Beyond Agile Methodologies: a Conceptual Analysis for Software Process Pipeline in the Industry 4.0

Lapo Chirici
Dept. of Digital Technologies
Istituto Europeo di Design (IED)
Florence, Italy
lapochirici@ied.edu

Kesheng Wang
Dept. of Production and Quality Engineering
Norwegian University of Science and Technology
Trondheim, Norway
kesheng.wang@ntnu.no

***Abstract*— Accelerating software development schedules is a paramount activity in the continuously evolving digital scenario of industry 4.0. Software companies are competing to bring more efficient processes to improve quality and productivity in the pipeline. In order to accomplish this, the entire segment have been considering the advantages released by the adoption of agile methodologies, although not without several adjustments. In facts, in the last five years, a significant shift that brought both managers and technicians to move from traditional Waterfall model to iterative agile/scrum methods, has been registered. In this context, mapping the value stream results crucial, since it contributes to identify, eliminate and reduce all those non-value-add activities during the cycle.**

Keywords – agile; lean methodology; scrum; software pipeline

I. Introduction

In today's global economy, software industry is one of the most strategic actor, since it creates new opportunities for business change by providing cutting-edge solutions. The resulting transformation in the environment places large demands on software development capabilities, productivity and quality. With the Industry 4.0, all data about operations processes, process efficiency and quality management, as well as operations planning are available in real-time. In this evolving environment, technological advances, increased globalization and other competitive needs are influencing the way the software companies manage their development projects. The traditional waterfall model is the dominant project management paradigm. Agile methodology is direct response to waterfall model, and lean is recent one. But effectively, when software companies attempt to adopt agile/scrum methods, often the developers team is unwilling to alter their routine in terms of way of thinking. This mismatch between processes within the pipeline makes the team very difficult to accept the introduced approach and at the end regress to their former process, not without consequences. On the other hand, Kanban system offers an opportunity to build on what is already working and gradually makes the transformation. And the Value Stream Mapping (VSM) helps to bring in significant improvement by identifying and eliminating those non-value-added activities during software development.

II. Lean Anthology

Initially, Lean movement was born in Japan in the mid 1950-s in manufacturing industry (automotive industry) and was mainly aimed at loss reduction and sustainable production. In 2000s, Lean was also adapted for software development by Mary and Tom Poppendiecks who related it with 7 initial Lean principles and Agile philosophy.
Following the trend that Lean could be extended to any industry, Lean was applied in the startup industry in 2008 by Eric Reis as a way of developing “new products and services in circumstances of extreme uncertainty.” A typical lean company follows a learn – measure – build cycle, and conducts many tests, frequently connects with customers, understands their value and focuses its key processes to continuously improve it. A never ending cycle leads the startup to sustainability, smart development and success.
Lean software development method is based on the principle "Just in time production". It aims at increasing speed of software development and decreasing cost. Lean development can be summarized in seven steps:

- Eliminating Waste
- Amplifying learning
- Defer commitment (deciding as late as possible)
- Early delivery
- Empowering the team
- Building Integrity
- Optimize the whole

A. *Differences between Manufacturing*

As conceivable, software development is not manufacturing. In manufacturing the goal is the production of physical parts and products. Time frames are short, the work is linear, tangible and repeatable, and the scope is more constrained with limited communications channels. This is not the primary paradigm of software development. Treating software development as Lean Manufacturing leads down a path of optimizing the mechanics of developing software, which can yield limited benefits but fails to address the much larger software product development issues. Moreover, the software development process has more uncertainties and complexities as compared to manufacturing. The result consists about the creation of an intellectual property, which is highly dependent on software professionals' innovative thinking, creativity and efficiency [1]. Moreover, the software development environment is geographically distributed, that means multi-location teams with different time zones. The single operator has to respond quickly to rapidly changing customer requirements (between 30 and 50 percent of all features are not necessary and add overhead). The team members are assigned to multiple projects and often overloaded [2]. In this scenario, not-disruptive and mixed approaches offer an opportunity to start out with incremental and evolutionary transformations by uncovering the non-value-added activities and recognizing the bottlenecks [3].

