e-Informatica Software Engineering Journal, Volume 11, Issue 1, 2017, pages: 39–57, DOI 10.5277/e-Inf170102

Experience Report: Introducing Kanban into Automotive Software Project[‡]

Marek Majchrzak*, Łukasz Stilger**
*Faculty of Computer Science and Management, Wroclaw University of Science and Technology
**Capgemini Polska
marek.majchrzak@pwr.edu.pl, lukasz.stilger@capgemini.com

Abstract

The boundaries between traditional and agile approach methods are disappearing. A significant number of software projects require a continuous implementation of tasks without dividing them into sprints or strict project phases. Customers expect more flexibility and responsiveness from software vendors in response to the ever-changing business environment. To achieve better results in this field, Capgemini has begun using the Lean philosophy and Kanban techniques.
The following article illustrates examples of different uses of Kanban and the main stakeholder of the process. The article presents the main advantages of transparency and ways to improve the customer co-operation as well as stakeholder relationships. The Authors try to visualise all of the elements in the context of the project.
There is also a discussion of different approaches in two software projects. The article focuses on the main challenges and the evolutionary approach used. An attempt is made to answer the question how to convince both the team as well as the customer, and how to optimise ways to achieve great results.

Keywords: kanban, lean, automotive, scrum, agile

1. Introduction

Lean thinking is important because it can dramatically reduce waste and introduce built-in quality in every process step. It has been shown that when applying this approach in the manufacturing or service organisation, the productivity has at least doubled. Moreover, this method also significantly reduces delivery time for new products and decreases overall costs [1, 2].

The paper also describes two software projects in the automotive industry, which have employed the Kanban technique in an evolutionary way. In each of these cases, Kanban was used to optimise a different process and was motivated by other business problems. However, the mutual characteristic was the simplification of processes and evolutionary adaptation of both the developer teams and the collaborating client teams.

The idea was to make the communication more efficient, and thanks to that do a project more efficiently. It is necessary to continuously identify bottlenecks and wastes. With the lean principles and Kanban specific practices it is possible to visualise the state of the project and focus on the process.

This paper is organised as follows: an introduction to Lean Software Development and Kanban technique in Sections 2 and 3. Section 4 presents the related work. Section 5 describes projects and their Kanban technique adoption history. General results and conclusions are discussed in Section 6.

[‡]This paper was originally published in the KKIO 2015 proceedings P. Kosiuczenko and M. Śmiałek (Eds.), "From Requirements to Software: Research and Practice".

2. Lean software development

The principles of Lean thinking focus on value added for the customer [3]. By removing the unnecessary processes, activities and artefacts, and on the other hand organising work as a continuous flow, which recombines labour into cross-functional teams dedicated to that activity and constant improvements across the entire company, we have been able to develop, fabricate and sell with half or less of the human effort, tools and overall costs. By introducing Lean thinking and its associated style of operation, we have been able to react faster and more flexibly to the ever-changing needs of our Clients and the modern market. Lean thinking requires continuous learning, growth and most importantly, commitment and understanding from the personnel of any level including management.

Lean Software Development is the application of Lean Thinking to the software development process. The Poppendieck and Poppendieck [4] illustrated how many of the Lean principles and practices could be used in the software engineering context. They proposed seven principles eliminating and managing the waste in software development:

- Eliminate Waste – Do only what adds value for a customer, and do it without delay.
- Amplify Learning – Use frequent iterations and regular releases to provide feedback.
- Delay Commitment – Make decisions at the last responsible moment.
- Deliver Fast – The measure of the maturity of an organisation is the speed at which it can repeatedly and reliably respond to customer needs.
- Empower the Team – Assemble an expert workforce, provide technical leadership and delegate the responsibility to the workers.
- Build Integrity In – Have the disciplines in place to assure that a system will delight customers both upon initial delivery and over an extended period.
- See the Whole – Use measurements and incentives focused on achieving the overall goal.

The automotive software projects use proven technologies, mostly not the latest development techniques. It is because the business functionality is more complex than the other software projects. There are many external system interfaces which extend at the same time. Lean principles offer support to optimise the processes with focus on following aspects:

- Time to market – it is crucial for the software used in the automotive sector to keep stability and compatibility. However, the rapid development of industry requires also more flexibility of software. Automotive companies constantly release new car models or versions of the same car.
- Stakeholder management – at the same time, stakeholder structure was extended in big organisations. Each stakeholder (or group of interested parties) has their goals which need to converge.
- Domain knowledge – another characteristic dimension in software for the automotive sector is domain knowledge which is based on experienced subject matter experts.

Lean principles support the optimisation of the processes with a focus on these three aspects: time to market, stakeholder management and domain knowledge. The main advantages of lean principles for automotive software projects is a fast visualisation of the executed processes and based on it, improve our efficiency.

3. Kanban in software engineering

The name “Kanban” originates from Japanese and could be translated as “signboard” or “billboard”. It is a flow-control mechanism for pull-driven “just-in-time” production. The idea behind Kanban is to execute Lean principles in practice.

David J. Anderson defined 5 Kanban core principles which to agreat extent overlap with Lean principles [5].

- Visualise the workflow – one has to understand the route covered by an item between a request and its completion.

- Limit WIP – limiting work-in-progress implies that a pull system is implemented on parts or on the whole workflow. New work is "pulled" into the new activity when there is available capacity within the local WIP limit.
- Manage Flow – the flow of work items through each state in the workflow should be monitored and reported.
- Make Process Policies Explicit – the process needs to be defined, published and socialised explicitly and concisely.
- Improve Collaboratively (using models & the scientific method) – the use of models allows a team to make a prediction about the effect of change (or intervention).

In software projects, using "Kanban" is becoming increasingly popular regardless of the project stages or production methods. An interesting aspect of this technique is that it is becoming an inside tool in both waterfall and agile processes.

Kniberg [6] points out that Kanban is less prescriptive than other agile methods like RUP, XP or even SCRUM.

Scrum, XP and RUP are highly adaptive while Kanban leaves almost everything open. The only constraints are *Visualize Your Workflow* and *Limit WIP*, which makes it a great tool for quick and efficient workflow and a process management tool. Especially, in the case when the prescribed rules and artefacts do not fit project needs. Scrum prescribes the use of timeboxed methods, but in the case of a support team or a firefighting team, it is hard to plan tasks in a sprint timebox.

