualities". Later during the project he followed up how far team members had managed to fulfil their goals.

A further motivational factor for scrum team members was that they were directly responsible, together with the client, for formulating requirements and determining goals for all the sprints.

'Being able to take the initiative and get immediate feedback on design proposals made our team very effective,' says Tina Lenshof. 'We had daily access to the client if we needed to ask questions, which made it easier for us to fully understand the aims of the project. It also meant that we became even more engaged in the project.'

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The work with the client started with a "Sprint Zero" which aimed to present the team to the client and stakeholders, to set up the working environment and servers, and to produce a proof-of-concept in line with the project. The role of Product Owner was taken on by one of the client's project leaders, while the role of Scrum Master was shared between the team members. The scrum team was housed at the client's offices in a room dominated by linked desks where they sat and worked in concentrated silence, each with their own laptop.

In all, the project comprised 12 sprints of 1–2 weeks.

Essential feedback

Every morning the team held stand-up meetings to synchronise their work, to take new information on board, and to deal with any unplanned work which had emerged. Guiding

Photo: istockphoto.com

Daily stand-up meetings is a useful exercise in all types of endeavors

'Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale'

principles for the meetings were to keep them short and not to discuss solutions.

'We agreed that we would have a specific aim for each meeting and that we would close the meeting as soon as the aim had been achieved. This made sure that we saved both time and energy.'

The team was re-energized with a retrospective after every sprint. By sharing their experiences from the sprint about what worked well or badly, team members were able to communicate and share their expertise, and all were better prepared for the next sprint.

'Getting personal praise from other members strengthens your own positive attributes and increases your motivation,' says Tina Lenshof. 'The retrospective is also a sort of forum where you can raise problems, find ways of solving them, and sort out how to make sure they don't happen again. It's also a great opportunity to bring up worries for the future, so that the team and coach can manage them together.'

It emerged during the retrospectives that when team members worked alone on difficult assignments, they found it difficult to focus, and on other occasions they spent unnecessarily much time fixing details. The coach's solution to both problems is pair-programming.

'Working together has several benefits,' says Tina Lenshof. 'You maintain your focus on the task, you reach a solution more quickly, the solution is automatically reviewed and is often better. At the same time, it's an efficient way of spreading knowledge and skills across the whole team.'

Task completed – energy dissipated

With the Danish spring at its most beautiful in May, the team experienced a sudden dip in energy levels. It was hard to explain, as it happened immediately after several weeks of unexpectedly high productivity and satisfactory results. What had happened?

'We brought it up with our coach Ola Sundin at the next retrospective,' says Tina Lenshof. 'We eventually decided that the problem was that we had reached our goal too early – the task was completed before the sprint was over. Instead of finishing the sprint early, we tried to find new tasks, and this led to frustration. From then on we never extended a sprint just to suit the schedule.'

'When you make frequent deliveries, you keep your focus. You get an overview of what's needed to complete a delivery and can estimate how long

it's going to take to reach your goal. On the other hand, dragging tasks out just to make time pass decreases motivation and energy. By having a goal for every sprint, the team knows when to bring the sprint to an end and congratulate itself on a job well done.'

Something which also affected the team's energy levels positively was being given new challenges. As a result, they were determined to deliver fully working software. It saps the energy to have to repair something which used to work once upon a time, while new challenges increase energy. Occasionally the team had to go back and improve its deliverables. Taking time to reflect on solutions and rework the code gave increased satisfaction, since the team knew they were delivering an improved product.

The competitive instinct can be counter-productive

The team managed for long time to avoid conflict, until it was divided into two to develop two different solutions for the same product. This put a damper on the feeling of solidarity achieved earlier in the project. The competitive instinct took over, and the two groups started to rile each other. Instead of being a spur, the competition actually stifled creativity, and the only goal was who would get there first.

'Eventually we found ourselves in a situation where the client had to choose between two different solutions, though there was no way of deciding which solution was the best,' says Tina Lenshof. 'We got stuck in an emotional discussion where prestige was more important than technique.'

In the retrospective, the team came to the conclusion that when there's no way of measuring the quality of different solutions, it's better to discuss the different proposals' motivation at the outset, before development begins, and let the best motivation determine the outcome. Team members also felt their motivation shrink when they were compared with each other, and were prevented from working together.

'When we summarized the project we concluded that there were three factors that were vital to building up team spirit. Firstly, that we had access to a team coach thoroughly immersed in agile development and group dynamics. Secondly, that we were open towards each other. And finally, that we placed great importance on improving our working practices and following agile principles for a software project.' ■

