A guide to

Agile AND Lean

and how organisations can become both

Clearvision

CONTENT

INTRODUCTION

Although agile software development has been around since the turn of the century, a surprising number of organisations are only just getting to grips with it.

In a 2019 report by Outsystems, 60% of the organisations interviewed said that they had invested in agile tools and services in the past year. However, the average agile maturity score was only 2.7 out of 5, which means most organisations are struggling to define and establish their agile processes.

Half the problem is a lack of understanding agile and lean, even amongst teams who practise them. Are you agile? Are you lean? Are you both? Can you be both? The other half of the problem is a lack of resources and talent available to help organisations with agile and lean adoption.

This white paper aims to help IT professionals better understand agile and lean concepts and provide a framework for initiating real change within their organisations.

THE EVOLUTION OF AGILE AND LEAN

In the earliest days of software, developing was chaotic and not well-planned. In the 1970s, a framework was created to bring some order. The framework equated software engineering with physical engineering and borrowed heavily from how construction projects are approached. All the major phases of the application development life cycle were to be clearly defined at the outset from requirements to deployment. Then each stage of development had to be completed in full before moving onto the next. This was known as the waterfall model.

It was termed "waterfall" because progress could only flow one way. Once a stage was complete, there were rarely provisions to go back to an earlier stage and improve something. However, construction projects such as bridges and skyscrapers don't normally require modification over a year or two, whereas software projects do. So, although the waterfall model brought a sense of organisation and engineering practice to software development, the inflexibility of it would soon prove problematic.

It was in the 1990s, when PC's spread to the enterprise, that an "application delivery lag" really became apparent. The time between a validated business need, and delivery of a working application was estimated to be about three years. But requirements, systems and even whole businesses can change in that time. As a result, many projects either didn't fulfil business needs on completion, or had to be cancelled part-way through.

These frustrations led to the evolution of various new software development methods deemed "lightweight" in comparison to the "heavyweight" waterfall model. In 2001, 17 software leaders met at a resort in Snowbird, Utah, to discuss these lightweight methods. This now-famous meeting led to the publication of the Manifesto for Agile Software Development. The manifesto proposed delivering working software in increments by way of a collaborative, adaptive and continually evolving process, instead of betting everything on a "big bang" launch.

Lean software development came a little later. Its roots are in lean manufacturing which attempts to make obvious what adds value through reducing everything that doesn't. Lean principles were translated to the software domain by Mary Poppendieck and Tom Poppendieck in their 2003 book, Lean Software Development: An Agile Toolkit. The connection between lean and agile was made at the outset, the Poppendiecks created a set of tools which they compared to corresponding agile practices. Becoming involved in the agile software development community and talking at several agile conferences has led to wider acceptance of lean within the agile sphere.

UNDERSTANDING THE DIFFERENCES BETWEEN LEAN AND AGILE

The fact agile and lean are both about helping teams deliver software more efficiently and effectively than the waterfall model means they're frequently used synonymously. Even teams practising them often don't have a clear understanding of the differences or similarities between the two.

The first thing to note is that neither agile nor lean are methodologies. They are a collection of principles upon which different methodologies are based, e.g. Scrum and Kanban. It would be wiser to refer to agile and lean as philosophies.

The second thing to note is that while there is a deep connection and overlap between them, agile and lean are not equivalent.

Agile is focused on delivering working software as quickly as possible, with frequent customer involvement.

Lean is focused on delivering value by reducing waste and improving operational efficiency.

When described like this, agile and lean sound very different. Agile is about speed and results, lean is about process efficiency. One could argue however, that becoming lean helps an organisation become agile.

Taking a closer look at the principles of each reveals the connections between them.

The Agile Manifesto values are:

1. Individuals and interactions over processes and tools

2. Working software over comprehensive documentation

3. Customer collaboration over contract negotiation

4. Responding to change over following a plan

ITS 12 PRINCIPLES ARE:

1. Customer satisfaction is the highest priority
2. Welcome changing requirements, even late in development
3. Deliver software frequently, in weeks rather than months
4. Developers and business people should cooperate daily
5. Build projects around motivated people and trust them to get the job done
6. Face-to-face conversation is the best form of communication
7. Working software is the primary measure of progress
8. Development pace should be sustainable
9. Continuous attention should be given to technical excellence
10. Simplicity is essential, i.e. maximising the amount of work not done
11. The best architectures, requirements and designs emerge from self-organising teams
12. Reflect regularly on how to become more effective and adapt accordingly.

LEAN'S SEVEN PRINCIPLES ARE:

1. **Eliminate waste**

Eliminate anything that does not add value to the customer, e.g. incomplete work, extra processes, task switching, rework and management activities. Only work on what absolutely needs doing in the here and now.

2. Deliver as fast as possible

Deliver working software quickly in increments to get user feedback. Incorporate this feedback into the next iteration. Keep iterations short for better learning and team communication.

3. Decide as late as possible

Keep options open until the last minute. The later critical decisions are taken, the more information is available to make them the right ones.

4. Create knowledge

Structure the work so that teams are continuously learning and sharing knowledge. This enhances their longer-term productivity and flexibility.

5. Build quality in
Instead of finding defects at the test stage, optimise processes to create fewer defects in the first place.

6. Empower the team
Respect that the people doing the work are the ones who know best how to do it. Create an environment that gives them what they need to be effective.