3.1. Metrics – a way of observing facts and finding bottlenecks

To make decisions, the management of a given project requires capabilities for adequate situation analysis [7, 8]. This role is served by project metrics, correctly selected criteria according to which the defined parameters can be observed.

A simple visualisation is a great way of investigating the work of a team and the current state of progress. However, it is mainly employed in the day-to-day planning. When more accurate analysis based on a larger volume of data is needed, it is crucial for creating metrics or information-gathering schemes. Metrics are collections of updated and adequately represented data used for problem identification and decision-making.

The value of a good metric is to find a proper way to monitor bottlenecks in the whole processes or at its stages. For instance, one can measure the development processes in detail, i.e the devlopemnt team, integration, defect handling, system tests and network integration (target environment). One of the key findings is to focus on measuring the flow and not the constraints, which means it is better to identify and then remove organisational impediments instead of measuring it. Another interesting aspect is not focusing on speed measurements but on monitoring capacity. Speed is usually related to the human aspect and could foster unwanted competition instead of the collective responsibility for the project [9]. However, the case presented in this paper depends more on dynamics development analysis and not always on sustainable development of new features. The key concepts in measuring work efficiency in this specific case are as follows:

- Reaction time: it is the interval time between reporting an issue and starting an analysis of the problem. A reaction can be defined in a number of ways, however, according to the most common definition the reaction means delegating a task to a team.
- Lead time: a total time measured from task creation until its c ompletion. Lead time takes into consideration all of similar events between these two points, both predictable and unpredictable.
- Cycle time: the correct volume of work.

Figure 1 is used to portray these two concepts. It must be noted here that the entry and exit points for work units, as well as the in-between points, are defined in each project. The purpose of both of these metrics is to show the current work efficiency and the potential decrease in the time and costs of delivering a valuable work unit. More practical details and the corresponding results are described in Section 5.2.

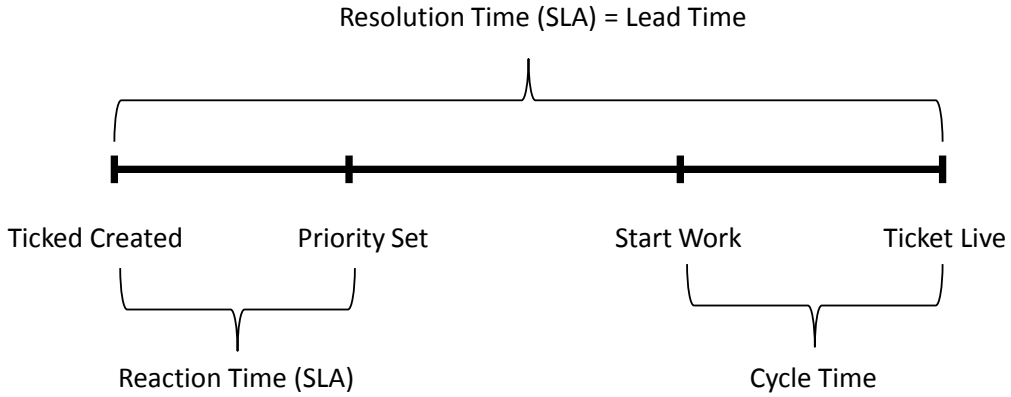

Figure 1. Lead time and cycle time

4. Related work

Mattias Jansson, Operations Engineer at Spotify[1], introduces [10] Kanban in the operations team as the answer to the growing number of different kinds of tasks. Before testing Kanban, the team noticed that although they were quite efficient, they were not able to plan far in advance. The problem was that they were reactive and not pro-active. The growing number of "urgent" jobs from other departments was always more important that the internal tasks and the context switching decreased the team's effectiveness. They realised that the company was growing too fast for them to accommodate.

Soon after Kanban's introduction, they noticed that their lead times became shorter, more internal tasks were done, and the departments they interfaced with were more satified.

In his book *Lean from the Trenches* [11] Kniberg described PUST – a digital investigation system for the Swedish national police authority. Due to the project scale, the teams, as well as the Kanban boards, were divided into subsystems. Besides having WIP limits in regular tasks, they also decided to restrict the number of bugs reported in the bug tracker. In the case of blocker priority, the bug had to be fixed immediately or if it was less important, it had to be replaced with an existing one from the top thirty. Otherwise, it would be ignored. He claimed that such an approach not only allowed for effective communication (lower number of bugs, highly prioritised bugs were immediately fixed), but also avoided continued change control meetings to manage long lists of bugs which would probably never be fixed.

Ikonen et al. [12] conducted a study in a medium-sized project (13 developers) in the R&D field. The investigation focused on the following project work aspects: *documentation, problem-solving, visualisation, understanding the whole, communication, embracing the method, feedback, approval process, selecting work assignment*. The presented results indicated considerable benefits of the Kanban technique including team motivation and control over project activities. Most of the work aspects were positively supported by Kanban techniques inside the team.

Middleton and Joyce in their BBC Worldwide case study [13] showed that as a result of the introduction of the Kanban technique, the lead time to deliver the software improved by over 37% and the number of defects reported by customers decreased by 24%. They observed very similar obstacles to those which may occur after the introduction of the lean principles connected with the environment and workspace, such as the tension within the existing corporate standards and processes. Some especially common obstacles encompass, e.g. office space designed inappropriately for Kanban boards, Kanban and reduction of WIP inability to work with milestones and Gant charts, close team co-operation with the customer may be seen as working beyond the

[1]Spotify is a music streaming service for desktops and smartphones, which aims to provide a wide-ranging music collection.

remit, and the self-managing team of specialist may be challenging to the managers.

They observed that the Agile approach, especially Scrum, has some similarities. However, they also noticed that the Kanban technique and Lean have several advantages over the Agile/Scrum approach. They claim that WIP limits the pull work model compared to the Scrum Push and timeboxed approach, it reduces delivery time and allows to develop better quality software. They also noticed that the ownership and responsibility of the Scrum "impediments list" are diffused. On the other hand, the Lean team must solve the problem immediately if they are blocked, because of limited WIP and visualisation on the Kanban boards. In this case, all staff members are obligated to eliminate the bottlenecks.