B. From sequential Waterfall to iterative methods

Currently, two models are dominating the software development activities: waterfall model and agile/scrum model. Waterfall model is sequential development model; it is the predominant in the software development community. Agile/Scrum model is iterative development model, and - according to a survey released by Forrester – in 2010 only about 35% of software majors use frequently agile and lean principles [4].

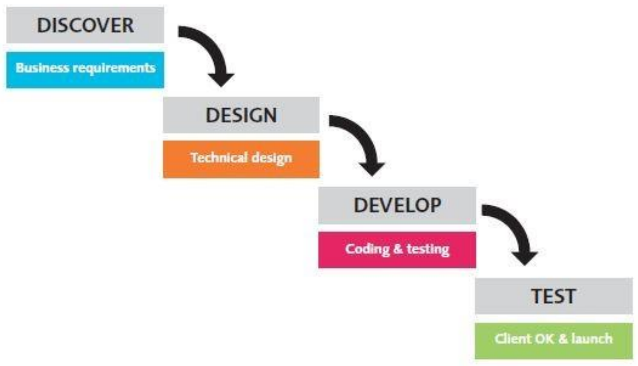

Fig. 1. *The Waterfall Software Development Model*

Here, 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. The major problem with the waterfall model is its inflexibility, since it is inefficient in responding to changing customer requirements. In Waterfall model, testing normally takes place only after coding has been completed. Nowadays it's hard to apply because, since the customer requirements and priorities are rapidly changing.

C. Towards Lean: the "V-Model"

An alternative way to address some of the issues, is represented by the V- Model, an extension of Waterflow that is based on association of a testing phase for each corresponding development stage.

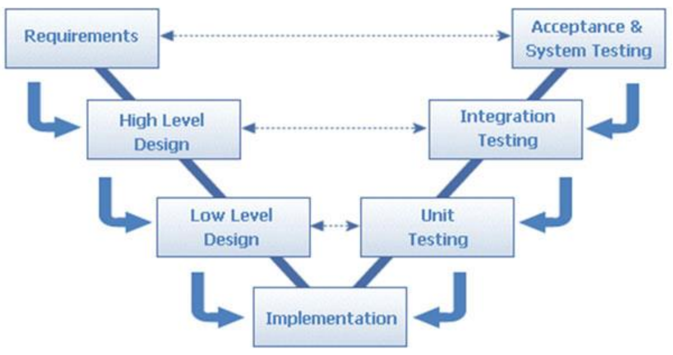

Fig. 2. *The V – Model*

For every single phase in the development cycle a directly associated testing phase is provided. On the left side, development activities including requirement analysis, feature specifications, high-level design, and low-level design proceed from top to bottom. On the right side, testing activities, including, unit testing, integration testing, system testing, and acceptance testing are completed in a bottom-up fashion. In this model, test plans are developed along with each development activities, and the tests will executed in reverse order after coding is completed. Coding phase joins the two sides of the V – Model [5].

III. Agile: An Iterative Approach

Agile method is basically an iterative approach to software development. It represents a major departure from traditional waterfall model and it's based on short iterations and quick releases. It relies on Agile manifesto, written by experienced practitioners (in 2001), that agreed upon four core values become the cornerstones of this ideology [6]:

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

A. The Agile's ancestors

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 [7]. The Lean principles have been applied to Agile software development, and quite often are referred as Agile/Lean. Agile and Lean are relatively broad concepts, since there are several more detailed software development models, the developers of which call them as Agile methods. These models include Scrum, Extreme Programming (XP) and Agile Unified Process (AUP), among others.