7. Optimise the whole
Don't focus on optimising individual developers just to make sure they're always working. That can be counterproductive. Step back, look at how the whole team is operating, and find ways of optimising the "system" rather than the "parts".

Looking closely at the principles of agile and lean, it's easy to see how they connect to one another. The lean principle, "Eliminate waste" is an extension of the "Simplicity is essential" principle in the Agile Manifesto; while agile states that teams should maximise work not done, lean pinpoints what work ought not to be done.

The lean principle about empowering teams rather than micromanaging them is directly related to the agile principle about finding motivated people and trusting them to do their own job. The traditional belief in most business contexts is that managers tell workers how to do their jobs. Lean and agile reverse this, teaching managers how to listen to the developers, enabling them to encourage progress, remove impediments and provide suggestions for improvement.

Lean principles — "deliver fast" and "decide late" are in effect process-oriented restatements of agile's principles about delivering working software frequently, responding readily to change and ensuring customer satisfaction. Lean focuses on workflow, minimum viable products, customer feedback and ensuring there is as little time as possible between the product owner stating their requirements and the team delivering the product.

When you break the two philosophies down into components, one could argue that lean complements agile by homing in on the processes that are necessary to make organisations more agile. Indeed, some commentators believe that the values and principles of agile work because of the science behind lean.

WHY IMPLEMENTING A SET OF PRACTICES DOESN'T MAKE YOU AGILE

Some development teams believe that they are agile because they do daily stand-up meetings (known among users of the Scrum framework as "daily Scrums"). Others believe they can achieve agility just by implementing a process such as Scrum, Extreme Programming or Crystal. Before long the cracks start to show and when expectations get missed and projects fail, teams start questioning whether "going agile" was really worth it. However, the real reason a project fails is that the team never properly understood agile to begin with.

A development team can't just whip out a handy list of agile practices, pick a couple that look easy to implement and think it's going to make them agile. Agile is set of values, not practices. While there are practices to help organisations achieve agility, those practices are not what makes them agile.

Agile is about the why, not the how, which is the reason gaining consensus on its principles is essential and anything else is jumping the gun. Teams need to establish why they want to become agile and what they hope to achieve with it. Once they've defined their intent, that's when they can select some practices that will help them. If organisations know what they want to get out of agile, they can evaluate their progress.

Lean proves useful here. Since it is more process-oriented, it provides a compass for measuring whether the practices selected are helping make the team more agile or taking them in the wrong direction.

For example, daily Scrums are intended to help the team, however if the team finds them a waste of time, the organisation needs to re-evaluate. It could be that the daily Scrums are being run incorrectly or that the other parts of the Scrum framework haven't been implemented, i.e. sprints, backlogs, definitions of "done". Until these issues are resolved, it's questionable whether the daily Scrums are providing value or just waste that, under lean principles, ought to be eliminated.

BUILDING AN AGILE-LEAN TRANSFORMATION INITIATIVE

As mentioned in the introduction, organisations are struggling to become agile, with OutSystems reporting a low 2.7 out of 5 agile maturity score. And while organisations continue to struggle to become agile, customers continue to wait for their software. Respondents to the OutSystems survey reported a 60% increase in the number of applications slated for delivery in 2019, while 46% reported the average time to deliver a web or mobile application to be five months or more. In addition, 64% said they had a backlog of apps in development and only 39% said that their backlog had improved over the last year.

Compounding a lack of understanding of the core principles is a lack of available talent to help organisations with agile and lean adoption. 75% of IT professionals interviewed by OutSystems believe that skilled developers are scarce.

For organisations having trouble defining their agile processes or not yet seeing the benefits of agile as expected, Clearvision and the Blended Agile Delivery (BAD) Toolkit can help them build an agile-lean transformation initiative.

The BAD Toolkit is an open-source set of tools and techniques built on agile and lean values. It is designed to help development teams better understand the principles of agile and lean and their practical applications. Consisting of an extensive and growing catalogue of downloadable, easy-to-use, visual "cheat sheets", the toolkit leverages the knowledge of thought leaders in the global agile community, and is the culmination of decades of experience.

Clearvision offers the full stack of Atlassian solutions for software developers, as well as tutorials, training and consultancy for businesses keen to realise the benefits of agile. Clearvision also provides organisations with access to over 2,000 expert Atlassian contractors, with built-in Atlassian platinum support, via its human cloud platform ClearHub.

In essence, The BAD Toolkit provides organisations with a strong agile and lean framework, while Clearvision provides them with everything they need to put it into practice.

Together they offer a powerful ecosystem of support and expertise for helping organisations make transformative changes to the way they deliver software to their customers.

CONCLUSION

Becoming agile results in more productive teams and faster delivery of projects, but only if everyone is on the same page. Too often companies attempt to go agile before they truly understand what it means. Simply implementing a process like Scrum doesn't make organisations agile. Establishing why they want to become agile and what they hope to achieve with it must happen first.

Once teams have gained consensus on the principles of agile, a great next step is to look at lean. Lean is a different but deeply connected set of principles that complements agile by concentrating on the processes necessary to achieve agility within an organisation. When it comes to implementing agile practices, lean provides a compass for measuring whether those practices are actually working.

It sounds like a complicated journey, which is why Clearvision and the Blended Agile Delivery (BAD) Toolkit are here to help make it simple. The BAD Toolkit provides organisations with a powerful agility framework while Clearvision provides them with solutions and expert talent to support it.