In another case study by Middleton et al. [14] the Timberland Company was analysed while practising Lean thinking for two years. They noticed many steps without any aded value in their processes. A survey amongst company staff showed that the majority of them supported lean ideas and thought they could be applied to software engineering. Interestingly only a small minority (10%) was not convinced of the benefits of lean software development. The company showed a 25% gain in productivity and time for defect fixing was reduced by 65–80%. The response on the product released using lean development from customer site was overall positive.

The authors of each study emphasise the benefits of of the introduction of Kanban and collaboration both in the team and with the client. The presented work was mainly conducted as internal projects. Moreover, most of the described projects were relatively small. Hence, they did not require the governance of the customer collaboration process. Additionally, they did not show how to evolve and build the lean values in the team and with the client to establish and use the Kanban technique effectively. Thus we want to present the Kanban technique in the extensive project setting and the way of collaboration with the external customer.

5. Discovering Kanban

This section presents two different approaches and two different perspectives of Kanban introduction. In Project A[2] Kanban was introduced as a tool for dealing with unplanned tasks in Sprint. In Project B the main goal was to unblock communication in the extensive stakeholders structure.

5.1. Project A

5.1.1. Background

The system under investigation covers all aspects of car purchasing in one of the premium car manufacturers in Germany. The system was designed for experts and is used internally by the customer. It allows to buy, lease or rent cars by the clients' employees, institutions or VIPs, car fleet management and used car dealers.

The system consists of two main components. One is a new version of the system developed as a modern web-based application. The second component written in COBOL is the old system's version, which is to be replaced by a new version step by step. Thus, both systems are available to the end users, and this, in turn, requires data synchronisation in real time.

The project uses the Scrum framework with certain small additional procedures, like the additional Scrum of Scrums meeting and the PO-Team meeting. A typical Sprint takes three weeks, some of the user stories (US) are approved during the sprint, some at the end during the demo. The majority of US are confirmed and tested by the PO-Team, however, in the case of larger epics, there are more people involved, including many external IT specialists.

5.1.2. Timeline

The old system version has been being developed since 1990 using the waterfall software development model. At the beginning the new version of the system was also developed using the waterfall software development model. The devel-

[2]Due to a commercial agreement, the project names have been anonymous.

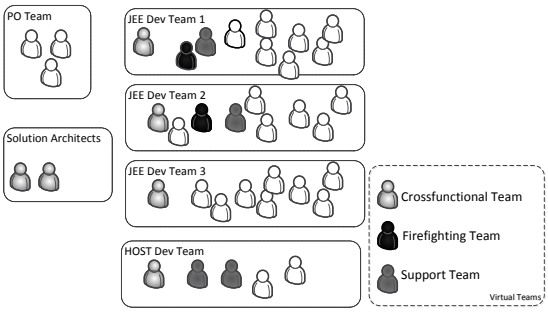

Figure 2. Project A – team structure

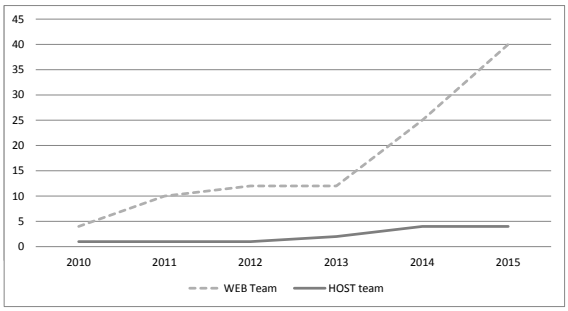

Figure 3. Project A – team grow

opment started in 2010. The first release after 16 months showed that it was not possible to integrate with the old system and that the minimal end user needs were not covered. In 2012 it was decided that to improve co-operation between both systems and ensure faster delivery, new requirements for the whole project will be developed as one Scrum project. After the final transition in 2013, as it was mentioned above, the entire team and the customer used Scrum. Currently, a new version is being released at least quarterly. In the case of urgent requirements, we will provide minor releases extending the latest production version.

It is important to note that the team started to expand rapidly in 2014 (Figure 3). Until mid-2013 no particular need of improvement had been noticed. Support and bug fixing were performed on a daily basis by one or two experienced developers. Also, issues were stored in several systems or delivered via email, hence developers responsible for support and bug fixing had a detailed overview of pending issues.

A dynamic increase in the number of team members forced a change of the project processes. A growing number of developers caused code integration problems. Instead of one Scrum Team the Scrum of Scrums concept was introduced and a different project governance model (3 Scrum Masters, Project Manager, PMO[3] support). The response time and cycle time became longer, mainly due to the insufficient expertise in the automotive industry among new developers and new Product Owners. Even though the developers mentioned above have been previously involved in different automotive ventures, domain knowledge is often one of the main obstacles in a software project. For example, the automotive domain consists of several specific sub-domains and a vast number of process details, – which is one of the main reasons for building custom software. In general, team members do not have all the required knowledge, and the project must acquire additional information before accomplishing productive work. The sources of this information can be relevant documentation, formal training sessions, meetings with domain experts and key users.

Additionally, several new business components started to be delivered, which, in turn, resulted in a growing number of support requests and end-user bugs.

5.1.3. Team composition

The Team consists of 45 people. Around one-third of them are connected with the project from the initial stage. Approximately half of the workers had several years of experience in enterprise projects. The entire team is divided into seven sub-teams (see Figure 2), two of them are virtual. A team member could be assigned to more than one team because of his/her function.

- *JEE Development Teams* (x3, DT) are responsible for the new system version, they use Scrum. Each team has about six members and the Scrum Master.
- *Host Team* is responsible for developing the old system in a Cobol technology. The team consists of 5 members and the Scrum Master.
- *Fire Fighting and Support Team* (FT and ST) consists of 6 members. In most cases, they are nominated in the sprint beginning from

[3]PMO stands for Project Management Office.

each development team. The team is responsible for the integration and production of bug fixing and for providing 3rd line support. The members of the team change every Sprint session.
- *Cross-Functional Team* (CT) consists of technical leaders from each development team and solution architects (SA). It focuses on long-term technical and business decisions and designs new components and supports customer.
- *Product Owners* (PO) *Team* consists of 3 business architects focused on the new User Story development. They work and agree on a new functionality directly with end users and major stakeholders within the organisation.

Up to a 70% of the capacity of people who lack industry experience are used in the initial few months of the Sprint. The slack time is used for internal project training provided by experienced software developers and architects.

