Adoption of Agile Methodology in Software Development

D. Duka
Ericsson Nikola Tesla, Split, Croatia
e-mail: denis.duka@ericsson.com

Abstract - As adopting Agile software development becomes a trend, there is a need for a more structured definition of what is Agile and what is a high-level of Agile maturity. Traditional development methodologies rely on documents to record and pass on knowledge from one specialist to the next. Feedback cycles are, in many cases, too long or even nonexistent. Agile principles emphasize building working software that people can get hands on quickly, versus spending a lot of time writing specifications up front. Agile development focuses on cross-functional teams empowered to make decisions, versus big hierarchies and splitting by function. It also focuses on rapid iteration, with continuous customer input along the way. This paper deals with Agile methodology and scaling. The special highlight is put on people investigating their contribution in Agile approach success. Some reflections after using Agile in our own organization are also presented.

I. INTRODUCTION

Traditionally the Telecom business has been standardization-driven and regulated on a national level. The development lead times have been long. Consequently the Telecom vendors have developed capabilities to influence standards, develop products and interact with the Telecom operators in a slow-moving industry. The business landscape has changed leaving the major slow-moving vendors to struggle with the pace of the newcomers [1].

The slow-moving market forces us to develop our ability to run major multi-year projects. We became predictable development machinery with extensive mechanisms to ensure predictability and control on the expense of flexibility and customer closeness.

This, in turn, led to organizational setups focusing on the alignment with the project structures and deepening the competencies in narrow areas both in the product and the functional dimensions. The result was organizational silos with multiple related hand-over challenges.

This gradually led to possessing less and less people with a broad systems understanding, which in turn slowed down both the organizational capability to handle broad technical challenges and the individual opportunities to learn and broaden the contribution. The feedback loops within the product development became longer and longer [2].

Only recently the Lean principles and the Agile development philosophy has been recognized as the source for solutions to these challenges.

Paper starts with describing Lean, Agile and Scrum methodology highlighting the difference between traditional *waterfall* and new approach in software development. The third chapter describes some main Agile characteristics. Scaling the Agile architecture and four proposed approaches are also presented here. The final chapter deals with real experiences following the Agile implementation in Ericsson underlining some observed changes and used practices while applying this new way of working. Some concrete measurements following the new concept are also presented.

II. ABOUT LEAN, AGILE AND SCRUM

The classical way to organize software development of large systems is the *waterfall* model, where the development starts from defining and analyzing the requirements and ends to operating (or maintaining) the software. Actual coding is only a minor part of the entire process, whereas there is much emphasis on defining, designing, documentation, testing and operating (maintaining) the software system [3].

Figure 1 [4] illustrates the waterfall software development model.

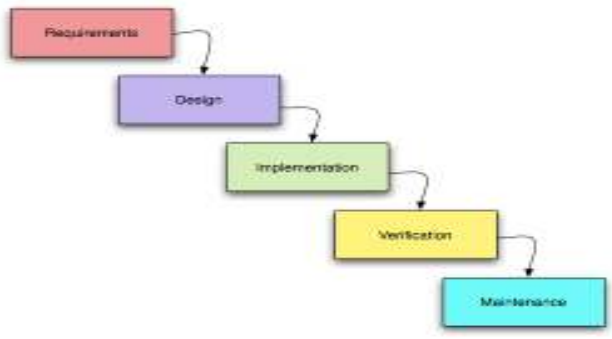

Figure 1. Waterfall Software Development Model

Agile software development model based on short iterations and quick releases challenges the *waterfall* models having emphasis on design and documentation.

The Agile manifest [5] recaps the ideology:

- Individuals and interactions over processes and tools,
- Working software over comprehensive documentation,
- Customer collaboration over contract negotiation,
- Responding to change over following a plan.

Lean manufacturing, initially called Just-In-Time production (JIT) was originally developed by Toyota in the latter part of the 20th century. Lean manufacturing system is built on several principles and philosophies. These include minimization of waste (through pull-production), Kaizen (continuous improvement), getting quality right from the beginning (stop production for fixing), among others. The Lean principles have been applied to Agile software development, and quite often are referred as Agile/Lean. Poppendieck Mary and Tom have published several books on the principles [6].

Agile and Lean are relatively broad concepts. There are several more detailed software development models, the developers of which call them as Agile methods. Some can also be considered Lean. These models include Scrum, Extreme Programming (XP) and Agile Unified Process (AUP), among others.