Advertisement

KURSPROGRAM SOFTHOUSE EDUCATION

KURSER INOM LEAN OCH AGILE JANUARI–JUNI 2010

SOFTHOUSE education

KURSPROGRAM VÅREN 2010

Datum	Kurs	Ort	Språk	Lärare	Pris
13–14 jan	CSPO – Certified Scrum Product Owner	Malmö	Svenska	Arne Åhlander	1500 Euro
26–27 jan	CSM – Certified Scrum Master	Linköping	Svenska	Arne Åhlander	1500 Euro
10 feb	Kanban i praktiken	Stockholm	Svenska	Staffan Persson	600 Euro
16–17 feb	CSM – Certified Scrum Master	Malmö	Svenska	Arne Åhlander	1500 Euro
9–10 mar	CSM – Certified Scrum Master	Malmö	Svenska	Arne Åhlander	1500 Euro
17–18 mar	Lean & Agile for Managers	Malmö	Svenska	Anders Sixtensson	1500 Euro
13–14 apr	CSM – Certified Scrum Master	Göteborg	Svenska	Arne Åhlander	1500 Euro
4–5 maj	CSPO – Certified Scrum Product Owner	Göteborg	Svenska	Arne Åhlander	1500 Euro
9–10 jun	CSM – Certified Scrum Master	Malmö	English	Arne Åhlander, Jim Coplien	1650 Euro

Alla som blir certifierade Scrum Masters i våra öppna kurser erhåller Softhouses världsberömda Scrum Master Kit.

CSM – CERTIFIED SCRUM MASTER

Scrum Master Certifiering

Detta är Scrum Alliances standardkurs för att bli certifierad Scrum Master. Under kursen blandas teoretiska genomgångar och praktiska övningar med målet att deltagarna under kursens gång skall få uppleva känslan av att delta i ett Scrum-projekt. De enda förkunskaper som krävs är grundläggande erfarenhet av mjukvaruprojekt.

CSPO – CERTIFIED SCRUM PRODUCT OWNER

Product Owner Certifiering

CSPO är Scrum Alliances standardkurs för att bli certifierad Scrum Product Owner. Product Ownern är den roll som ansvarar för att prioritera kraven så att producerat värde blir högsta möjliga. Han/hon har som huvuduppgift att hantera vision och backlog för produkten. Huvudmål för kursen är att lära deltagarna att definiera en produktvision, skapa, prioritera och underhålla backlog och release-planering, samt sköta kommunikationen med projektet på bästa sätt.

KANBAN I PRAKTIKEN

I denna endagskurs lär sig deltagarna hur man använder Kanban för att synliggöra ett arbetsflöde och förbättra detta flöde. Kursen bygger på praktiska övningar blandat med diskussioner och frågor, varvat med korta teoripass. 'Kanban i praktiken' vänder sig till dig som arbetar som t.ex. produktansvarig, projektledare, teamledare eller annan beslutsfattare i en mjukvaruutvecklande organisation.

LEAN & AGILE FOR MANAGERS

Denna kurs är till för dig som är ledare inom en mjukvaruorganisation och vill lära dig de senaste metoderna och verktygen från Lean och Agile för att kunna effektivisera era utvecklingsprocesser. Kursen, som är unik i sitt slag i Sverige, är delad i fyra delar där huvudpunkterna är följande:

- Value stream mapping och minimering av waste (Muda) i din organisation
- Kanbansystem och hur du effektiviserar flödena i dina processer
- Agile Retrospectives för ständiga förbättringar (Kaizen)
- Hur du baserar dina prioriteringar på affärsvärde genom en strukturerad Backlog

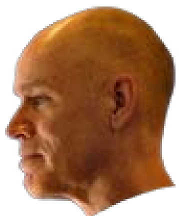
Aktuell information och anmälningsformulär finns på Softhouse Educations webbplats: http://softhouseeducation.com/kursprogram
Samtliga kurser i detta program kan hållas som företagsanpassade kurser med valfri tid och plats. Kontakta education@softhouse.se med din förfrågan, eller gå till Softhouse Educations hemsida: http://softhouseeducation.com/foretagsinterna-kurser

Coming soon: Jim Coplien's book on Lean Architecture

Jim Coplien and **Gertrud Bjørnvig** have finished the long-awaited **Lean Architecture for Agile Software Development**, and it's due to appear on bookshelves in May 2010. The book covers broad issues of software architecture ranging from problem definition and organizational responsibilities in taking architecture forward, to techniques for eliciting and developing user mental models and expectations, as well as the technology of domain-driven design and DCI architecture. It is not a theory book nor a philosophy book, but a book of advice that is broad, enabling, and concrete

Read more about the book at www.leansoftwarearchitecture.com

Ken Schwaber sets up a new Scrum organization: scrum.org



In the fall of 2009 it was spread in the news that Ken Schwaber, the co-founder of Scrum, resigned from his post as President and Chair of the Board of Directors of the Scrum Alliance. Later on we learnt that he had started a new Scrum organization called "Scrum.org".

Judging from agile blogs on the Internet, Ken's resignation stems from a disagreement over the definition of Scrum, which is kept in a document called the "Scrum Guide", which is originally written and maintained by Ken and the other Scrum co-founder Jeff Sutherland.

Ken Schwaber's comment to Lean Magazine when we asked about his plans with the new organization was:

'I founded Scrum.org to ensure the integrity of the Scrum process (stored on Scrum.org and maintained by Jeff and myself).