Project teams are located in 3 different cities. This type of work is organised in accordance with the Distributed Scrum concept as described by Majchrzak et al. [15].

The development team and project management are located in two cities, the stakeholders and PO work in the third location. The main Scrum meetings were conducted in the customer's office. These meetings were attended by several team members only; the remaining ones participated in them via video calls.

Regarding daily contact with the PO team as well as the major stakeholders, these are managed mostly by means of email communication and video calls.

5.1.4. Engineering practices

From the very beginning, we were focused on XP techniques [16] which could be applied in both the waterfall and then in Scrum framework:
- test-driven development (unit tests);
- clean code [17] instead of code documentation;
- automated end-2-end (E2E) testing covering the user stories;
- continuous integration after each source code change, nightly build includes long-running E2E regression tests;
- source control software and rigorous configuration management;
- bug-tracking software (JIRA [18]);
- documentation in wiki (Confluence [19]).

This results in high test coverage. The team can provide a new release after each sprint. Due to E2E testing, business and technical complexity, the results are not always stable. About 5% of the tests fail regularly. The problems appearing before the release are checked manually to ensure that the reported ticket occurs because of new functionality or because of bugs. Every time a bug is found, it is promptly reported in the issue tracking system. Another aspect which needs to be included in agile projects is the branching strategy. Similar to Shabib et al. [20] we have found that a complex branching strategy could impact the quality of software. Despite the fact that the project consists of several teams, a decision is made to reduce the number of branches to the minimum, and to use the simplest branching strategy possible (Figure 4). The reason behind that was the need of frequent merging (even several times per day) as well as the choice to use one branch for the sprint, maintenance branch and release branch. This was only possible provided that the team decide to use rigorous code-change rules, such as frequent commit and integration (several times per day), and quality rules such as no failing JUnit tests (immediate fix or revert) static code analysis before commit, clean code and alike.

The branching and quality assurance rules, such as XP practices, clean code and fully automated E2E, were also valid for each team working in the Kanban process, hence the changes in Sprint and Kanban were visible immediately for every team member in the project.

5.1.5. Kanban introduction stages

The main impulses for employing Kanban were the doubts expressed by the team with regard to bug correction and new feature implementation, which had not been explicitly defined during the

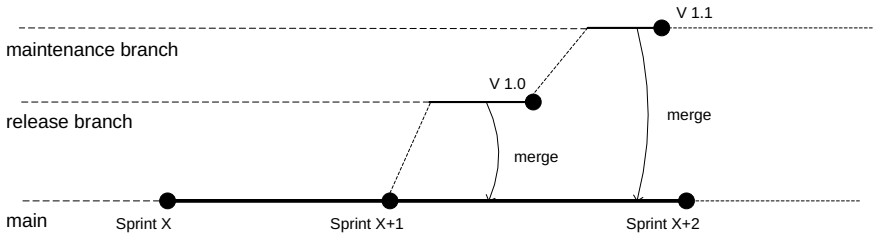

Figure 4. Project A – branching strategy

sprint planning. The change requests were often reported by end end users as tickets sent directly to the support team instead of PO. It is worth noting that the problem was not only which bug and in what order a given task or bug was to be corrected, but also the decision determining the incorrect workings of an application and the decision about who would be the sponsor of a given change.

The main stimulus to implement the improved process of error and support request management originated from the developers' team. The team initiated efforts to improve the already existing processes due to the growing amount of correspondence (delays and handoffs), new team members scattered across localities (delays), domain knowledge (waiting for solution approvals) and some restrictions stemming from the Scrum framework such as timeboxing and the sprint target (switching tasks and relearning – in the case of a new sprint).

Step 1: Identify work to be done. The first step aimed at systemizing the volume of work being done outside the sprint was to create one list of errors and problems from various sources and then organise them in the JIRA system. In the project, because of diverse users and conditioning, the above mentioned errors and queries could be called in using many ways, i.e. by email, by telephone but also with the help of HPQC and Peregrine. From the developer's perspective, many sources of those were impossible to accommodate and respond to, similarly to prioritising decisions.

The unified bug list was not the right solution as the following drawbacks were found very quickly:

1. The developer had to arrange who would be the sponsor of a change or error – fixing.
2. In the case of the lack of symptoms, many tasks had the In Progress status. It is worth noting that the developer was usually assigned to the FT team for approximately only two weeks. As a result, the said developer would begin many tasks (the work in progress was not defined or limited) and practically would not complete any in real life. Then the tasks would be returned to the "To Do list" and the whole procedure would be repeated. Consequently, some errors would wait for weeks before being solved or rejected.
3. All of the agreements and contacts with PO and stakeholders were chaotic. From the point of view of each user group their bugs or tasks had the top priority. Undoubtedly, it resulted in misunderstandings and also caused additional arduous communication by email.

Step 2: Identify workstreams. The next step was to identify the workstream and WIP arrangement (Figure 5). For instance, people working on subjects connected with support received their boards or were able to use the common one, but their tasks were marked with different colours. Similarly, people working on errors (FT) owned their boards. Very quickly another problem emerged, it was connected with the project characteristics. Some of the bugs had to be fixed using a different budget, which meant for example that only 1 FTE[4] could be assigned. Another problem was the fact that many errors were marked as a new feature (CR) and from the project standpoint, they were then investigated in a different budget and with the use of various resources. The constraints mentioned above required the introduction of additional Kanban boards. Then the question concerning the person who would make a choice between

[4]FTE – Full-time equivalent is a unit that indicates the workload of an employed person.

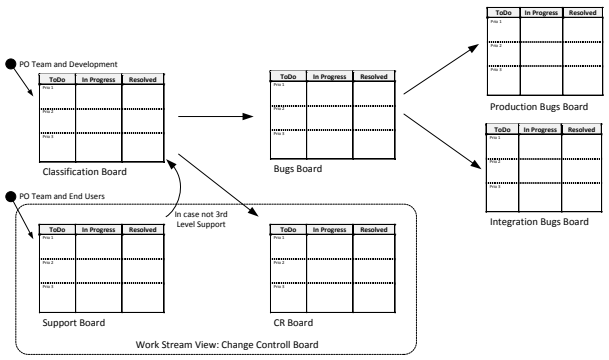

Figure 5. Project A – Kanban boards – work streams

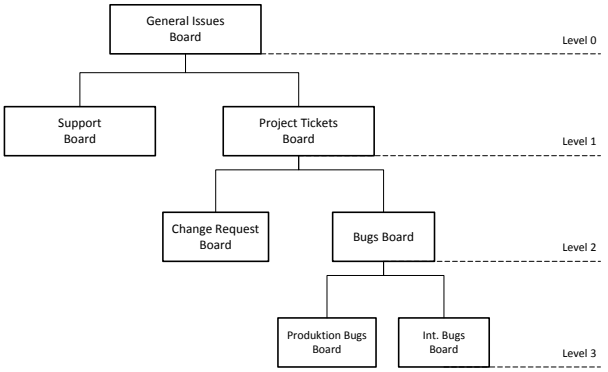

Figure 6. Project A – Kanban processes view

workstreams arose. Certainly, more boards were added and experienced developers or solution architects used them to investigate the issue and decide where it belonged.