Stober [7] describes Scrum as a drastic simplification of project management containing three roles, three documents and three meetings. As can be seen from Figure 2, in a Scrum software development project, Product Owner (PO) decides the product backlog, i.e. the features expected from the software (for the next release), signs off all deliverables and represents budget and interests of the stakeholders. At the start of each sprint, the project team decides in a sprint planning meeting, which items from the product backlog are taken to the sprint backlog as use cases for the software. This is based on the prioritization by the Product Owner and teams work estimates and commitments. During each sprint, which duration is typically two weeks, the team completes the sprint backlog. A daily short Scrum meeting is held to follow-up the ongoing work and solve rising issues. Scrum Master (ScM) facilitates issue solving outside the meetings. At the end of each sprint a sprint review meeting is held to review the sprint results. Each sprint results in a working increment of the software product. Sprints are repeated until the product backlog is depleted and the software release is ready. Releases again are repeated for major software updates.

In this paper I refer principally to Scrum model, when discussing Agile, as it constitutes a clear and tangible reference model.

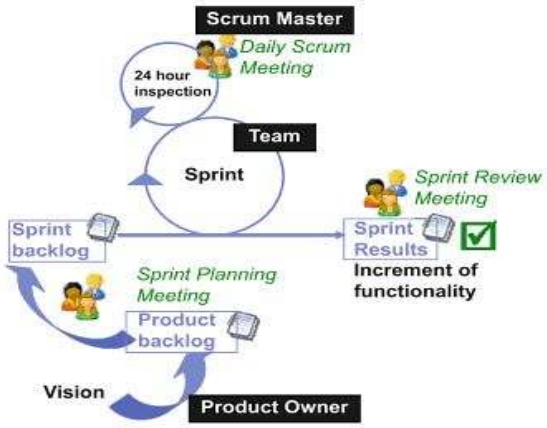

Figure 2. Scrum overview

III. AN AGILE APPROACH

The Agile approach aims to nurture organization assets and to support other groups, such as development teams, within organization. These groups should act in an Agile manner that reflects the expectations of their customers and the ways in which their customers work [8].

First and foremost, the Agile values, principles and practices should help to guide organization architecture modeling and documentation efforts. This is just a good start though. In order to be successful at organization architecture you need to rethink your overall approach and address some fundamental issues. These issues are connected in a synergistic manner; you must address all of them otherwise you will put your effort at risk. Some of these issues are:

- Focus on people, not technology or techniques:
  All of the arguments over "which model is right", "which notation is right", and "which paradigm is right" are meaningless if you don't have a viable strategy for working together effectively. You could create a perfect organization architecture model but it doesn't matter if project teams can't or won't take advantage of it.
- Keep it simple:
  A critical concept is that organization architecture models and documents just need to be good enough, they don't need to be perfect. By keeping organization architecture artifacts simple you increase the chances that your audience will understand them, that project teams will actually read them, and that you will be able to keep them up to date over time. Overly detailed documents might look impressive sitting on a shelf, but a simple model that project teams actually use is what your true goal should be.
- Work iteratively and incrementally:

Agile organization architects work in an iterative and incremental manner. They don't try to create complete models. Instead they create models that are just good enough. When they find that their models are not sufficient they work on them some more. The advantage of this approach is that they evolve their models incrementally over time, effectively taking a just-in-time (JIT) model storming approach that enables them to get the models in the hands of their customers as quickly as possible.

On large Agile teams, geographically distributed Agile teams, or for organization-wide architectural efforts, an architecture owner team will be required. Agile teams at scale are organized into smaller sub-teams. The architecture owner on each sub-team is a member of architecture owner team, which helps to increase the chance that each sub-team understands and follows the overall architecture as well as increases the chance that the overall architecture strategy will address the full needs of the overall solution. There will be an overall architecture owner, this could be a rotating role, who is responsible for facilitating the group [9].

There are four basic strategies for organizing Agile teams at scale:

- Architecture-driven approach:
  With this strategy you organize your sub-teams around the subsystems/components called out in your architecture. This strategy works well when your architecture is of high quality and the interfaces to the subsystems have been identified before the sub-teams really get going. Although the interfaces will evolve over time, they are needed to get a good start at them initially. The challenge with this strategy is that it requires your requirements to be captured in a way which reflects the architecture. For example, if your architecture is based on large-scale business domain components then a requirement should strive to focus on a single business domain if possible. If your architecture is based on technical tiers then requirements should focus on a single tier if possible.
- Feature-driven approach:
  With this strategy each sub-team implements a feature at a time, a feature being a meaningful chunk of functionality to your stakeholders. This strategy should be applied in situation where the architecture exhibits a lot of coupling and where you have sophisticated development practices in place. The challenge with this approach is that the sub-teams often need to access a wide range of the source code to implement the feature and thereby run the risk of collisions with other sub-teams. As a result, these teams require sophisticated change management, continuous integration, and potentially even parallel independent testing strategies in place.
- Open source approach:
  With this strategy one or more subsystems/components are developed in an open source manner, even if it is for a single organization. This strategy is typically used for subsystems/components which are extensively reused by many teams, for example a security framework, and which must evolve quickly to meet the changing needs of the other systems accessing/using them. This strategy requires you to adopt tools and processes which support open source approaches.
- Combinations thereof:
  Most Agile teams at scale will combine the previous three strategies as appropriate.

