Five Principles of Lean-Agile Product Development

Introduction

A lot of teams and organisations are struggling to implement and improve lean-agile ways of working. The guidance and helpful tools such as SAFe (Scaled Agile Framework) competencies are “too detailed” and organizations need to make mighty decisions to implement SAFe in the first place. UK’s GDS (Government Digital Services) assessments are only for the Public Sector and they have limitations and limited capacity. Consequently, it’s been applied on a very small percentage of the government digital products and product delivery streams so far. Therefore, I wanted to put together a kind of lightweight competency/improvement checklist together to help Lean-Agile product development teams and organizations.

Principles of Lean-Agile Product Development

Although in recent years, Lean-Agile product development has become one of the major methodologies for software development, it hasn't always been the case. Introduction of the lean approach inspired by the Toyota Production System into western audiences can be traced back to late 80s and early 90s[1], it took another 10 years until the Agile Manifesto was released in 2001[2]. While the Lean approach was mainly seen as a revolutionary approach for specifically production of the goods, Agile was introduced as another revolutionary approach for software development. We need to wait for another ten years to see these two being used together.

Agile software development was born as a response to the waterfall project management approach in software development. In fact the waterfall project management approaches were born mainly in the Construction Industry in the first half of the 20th century[3]. One of the main challenges around applying this methodology to software development was around planning. For example, in the construction industry, a project is planned in very fine details at the beginning of the project. After that it’s expected that there would be very little changes in the plan during the development. And any diversion from the original plan should be also managed via a strict control mechanism to make sure that it’s still viable. This might work for small software projects limited within a couple of months. But a lot of software projects span a period of time the technology as well as user needs and perceptions can change. In addition to this, it’s very unlikely that we can capture all “user needs” at the very beginning of the

[1] Taiichi Ohno’s Toyota Production System was published in English in 1988. The principles of Lean were first introduced to the western audience in the book The Machine That Changed the World (1991) by James P. Womack, Daniel T. Jones and Daniel Roos
[2] https://agilemanifesto.org/
[3] https://home.gwu.edu/~kwak/PM_History.pdf

product development before even testing it via our prototypes and early release of the software. Therefore, it's important for a successful software project to respond to the change effectively. Agile was born as an answer for managing the change.

The Lean on the other hand was born in the automotive industry in Japan in the 50s after war economic conditions[4]. Toyota wanted to survive first then compete with American automotive giants in an environment where resources were limited. They developed a system where they can cut the waste where possible and focused on the value that was identified by customers. Cutting waste in the form of stock, wasteful work and delay by making sure that they have continuous flow became major objectives of the production. You can only cut the waste if your product is loved and used by your customer. if your end product is not being bought and used, otherwise all other activities around cutting waste would be useless. Therefore, on-time production as well as responding to ever changing customer needs have been also major principles for Lean product development.

We had to wait another ten years to see these two incorporated together. In 2011, Scaled Agile Framework was released by Dean Leffingwell. In the UK the Government launched Digital Service Standards and Service Manual Beta in March, 2013[5]. In 2016, Jeff Gothelf and Josh Seiden published Lean UX and introduced Lean UX Framework. The marriage of the lean and agile frameworks made a positive impact on scaled agile product development. Now it's more possible and effective to respond to changes in an agile way as well as staying user centred and cutting wasteful activities and processes at scale.

However, this short history of Lean-Agile brings some difficulties alongside. Because the successful lean-agile implementation (especially if it's at scale) requires a cultural shift and cultural changes require time. A lot of large organizations including public organizations are struggling to implement the lean-agile frameworks successfully. It's also hard to recruit experienced professionals in these frameworks. Therefore, a new employee without lean-agile experience and mindset might cause disruptions regardless whether it's in the team level or senior management level. I also experienced that some organizations have the program or higher level "command and control" approach whereas hoping that in the team level things can work in the Lean-Agile way. Therefore, it's important for a team and organization to continuously check the improvement against defined improvement areas under lean-agile principles.