Because of plenty of boards, this approach might seem very complicated. However, from the FT point of view, the "To Do Lists" were shortened and the focus was only on bug fixing.

Step 3: Improve the communication. The introduced division between work streams was optimal from the FT and ST point of view. On the other hand, from the management's and the client's perspectives (PO-Team), the existing work streams did not always meet their needs Because of this, to improve communication the efficient conduct of prioritisation meetings, we defined many options which grouped the chosen work streams but still retained simplicity. For instance in Figure 5 Change Control Board formed from Support Board and Change Request Board was identified.

5.1.6. Results

After nearly a year since the introduction of the above process, process, an interview similar to the one suggested by Ikonen et. all [12] was conducted. Opinions concerning the high complexity of the project structure and different expectations were collected from each of the teams. Since the main work stream was done in sprints, we have focused only on selected work aspects.

The hierarchy of the Kanban processes was built (Figure 6), it provides information for particular team members depending on their level. Each project member can find the right perspective and reduce the amount of information depending on their needs. For instance Level 3 suits the FT as they will concentrate solely on selected and initially analysed bugs. On the other hand Level 0 or Level 1 would meet the needs of the project management team in terms of the general project state and SLA.

Documentation. Dispersed exchange of information by email and arrangements during several meetings have been replaced by cohesive comments within the scope of a given task. They allowed us to understand each given problem and the process in which a decision was made. To a developer, it become evident what and why needs to be done. A member of the ST additionally states:

> "Documentation has been improved, there is no worry about losing parts of data. Once something has appeared on a Kanban board, it will not be forgotten or omitted, and it will be equipped with the correct commentary serving afterwards as a source of knowledge in similar problems." [ST]

The error log and new feature request stored in one place allow for a significantly more straightforward analysis of the changelog. It consequently helps to understand who made a decision to introduce a change, and what were their reasons as well as motivations for such an alteration.

> "Due to the significant number of tasks regarding various components and business processes, we have made a decision for all the information to be included in the comments of a given bug or task. This simplifies the

correlation detection between tasks and bugs and the root cause analysis." [CT]

"Bugs are well commented on, hence we can reuse historical information during later stages." [SM]

From the developer team's point of view, the key aspect is the task status and in particular the information which may impact the current tasks in the sprint.

"We are aware of errors and how they are managed; earlier on we used to lose such data, whereas at present we can obtain statuses of errors which are of interest to us." [DT]

Problem solving. The introduction of Kanban allowed for easy task assignment to suitable people in the correct order. In the case when a developer or a customer finds a bug or requests a new feature, he or she can easily issue a new ticket without the need for consultation with the Scrum work model and budget, which made it possible to continue work without delays:

"It facilitates work and provides a structure error correction." [FT]

"Developers are not blocked and know that the reported problem will be properly classified and solved. They are not blocked by unplanned tasks and can develop new user stories without changing the context." [CT]

Even though most of the team thinks that certain progress has been made, ST and CT still see some room for improvement:

"Using Kanban has not solved all of our problems. Too much of a mess occasionally still happens." [ST]

"Still, a large number of tasks, e.g. copying emails and HPQC Bugs into JIRA require additional time and should be automated." [ST]

Also, one of the FT members pointed out that there are still problems due to a largely rotating team:

"On the other hand, we have to improve the process of bug fixing itself. A task once started does not always get completed before it is time for a developer to leave the FT team." [FT]

Visualization. The process of error fixing and CR management is much simpler and more transparent from the team's point of view. Bug statuses and workloads are always visible. Each member can easily select a given board and then the related task:

"Kanban boards look much better and provide more information than a long bug list." [SM]

The Team and stakeholders understand what the support and bug fixing process looks like:

"Once upon a time, I found it difficult to describe the way in which we worked. A project seems to be much more mature, once it it becomes clear how a given process functions." [ST]

"Above all priorities are error statuses and developers who make such errors. Such information is indeed favourable in case of a significant number of various errors." [FT]

Visualisation also helps people working on regular Sprint tasks, and they claim that:

"It is easy to observe whether a person is working to solve a problem which has blocked us and what is its priority." [DT]

"When we need to have a general overview, Kanban boards offer a great deal of assistance." [CT]

Communication. Internal communication between teams and the PO has improved dramatically. Instead of having dedicated meetings to discuss each critical error, regular meetings for the selected work streams have been initiated:

"The number of meetings has grown, however, they have become shorter and allowed us to manage the tasks on given boards effectively." [CT]

"The number of meetings has increased, which could be considered a major improvement. Instead the inefficient information flow via emails, it is much faster to conduct a meeting drawing on the Kanban board and thanks to this keeping the changes up-to-date." [SM]

From the team's point of view, the information concerning prioritising error-correction and the details concerning them have been set in place. On the other hand, from the PO's point of view, communication with the development team has been improved:

> "The client knows who to speak to with regard to a given task and when to expect the solution to a particular error." [FT]

Another equally important aspect is the option of regular progress monitoring.

> "We can see the status of work immediately. I don't have to interrupt people and ask what they are doing." [SM]

A different point of view is presented by developers working on tasks in Sprints. As far as they are concerned, communication has been reduced to the minimum.