B. SCRUM

Ken Schwaber and Jeff Sutherland developed scrum method in 1990s. Scrum is unique because it uses the real-time progress of a project, where feedback loops constitute the core element, to plan and deliver the application. In Scrum, smaller batch sizes and short delivery cycles called “sprints” are used. These sprints are typically one to four weeks in duration. The team members coordinate their work in daily stand-up meetings, which normally lasts for about 15 minutes. At the end of each sprint, product owner and team members meet to assess the progress of a project and plan its next sprint release. This allows the project manager to adjust and revise the project plan based on completed work, not predictions. This results in enormous improvements in quality work, delivery time, and customer satisfaction.

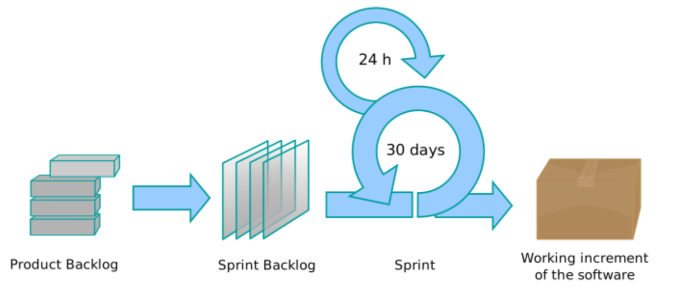

Fig. 3. *The SCRUM cycle*

Scrum is defined by Stober [8] as a drastic simplification of project management containing three roles, three documents and three meetings. As can be seen from Figure 4, 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 stakehlders. 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.

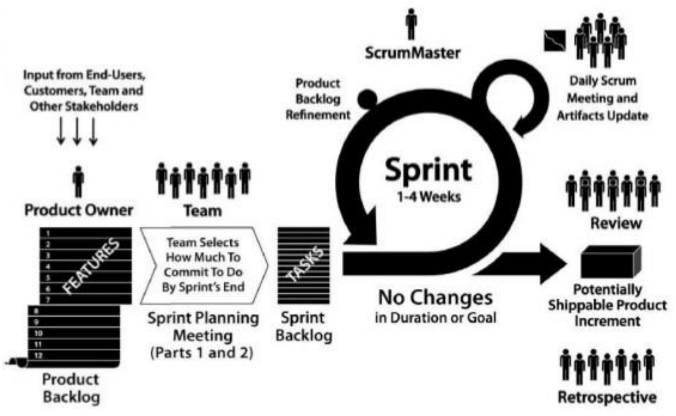
ScrumMaster
Input from End-Users, Customers, Team and Other Stakeholders
Product Backlog Refinement
Daily Scrum Meeting and Artifacts Update
Sprint
1-4 Weeks
Product Owner
Team
Review
FEATURES
1
2
3
4
5
6
7
8
9
10
11
12
Team Selects How Much To Commit To Do By Sprint's End
TASKS
Sprint Planning Meeting (Parts 1 and 2)
Sprint Backlog
No Changes
in Duration or Goal
Potentially Shippable Product Increment
Product Backlog
Retrospective

Fig. 4. *SCRUM overview*

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, and the cycle keeps until the product backlog is depleted and the software release is ready. Releases again are repeated for major software updates. Process flow of scrum testing is as follows:

- Each iteration of a scrum is known as Sprint
- Product backlog is a list where all details are entered to get end product
- During each Sprint, top items of Product backlog are selected and turned into Sprint backlog
- Team works on the defined sprint backlog
- Team checks for the daily work
- At the end of the sprint, team delivers product functionality

C. Developing Agile Approach

In a nutshell, Agile is a time-focused, iterative philosophy that allows to build a product step-by-step (incrementally), delivering it by smaller pieces. One of its main benefits is the ability to adapt and change at any step (depending on feedback, market conditions, corporate obstacles, etc.) and to supply only relevant products to the market. That is why an agile company is usually very flexible, quickly adapts to changes, iterates less while implementing faster, and is able to seize new opportunities as they appear. It enables a fast decision-making process through flexible organizational structure and simple communication. With the implementation of Agile approach, the orientation of the company has to reflect the expectations of their customers and the ways in which their customers work [9]. A research among 601 development released by TechBeacon [10] and IT professionals conducted in 2015, shows that nowadays Agile is the primary management approach. And they mostly use it to enhance collaboration and increase the level of software quality. Agile methodologies provide in facts a remarkable schedule acceleration, but sometimes it can be difficult for planners to determine the effects of these factors on the duration. This aspect can consequently entail a higher complexity in the determination of suitable choices to optimize the project performance. First and foremost, the Agile values, principles and practices should help to guide organization architecture modeling and documentation efforts. In order to be successful at organization architecture, rethink the overall approach and address some fundamental issues is a fundamental step. Some of these issues are:

- Focus on people, not technology:

  Fred Brooks (1995) wrote that “The quality of the people on a project, and their organization and management, are much more important factors in success than are the tools they use or the technical approaches they take. 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.

- Keep it simple:

  A critical concept is that your enterprise architecture models and documents just need to be good enough, they don't need to be perfect. A hand-drawn sketch today on a plain old whiteboard (POW) can often be far more valuable than a fully documented and validated document several months from now.

- Work iteratively and incrementally:

  Agile enterprise architects work in an iterative and incremental manner. Agile modelers will follow the practice Apply the Right Artifact(s) and use a wide variety of modeling techniques as required. They will also follow the practice Iterate To Another Artifact when they realize that the model that they are working on either isn't the appropriate one with which to depict a concept or because they are simply stuck and need to break out of their "analysis paralysis". Agile modelers also follow the practice model in small increments, modeling just enough to fulfill the purpose at hand and then moving on to the next task. It's not required to create complete models, instead it's more clever to enable models that are just good enough. 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.

D. *Implementing the Agile team*

Agile teams at scale are organized into smaller sub-teams. There are four basic methods to organise them:

- Architecture-driven approach: this strategy fits well with high-quality architecture. Here, the sub-teams are assigned around the subsystems/components within the software architecture.
- Feature-driven approach: each sub-team implements a feature at a time as a meaningful chunk of functionality for the stakeholders. This strategy should be applied in situation where the architecture presents a lot of coupling and where there are sophisticated development practices to be fulfilled. 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.
- Open source approach: 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 (e.g. a security framework), and which must evolve quickly to meet the changing needs of the other systems accessing/using them.

IV. Introducing Kanban and Value Stream mapping

A. *Kanban in software industry*

Kanban is a scheduling system of visual management aimed at just-in-time delivery excluding team overloading. As a part of Lean, initially the methodology supported japanese automotive industry. Similarly, to Scrum, Kanban tracks 'to do – in progress – done' activities, but it limits them by the number of 'work in progress' activities (the number is defined by the team manager and cannot be exceeded). This framework or method is quite adopted in software testing method especially in agile testing.
There are four fundamental Kanban principles:

- Visualize work to increase communication and collaboration.
- Limit work in progress to avoid an endless chain of non-prioritized open tasks.
- Measure and optimize the flow, collect metrics, predict future problems.
- Aim for continuous improvement as the result of analysis.

B. *Scrum vs Kanban*

Scrum and Kanban methodologies follow the principles of Agile and Lean. They track processes via scheduling system to ensure transparency and devoted teams manage them. Both methodologies limit the amount of work in progress (Scrum limits them by time units — iterations, Kanban limits work in progress per workflow state). Also Scrum and Kanban focus teams on breaking the work into pieces and supplying the releasable parts of the product earlier and more often.
Despite these similarities, Scrum and Kanban are not the same, and the image below illustrates some of the differences [11].