IV. IMPLEMENTING AGILE

Recently, our R&D management team started discussing how to adapt the way-of-working in our Unit to the increasingly turbulent business environment. Prediction of long-term market development, which had been the basis of our operations and requirement setting, was getting more and more difficult. Looking into the future, it became clear that we needed to do a proactive change in order to more flexibly react to customer wishes. After months of discussions, Agile and Lean principles were chosen as the guiding principles of our way forward.

In the beginning, we became aware of the concept of Agile in large scale telecom industry. It was challenging to realize in advance how we could change ourselves from a component-based technically-thinking organization into a customer-focused cross-functional and value-thinking organization [10].

Even before the beginning of the change journey we could identify multiple obstacles in front of us. However, the expected advantages were greater. Soon we realized that if we want to stay as a serious player in the fierce telecom industry, we need to start the journey towards becoming an Agile software company.

A. Less detailed level planning

We used approximately half a year to internally spread the knowledge of Agile software development methods. The change was complex and thus we were not able to justify the change by utilizing the normal business case study procedure.

We wanted to have a detailed plan of the one year change project. We wanted to understand and solve all the major potential problems beforehand. We wanted to mold this plan into our yearly strategic planning process and have each organizational silo responsible for part of the change. The coaches changed our approach more to "start implementing with a small scale and tackle the problems as they occur".

We started with one cross-functional development team and one Product Owner that had a task to implement a real customer feature into our product. In the beginning, it was a constant struggle to grant room for the experiment and, at the same time, keep the customer promises for the ongoing project. It was clearly two extremely different worlds that met. One world that we knew how it worked, that was predictable and going on with a slow but steady tempo. Then the new world with a greater tempo in which all current weekly and monthly meeting routines felt as road blockers on the way. The previous organization was not ready to react fast enough with the new requirements that two-week Scrum sprints brought to surface. Early on the team realized that our development environment was outdated and not built for making fast and small end-to-end deliveries. The team was encouraged to experiment with new tools and find better means that would support the faster feedback in our R&D organization.

Here are the steps that we have taken during the journey:

- One team, one Product Owner and one Scrum Master,
- Three teams (one off-site), one Product Owner and three Scrum Masters,
- Customer Support Request (CSR) and Fault Handling (FH) Kanban teams with Product Owners and Team Coaches,
- Building the continuous integration machinery,
- More new teams with a Scrum Master and a Product Owner,
- Feature and System integration in close co-operation,
- All development teams are cross-functional, self-organized and stable,
- Feature feasibility study done in co-operation with the lead feature team,
- The Release verification is in close co-operation with the feature development.

B. Intense learning & competence build up

Changing people mindset starts from small and visible actions that are repeated everyday. Management created a guideline to make clear what we want more in our everyday working environment. We believe that fundamental changes that are needed in our minds to succeed with this journey are as follows:

- More people initiative and less top-down control,
- More team players and less individual heroes,
- More courage and less risk avoidance,
- More conversations and less one-way communication,
- More personal growth and less comfort zone.

The teams are expected to learn in two dimensions. One is the functional dimension, i.e. system-design-testing. The other is the product dimension, i.e. different components in the system. They need to learn the tiny end-to-end functionality that goes through multiple components. This new dimension of learning needs different attitude than before.

Combining system, design and test competences into one team gave the possibility to share knowledge between the team members. Visualizing the task both in the Kanban board and the sprint planning board was found to be a valuable tool for the team coach to help the team members to broaden their competences.

C. Innovation

Change from the silo-based R&D center towards the Agile development R&D center included intensive learning, facing new challenges and risk taking which are also typical parameters of innovations. The elements in Lean & Agile and our innovation environment are complementing each others.

We have seen that by bringing people around the same table from the different backgrounds, it generates a spin of positive, innovative energy. Focal point of the current Agile software development is efficient software development in Agile teams. Hence, the majority of the creativity is naturally directed towards the improvements around the ongoing work in the sprints.

D. Involving the whole organization

From the beginning, we chose to be as open as possible during the planning and transformation. We realized that if we want value workers to actively contribute we need to involve them in the decision making and show all information openly. We had all meeting notes and workshop material visible in the intranet.

We introduced the “current best thinking“ to emphasize that we plan only short period ahead and want everybody to participate and look for alternatives.

Once a week we had an open question session called *Fast Forward Friday*. The sessions were found extremely valuable. Everybody had a chance to ask questions and grow their understanding of what is expected in the future. The level of questions told to the management team what the maturity of the rest of the organization was.