> "Not relevant in the case of a development team. We do not have to cope with bugs because we know that the right people from the Firefighting Team will support the process, and we do not have to be involved." [DT]

Approval process. The basic advantage of the defined process was the improvement of work stream choices for new tasks:

> "In the case of fixed bugs, our process still needs improving. Despite the fact that developers know that they should fix and test the bug, we additionally need a *Verify* column in order to explicitly emphasise the need for a test and verification." [DT]
>
> "It is easier to approve bugs and to assign them to the proper work stream." [SA]

Additionally, the introduction of rules concerning the flow and identification of the sponsors responsible for error-correcting has improved support work:

> "In the case when a task cannot be easily solved, because we have to deal with a non-trivial bug or simply with a new feature, we can easily move it to another board." [ST]
>
> "The PO Team checks the Kanban boards, provides some additional information and, most importantly, sets the right priority." [FT]

Despite the above improvements, the team has still seen the need for enhancing the process:

> "Unfortunately, as of now, we have not been able to establish a correctly-functioning approval process. We need another state (column) – Verified. Fortunately we have a proper release process, we can see on the Kanban board what got released and what didn't." [SM]

Selecting work assignment. Through close interaction with the Customer and PO, we have been able to set our priorities right, which in turn has allowed for optimal task accomplishment by the team:

> "You simply take the first task from the first column. You don't have to search for tasks or ask others." [SM]
>
> "Together with the PO-Team, we conduct the prioritisation, thus the most important tasks are at the top of the To Do column." [SA]

Taking into account the aspect of a budget for particular error types, the choice of a task is clearly sensible from the Team member's point of view:

> "Different boards help us find bugs from different work streams." [FT]
>
> "Tasks are split into work streams and could be easily selected based on priority order." [DT]
>
> "We use issue priority in order to select work assignments. If the tasks have the same priority, we simply select the oldest ones." [ST]

5.1.7. General problems and future work

One of the most difficult challenges found in the processes described earlier is the need to perform numerous activities manually. Clearly, using extra human resources, e.g. in order to copy emails to Jira, would be considered a waste in the process. The next step should be the introduction of such systems as Jira Service Desk [21], and e.g. ConnectAll [22] to integrate HPQC and Jira.

It was also observed that assigning new people to a virtual team at the beginning of each Sprint may result in wasting time and and resources. Certain tasks sometimes require several days to analyse, fix and test. If the members of virtual teams are changed while tasks are still in progress, new developers have to start them almost from the beginning. Thus, it was decided to allow people to work on a given task, even though they are assigned only to Sprint development.

5.2. Project B

The project involves the manufacturing of production software in a large automotive concern. A part of this software supports the direct steering of car production in three separate stages: body construction, paint shop and assembly. The steering systems are critical because each potential software error generates relatively expensive problems. The said software is employed in several dozen factories belonging to the aforementioned automotive concern. The goal of this project is the delivery of services within a specified time frame and with specified availability, namely the development of new functions and the support of current and existing functions in the production environment.The support is limited to the most difficult problems requiring changes in the software or specific changes in the system configuration. In Project B, the main challenge in the introduction of the lean philosophy with Kanban techniques was combining transparency principles and contractual issues. Many contractual constraints originate from the extensive structure of stakeholders on the customer's side.

In general, it is possible to visualise many processes on a Kanban board, e.g. governance, transition, staffing, knowledge management, technology and infrastructure, financial and contractual elements.

5.2.1. Extensive structure of stakeholders on customer side

In this case, the customer is one of the biggest world manufacturing consortiums with many layers of interests. On the one hand, there is a need for simplicity, however, however, on the hand the goals is to deliver the production of software – a crucial part of the customer's business. To meet these contradictory expectations a set of stakeholders was identified, however, this article focuss on the following groups:

- The IT department which is the main stakeholder from the contractual point of view.
- Factories which are most important in case of the continuity of the project.
- Quality assurance which is most important to evidence our quality.

5.2.2. Team composition

Due to the massive system function complexity, the team was extended. Taking into account various functions and tasks, the project was divided into the following teams (see Figure 7):

- *Feature Team* (x4) – these are people directly responsible for software manufacturing. The team consists mainly of Programmers and Testers. Each Team is responsible for specific business components.
- *Project Support* (*cross-functional team 1*) – responsible for the infrastructure and continuity of project functioning in relation to technical data, namely integrating both Client's and contractor's networks as well as supporting the build and configuration of management processes.
- *Governance* (*cross-functional team 2*) – responsible for the management and client co-operation takes key decisions concerning the project. It is involved in all the existing aspects of project management including change and risk management.

5.2.3. Kanban introduction stages

The deployment of the agile approach is much more challenging within the realms of a large organisation and an extensive stakeholder structure. The above project description does not focus on organisational or business limitations, it focuses on the employment of the Kanban techniques instead. Because of critical and limited functionality, all of the process changes had to be introduced carefully, i.e. with risk management, which is an indispensable element of the empirical project approach.

Step 1: Establishment of the common workflow. At the initial stages, the arrangement meant that each of the Teams functioned according to their rules and used their individual workflow. The following issues caused difficulties pertaining to the correct definition of the general state of work: defining a completed task (Defini-

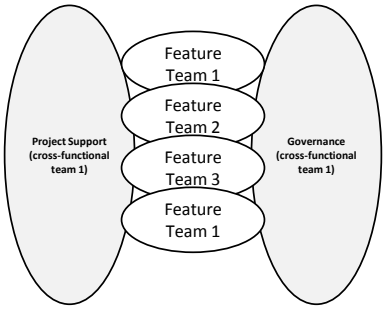

Figure 7. Project B – team structure

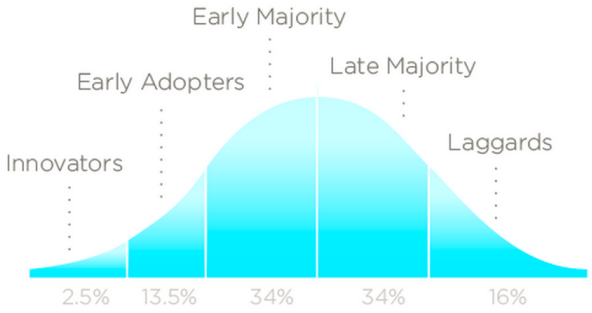

Figure 8. Innovation adoption lifecycle

tion of Done), reporting on critical productive errors (escalations) and seeing the fully complete picture of work in the entire project. After standardising the workflow, it was possible to create the root for visualising the Kanban board. In its initial stages, it comprises everyday work (daily business), meaning current tasks. It consists of the following stages:

- T-Shirt sizing: an initial assessment of a task, which is a relative description of the size of a task resulting from its complexity, uncertainty and repeatability [23, Chapter 7, 16]. At this stage, the estimates are not precise, and the analysis itself should not exceed 4 hours. The t-shirt sizing technique is similar to Planning Poker [24]. However instead of using the Fibonacci sequence, t-shirt sizes are used (XS, S, M, L, XL).
- Problem analysis: at this stage, a detailed analysis is conducted based on the earlier estimate. The purpose of this stage is the definition of the scope of work and its costs.
- Development: at this stage the earlier analysis is used to perform the task. The purpose of this stage is the engineering of a registrable change in software.
- Deployment: the final stage is the employment of software change and in the majority of cases this is the most complex process. The goal is the delivery of the change in the production environment.