Table I. Difference between Scrum and Kanban

SCRUM	***KANBAN***
Test must be broken down to be completed within one sprint	No particular item size is prescribed
Prescribes a prioritized product backlog	Prioritization is optional
Scrum team commits to a particular amount of work for the iteration	Commitment is optional
Burndown chart is prescribed	No particular item size is prescribed
Between each sprint, a scrum board is reset	A Kanban board is persistent. It limits the number of items in workflow state
It cannot add items to ongoing iteration	It can add items whenever capacity is available
WIP limited indirectly	WIP limited directly
Timeboxed iterations prescribed	Timeboxed iterations optional

C. *Mapping the Value Stream*

In order to "grease" more the tasks pipeline, one of the noteworthy tools is represented in the adoption of Value Stream Mapping (VSM). The activity arising from VSM enables extraordinarily effective way to analyze, capture and communicate the flow of the processes. Manufacturing VSMs are designed to capture the shorter, more serial workflows and limited scope of assembly lines. Value Stream Mapping (VSM) for software development is the set of all the activities required to develop a software application and includes how it is conceived, developed, and released. It includes both value-added and non value-added activities [12].

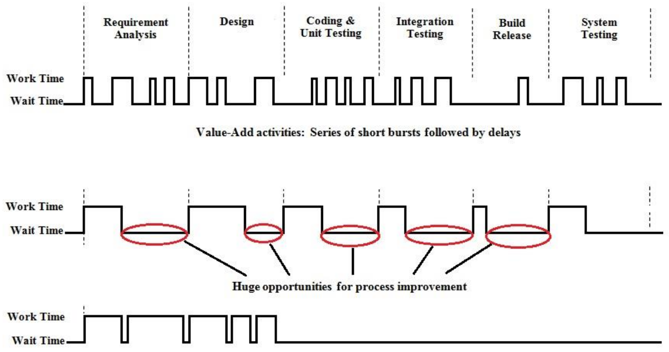

Fig. 5. *Value Stream Mapping in software developemnt flows*

The steps to create a Value Stream map are described here:

a. Identify the actions that take place;
b. Specify how much of the time these actions were being worked on actual work was taking place and how much time was spent on wasting;
c. Look, and denote any loop backs present in the workflow;
d. Total up the average time working on the project;
e. Analyze the entire value stream against value-added and non-value-added activities;
f. Develop and initial value stream map for pull system (or Kanban).

V. CONCLUSION

This conceptual analysis aims to put in evidence how the different methodologies derived by Logistics and Quality Management can be widely applied in the IT ecosystem. If they are combined together, they can offer alternative and clearest stages to deploy optimized performance at the end of the process flow. 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. This "reshuffled methodology" also has the advantage of building confidence in governance groups who have not been exposed to agile before. Another stakeholder management technique most project managers used was to share the scrum or Kanban boards showing key metrics on a regular basis, and concentrate on making the tracking and communications highly transparent.

VI. REFERENCES

[1] J. Widman, S Y. Hua, and S.C. Ross, "Applying lean principles in software development process – a case study," Issues in Information Systems, pp. 635-639, 2010.

[2] D. Joyce, "Kanban for software engineering," A BBC Program, 2009.

[3] D. J. Anderson, and Arne Roock, "The Journal of Information Technology Management," Cutter IT Journal, pp. 6-11, 2011.

[4] C. Ebert, P. Abrahamsson, and N. Oza, "Lean software development," IEEE Software, pp. 22-25, 2012.

[5] X. Zhang, T. Hu, H. Dai, and X. Li, "Software Development Methodologies, Trends, and Implications," Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA, pp. 173-178, 2010.

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

[7] D. Duka, "Adoption of Agile Methodology in Software Development", Information & Communication Technology Electronics & Microelectronics (MIPRO), Split, Croatia, 2013

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

[9] A.B.M. Moniruzzaman, and S.A. Hossain, "Comparative Study on Agile software development methodologies", Global Journal of Computer Science and Technology, pp 11-25, July 2013

[10] Survey: "Is The Agile The New Norm?", May 2015 http://techbeacon.com/survey-agile-new-norm

[11] N. Sirina, "How to choose between Agile and Lean, Scrum and Kanban — which methodology is the best?", August 2016 https://realtimeboard.com/blog/choose-between-agile-lean-scrum-kanban/

[12] H. K. Raju, and Y. T. Krishnegowda, "Value Stream Mapping and Pull System for Improving Productivity and Quality in Software Development Projects", Int. J. of Recent Trends in Engineering & Technology, Vol. 11, June 2014