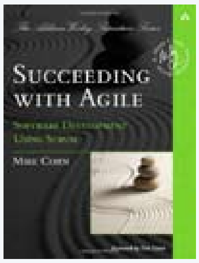
I also founded it to help people learn how to use Scrum better. We are working with organizations to produce and train with courseware that shows how to use their processes and products within the Scrum framework.'

Time will tell if this split into two organizations handling Scrum will lead to good or bad things, and we will probably come back to the matter in later issues.

Succeeding with Agile

Mike Cohn

The new book by **Mike Cohn**, **Succeeding with Agile Software Development Using Scrum**, is the answer to many prayers. Mike's book aims to assist all who struggle with implementing Scrum and Agile and has been at it for a while. The book is different from the basic books presenting values and principles of Scrum and Agile. It offers advice on aspects, problems and thoughts that arise after using Scrum for 6–12 months. The five parts of the book cover *Getting Started, Individuals, Teams, The Organization* and *Next Steps.* To me the chapters about *Overcoming Resistance*, *New Roles*, *Changed Roles*, *Leading a Self-Organizing Team*, *Scaling Scrum* and *Coexisting with Other Approaches* look like must-reads for many, and then there are 16 more chapters with interesting content to dwell on. It would not come as a surprise to me if we two years from now will consider *Succeeding with Agile* as a classic must-read.



Arne Åhlander

Read more about the book at www.succeedingwithagile.com

EXTRA! EXTRA!

New Scrum Certification exam

In the autumn of 2009 some new procedures have been introduced by Scrum Alliance for the Certification of Scrum Masters. Earlier it was enough to attend a two-day CSM course to receive a Scrum Master certificate. From October 1st 2009 you also need to pass an online exam within 90 days after you have attended the course.

The CSM online exam consists of 60 multiple-choice questions covering the areas of Scrum Basics, Meetings, Roles and Artifacts. The certificate is valid for two years. When this period ends you can recertify for another two years by passing a new online exam.

Short FAQ on the Scrum Alliance CSM online exam

Does the exam cost anything?
The cost for the exam is included in the course fee for the CSM course.

What about us who have been certified before October 2009?
You don't have to take any exam now, but your certificates will expire in 2011, and you will be requested to take the exam to recertify at that time.

Is there an exam for Certified Product Owner as well?
No, at the moment an exam is only required for Certified Scrum Masters.

Can you take a look at the exam or just take the exam to test your knowledge?
No, the exam is only open to those who have attended courses after October 2009, and those who will recertify after 2011.

Where can I learn more and get updated information?
On the Scrum Alliance website at: http://bit.ly/csm-faq

Through my gaming glasses

Games have been a big part of my life, both professionally and personally. When I was younger it was all about role playing games – high adventures where we fought dragons, rescued fair maidens and collected magical items in bulk. Today I play more board games since it requires a little less time, but I still like games where a great story is told!

Board games are a great way of bringing people together and get them to talk about other things than work which is exactly what you need after a hard iteration. If you choose your game carefully you can even get some interesting discussions about project processes and human nature out of it since there are a lot of similarities between a good game and a good process.

This is what I have come to value when it comes to games, no matter what type or genre it belongs to:

- First of all the game has to have a clear goal, if the players don't know the success criteria of the game it is very difficult to know how to play it.
- The second thing needed is the presence of choice and preferably some meaningful choices. A meaningful choice is a choice where I risk something to gain something else. If there are no risks and no rewards involved the choices become completely pointless.
- The third thing needed is visibility, the ability to see what consequences my choices have on the game. No one likes to play a game where you discover that you lost in the end simply because the success criteria were too complex.
- Finally the game should have a set of comprehensible rules. It should be possible to play the game without having to reference the rulebook all the time. If you have to spend a lot of time flipping through the rules or arguing about how the game works it rapidly becomes boring and frustrating.

One of my favorite games of all time is Blood Bowl, basically an ultraviolent form of American football in a fantasy setting. This is a game that brings out the worst in me and the only game that makes me go absolutely crazy when things go wrong – and they always do! One of the things that make Blood Bowl interesting is that if you play well your team players will gain experience and eventually new useful skills. Another thing is that if you fail with any of your actions your turn ends immediately, and you only have sixteen four-minute turns. You have to plan each round really carefully and fast – do the most important things first and minimize risk to avoid a premature ending of your turn as well as having a plan B since your opponent will spend his turn beating your team into pulp. If you are not careful you will eventually lose your valuable players and it hurts to get your players killed just because you did things in the wrong order.

'I tend to look at games through my Lean and Agile glasses all the time'

It doesn't take a rocket scientist to spot the references to Lean and Agile in the above section. I tend to look at games through my Lean and Agile glasses and the other way around of course. I think it is beneficial to look at any process or methodology where people are involved from a game perspective. Looking at Scrum, for example, through my gaming glasses I can see a lot of game mechanics at work – clear goals, meaningful choices, visibility and simple rules are all there. I try to live by the motto "what's fun gets done" and I believe that things become fun when your actions and decisions affect the outcome. The challenge as an Agile coach is simply to figure out what "fun" really means in the current context.

Ola Sundin

Agile Coach at Softhouse Consulting