It is possible that a problem is solved at each of these levels, which then completes the process.

Step 2: Visualisation processes. Visualisation is the best way to achieve a common understanding of the state of the project, the best way to keep a shared vision. It is possible to find the bottlenecks only when everything is measured and visualised to the whole team.

The reality of communication is that every stakeholder can have different interests. At this phase of introducing Kanban, it became crucial to start collaborating in the same "language". A Kanban board was created on the basis of the earlier study of the said workflow (see Figure 9). The workflow of problem management is described in Paragraph 5.2.3. Visualisation is not only communication improvement, but it is also a major factor in achieving the shared vision and promoting it in the whole project. After the introduction of the visualisations, the following observations were made in the teams:

- the processes were described and changes were continuously supplemented;
- the board was continuously adapted;
- the processes were always visible to all members of the team, and they were proposing improvements (feedback loop).

Step 3: Introducing the culture of self-improvement. The project approaches based on nimble philosophy are tough to implement for multiple reasons, such as the requirements for experience and courage. A given situation can be much simpler if there is an environment open to the Agile and Lean thinking. It is fair to say that their deployment is not possible without the culture of change and constant improvement in place.

In the process of change within a large organisation, one must not forget about sociological processes, an example of which can be the Adoption Curve (see Figure 8) [25]. It is precisely this model that became used in the process of employing change to the project and its close

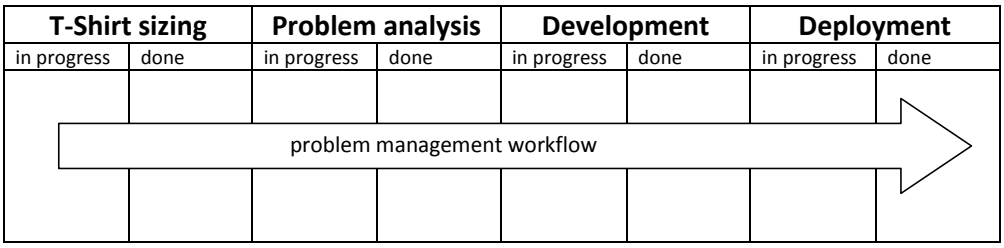

Figure 9. Project B – Kanban board – problem management workflow

environment. The technique used in the project was, among others, the selection of the "so-called" Change Ambassadors (early adopters), who were recruited from the management of selected feature teams. It was this group to be the main communication target in relation to Kanban deployment techniques. In the aforementioned "Early Majority" means the Team.

Instilling the Lean culture allows the use of techniques such as Kanban. Simultaneously, an organisation promotes an adaptive approach on a wider scale, moving far beyond the scope of this project.

Step 4: Managing improvement from the team. The Coach is a crucial role in this operation, and their position is not to be underestimated. However, their role is to guide the Team towards learning the process of coming to correct conclusions. Just as a parent bringing up a child teaches it to walk and then allows it to reach full independence, so does the Team Coach by pointing out specific problems and then teaching the Team members a lesson on independence.

The first dilemma, observed thanks to visualisation and the common workflow, was the lack of comprehensible understanding of the Definition of Done (DoD). At the beginning, each Team defined their DoD in their own way. Unsurprisingly, it invariably led to serious misunderstandings during the execution of the said agreement, especially at the final stages of the project.

Second of all, certain knowledge limitations became apparent within the Team. The Kanban board immediately made the team painfully aware which module lacked the necessary knowledge, where fewer tasks existed and where there was a potential for certain key moves. Through the act of standardising the workflow and programming it correctly in the JIRA, both the executive documentation procedure and the communication regarding production difficulties were successfully improved:

– documentation concerning current problems consists of the necessary meta information, i.e. contact persons, references to other existing documents (i.e. change request);
– summary of existing problems is documented in a uniform manner.

Step 5: Introducing the processes to the Customer. In the described here case, being able to implement the process of improvements was a direct result of the steps taken at the previous stages. One of the most efficient ways to achieve lean principles is visualising a given processes. Together with identified stakeholders (see Section 5.2.1) we decided to start with three working areas: problem management, governance processes and release management

Problem management board. This visualisation shows the whole scope and the parameters of the daily state of work. The content of the board consists of a set of tickets (problems) which were sent to the development team. The goal of the problem management board is to simplify the feedback loop with the factories – one of the crucial stakeholders identified for the project.

The problems (tickets) are prioritised and delegated to the appropriate team member. They can proceed with a particular case of the Kanban board relatively fast, aligning work to their processes and also completing the gaps in the specification.

Governance workflow. The work with the governance processes was dynamic, which was possible thanks to a frame contract joining two companies by agreements that set out the terms and conditions for delivery services. In this project the goal was the delivery of the 3rd level development support. The frame contract allows to adjust the financial part of the delivery – each

service can be negotiated separately. For instance, the workflow of offering (see Figure 10) consists of following steps:

1. *Service* request: the customer requests a specific offer.
2. *Capacity*: project management checks the capacity of the team, inclusive of the know-how in other projects (if needed).
3. *Offer*: a full offer is made to the customer.
4. *Confirmation*: the customer accepts or rejects the offer.

The real workflow is more complex than described here. However, this example shows how the crucial part of the processes can also be involved in the Kanban visualisation. The project management team and the customer's IT department work jointly on the governance board. As a result, a faster "one-piece flow" is achieved. It is the crucial part of Lean Manufacturing [26] and also works well with software development.