We utilized also anonymous web polls to gather individuals’ opinions. The free text answers also gave valuable information regarding where to put effort while driving the change into the proper direction.

E. Making the change visible

Already some years back we had the understanding that working according the strict organizational silo-

responsibility and budget-areas will limit the whole organization to improve. Therefore, we intended to introduce a new organizational setup. First, we started to break down the organizational silos by introducing flow based *pipes* that consist of several organizational resource pools. For instance, *the execution pipe* had resources from system, development and verification department.

The other issue was seating arrangement. We had a long history of everybody having their own 8 m^2 offices where one could close the door while wanting to concentrate in privacy. Therefore, it was understandable that the resistance for the new seating arrangement was substantial. Our management and also the project management team had shared an open space area for a year. Their positive experiences of how much easier it is to communicate in an open space area was supporting the move.

F. Agile practices used

While introducing Agile we have followed the basic Scrum and tried out how it fits into our environment without judging something that we have not tried.

In product maintenance, where the nature of work is a constant flow of customer service requests, we have utilized Kanban combined with practices from the Scrum framework. For instance, since the retrospectives are seen essential while experimenting the new way of working, we added that into the Kanban method.

Behind the decision of using Scrum there was a profound discussion and consideration. As a large R&D center developing a complex product, we needed to have one common approach for the teams. At Ericsson we have a history of tailor-making own tools and methods. This time the fact was that we did not have enough experience or competence to tailor a suitable process. It would have required too much time to develop an Ericsson-specific Agile process.

G. Agile experiences

Following the new methodology an assessment was performed in our organization. Overall 8% improvement is seen across all nodes put together. Results are presented through the following graph:

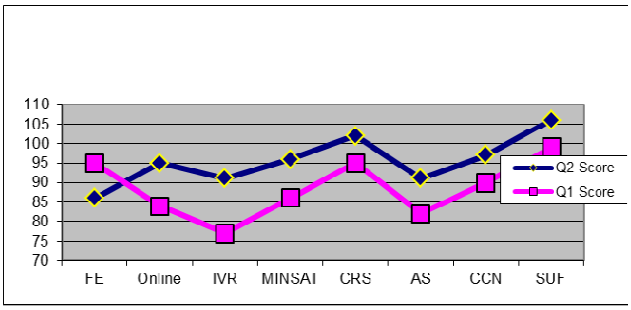

Figure 3. Agile Q1 & Q2 Assessment Scores Trend

V. CONCLUSION

Successfully implementing organizational wide change requires extra time and high energy level from the whole organization.

By using this approach in software design we are able to first recognize and then eliminate activities which don't add value e.g. partially done work without any guarantee if customer will take it in use, unnecessary task swapping, waiting, handoffs, faults, etc.

This way of working also implies improved responsiveness through rapid deliveries allowing customers to delay decisions. This is especially enabled by short feedback loops and continuous integration.

Our motivation behind the change was not only the current product under development but also to increase the level of our R&D center ability. First-hand experience of Scrum framework is the real source of learning. Only through experiencing it is possible to express real meaning of how well the new way of working fits into our situation.

Preliminary evidence we've collected from an assessment, but also from other sources all tell us that Agile software development is making a positive difference in our organization. In addition, interest and respect for Agile practices is constantly growing. Other parts of the company are interested in what we're doing and how we're making it work. Consequently, our future work will target other software teams who are adopting Agile practices and who need our guidance and support. Following such approach we'll be able to improve our pratices used and also to spread Agile culture within and outside our organization.

REFERENCES

[1] J. Astemark, " Agile@ITTE", Internal Ericsson Documentation, Sweden 2011

[2] D. Duka, "Agile approach in Software Development", Proceedings of the 35th Mipro CTI conference, 2012

[3] J. Sääskilahti, J. Röning "Challenges with Software Security on Agile Software development, Internal Ericsson Documentation, Sweden 2011

[4] R. Winston, "Managing The Development of Large Software Systems", Proceedings, IEEE WESCON, August 1970, pages 1-9. http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

[5] K. Beck, "Manifesto for Agile Software Development", 2001 http://Agilemanifesto.org

[6] M. and T. Poppendieck, "Lean Software Development Principles", http://www.poppendieck.com/

[7] T. Stober, U. Hansmann. "Overview of Agile Software Development", Agile Software Development: Best Practices for Large Software Development Projects, Springer. 2010.

[8] S. W. Ambler, "Agile Enterprise Architecture", http://www.Agiledata.org

[9] S. W. Ambler, "Agile Architecture: Strategies for Scaling Agile development", http://www. Agilemodeling.com.org

[10] K. Mikkonen, "How We Learned to Stop Worrying and Live with the Uncertainties", Internal Ericsson Documentation, Sweden 2011