PRINCIPLES AND IMPROVEMENT AREAS

Lean-agile product delivery requires a user-centred approach which is organized around the value creating activities and continuous flow of the product delivery. Aligned with this very brief statement, I tried to put some principles and defined improvement areas under the principles below. I believe principles should be taken as the minimum requirement for a

[4] https://www.lean.org/explore-lean/a-brief-history-of-lean/
[5] https://gds.blog.gov.uk/story-2013/

lean-agile implementation. However, **the improvement items as listed under the principles are indicative and they might vary from organization to organization and team to team**.

1- FOCUS ON USER NEEDS

A product or service can only be useful if it satisfies the needs of the users. *"Value can only be defined by the ultimate customer. And it's only meaningful when expressed in terms of a specific product (a good or service, and often both at once) which meets the customer's needs at a specific time."* [6]

Therefore, it's important to implement the right tools and techniques to continuously evaluate user needs in every stage of the product development.

- Capture user requirements via user stories rather than requirement lists or tasks
- Use prototypes to get user feedback before any development wherever possible
- Use user research tools and techniques[7]
- Make sure that user research is an organic and continuous part of Agile iterations
- Make user research a team activity

2- ORGANIZE AS EMPOWERED AND CROSS-FUNCTIONAL AGILE TEAMS

Agile team is a cross-functional, multidisciplinary group of individuals who can design, build, test, and deploy a small batch of the value creating artifacts in a committed, brief timebox. Agile teams should have authority and accountability to manage their own work.

An agile team consists of different roles such as Product Owner, Scrum Master/Delivery Manager, UX Designer, Developer etc. However, individuals in the team should aim to be T-Shape professionals who align and collaborate within the team and wider organization.

- Choose the agile framework, rituals and artefacts to fit the team preference and the organisation's dynamic
- Align with the organization, program and product vision and objectives
- Keep improving your product/service and way of working
- Learn from each other to strengthen capabilities in the team
- Work transparent and open to your teammates, stakeholders and wider organization
- Don't stop asking question

3- DELIVER ITERATIVELY

Iterative delivery allows the team to spot problems early and resolve them quickly. Early and often release approach improves the feedback loop and learning from users. Don't afraid of failing, "fail fast, learn quickly" and prevent creating big-bang releases ("too big to fail")

[6] Lean Thinking; James P. Womack, Daniel T. Jones; Simon & Schuster, 2003; page: 16
[7] For more info: User research – Service Manual – GOV.UK (www.gov.uk)

- Focus on delivering value to the user after each iterations
- Demonstrate the value to stakeholders via regular demos
- Identify important metrics and measure the change
- Use the Staged Agile Delivery approach to maximize the learning while reducing the risks (E.g., Discovery, Alpha, Beta, Live)[8]

4- PLAN CONTINUOUSLY

In lean-agile product delivery, the plan continuously evolves based on continuous learning from users and responding to ever changing product and program priorities.

- Organize a form of regular Program Increment Planning activities to create quarterly Team Roadmap (for 3-5 iterations)[9]
- Use an agreed and transparent prioritization method based on User and Business Value and Cost of Delay.
- Frequently update the Team Roadmap
- Review regularly to make sure that the Product Roadmap aligns with:
  - The Program Roadmap for each quarter (3-9 months)
  - The Portfolio Roadmap for mid and long-term product portfolio (6 months – 2 years)
- Use an objective measurement for capacity planning and progress (E.g., velocity, throughput and burndown charts etc.)

5- CONTINUOUS DELIVERY AND BUILT-IN QUALITY

All agile teams must define their own built-in quality practices aligned with the continuous delivery approach.

- Avoid overhead and delays associated with phase-gate and BUFD- methods
- Avoid code-branching, apply continuous integration and deployment principles
- Make all new source code open and reusable
- Test iteratively (User Story and Product Increment Level) and avoid big-bang test approach
- Use test automation from the start (if applicable)
- Continuously modify and improve the system (Refactoring)
- Make sure that data security measures are in place. Always respect the user data and privacy

[8] For more info Agile delivery – Service Manual – GOV.UK (www.gov.uk)
[9] https://scaledagileframework.com/pi-planning/