Release management processes. Besides ensuring the quality of the software solution, it is necessary to deliver software packages to the factories. The development team delivers various types of ensembles: release, service pack, fix pack and hotfix. The roll-out team in the factory installs the corresponding package and ensures the continuity of production. The development team supports packages in case of emergency. Quality assurance is the most important stakeholder in this area. With the visualisation of the release management processes, the delivery can be prioritised more easily and additionally, the steps of the processes can be adapted relatively fast. The workflow of release management (see Figure 11) consists of the following steps:

1. *Development*: preparation of delivery.
2. *Ready for tests*: finishing delivery and releasing it for the customer.
3. *Tests*: the customer conducts acceptance tests.
4. *Ready for roll-out*: finishing the delivery and releasing it for roll-out (installation).

5.2.4. Results

One of the most significant consequences of the introduction to Kanban is the ability to measure a process, for example by quoting such defined metrics as:

- an increase in the number of created tickets relative to the closed cases (see Figure 12);
- a possibility to measure the average time for closure and the costs of fabrication (Lead Time and Cycle Time);
- cyclical report of shifts in the original estimate, which meant a comparison of adequate work estimation in the T-Shirt sizing phase to the actual volume of work being done. The report made the early detection of the most incorrect estimates and their causes possible;
- the quality of task documentation coming from the Client is also measured, which in turn allowed for the introduction of multiple improvements. The final effect was improved communication with respective Client departments.

Additionally, SLA values (Service Level Agreement) are measured within the scope of the said project based on the previously agreed parameters. The achieved SLA may shift up to a certain extent. The following SLA indicators are used in the project:

- Reaction time from the moment a work unit was created to the beginning of actual work (early analysis),
- Time of the initial analysis (T-Shirt sizing).

In the analysis phase (Problem Analysis, Development and Deployment), the high level of vagueness made it impossible to introduce the SLA which would measure the end of work. An sample cumulative chart (Figure 13) highlighted the piling up of tickets in the analysis phase for the final period between September to November. Such visualisation enhanced the credibility of the reports for the Client. The goal of metrics is to monitor the state of the project and react when problems occur. In this case most valuable metrics are: Reaction Time, Lead Time and Cycle Time. In practical terms, the results of the metrics are not always easy to understand, which was also the experinece gained in this project. Matters which need to be interpreted separately are described below.

- Incompletion of the data – there was sometimes a gap between real processes and the

Service request		Capacity		Offer		Confirmation	
in progress	done	in progress	done	in progress	done	in progress	done

governance offering workflow

Figure 10. Project B – Kanban board – governance workflow

Development		Ready for tests		Tests		Ready for rollout	
in progress	done	in progress	done	in progress	done	in progress	done

release management workflow

Figure 11. Project B – Kanban board – release management

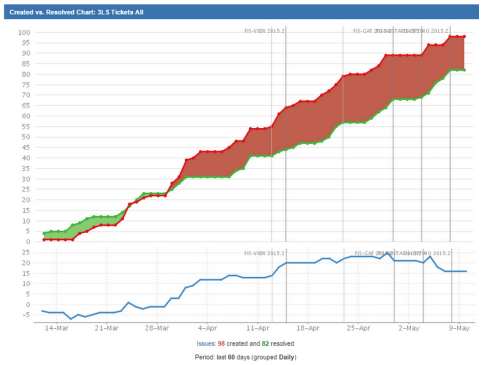

Figure 12. Created vs. resolved chart

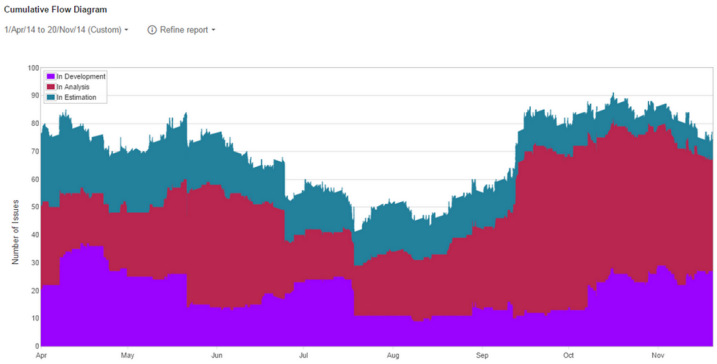

Figure 13. Cumulative flow diagram

documented data; for example, a request which was not registered in the system. It was assumed that these data were not crucial for this metric.

– The learning curve while introducing changes to the processes – the whole process of introducing lean changes with Kanban techniques is relatively time-consuming. Establishing the efficient workflow of tickets lasted more than six months, and the following six months were required to teach the entire team. It was assumed that there always was a learning curve and the measured data were calibrated with time. Hence, it effectively corresponded with the reality.
– Team rotation – the real problem was when the capacity of the team varied. All projects need to deal with such issues not only because of the growing numbers in teams on the team, but also because of technical experience and domain knowledge, which should be taken into account. Unfortunately, the impact of this issue on the presented results cannot be accurately compared.
– Bad multitasking – the more tasks there are in the progress, the less efficient the working time is. The Kanban visualisation allows us to minimise this problem. However, this aspect needs considering when interpreting the final results.
– The complexity of business knowledge – it is a well-known fact that the software for production systems is complicated because of various elements, such as interface systems,

real-time constraints, security aspects and production continuity. There is no direct relation between the level of business complexity and the quality of the project; yet in metrics, it was observed that in the tickets, comparison to the others, this adverse effect can be eliminated by using medians instead of averages.

Reaction time. The total time from reporting a problem to the moment the development team can start dealing with it. Table 1 and Figure 14 show the effects of the described aspect of the described aspect "the learning curve during the introduction of changes to the processes". The year 2014 was the learning phase. From 2015 Q1 tuning the reaction time through minimizing "bad multitasking" was started.

Lead time and cycle time. The lead time is the total time required to develop a solution to the problem including corresponding activities, both the predicted as well as the unpredicted ones. It is the time from task creation until its completion. Cycle time is the correct volume of work.

In both tables (Table 2, 3) one can observe the difference between averages and medians. The cause of the difference is that a small amount of tickets was extremely complex. It corresponds to the described issue "complexity of business knowledge". In Figure 15 one can observe how the cost of the delivery was optimised. The period between 2013 Q4 and 2014 Q2 was the time when measuring data was not complete. From the 2014 Q3 team rotation begins. In the first quarter of 2015, we finished installing all modules of the software.

6. Summary and key observations

In a progressively larger number of IT projects, one can easily notice a trend towards process and tools optimisation. The software companies and its customers have spotted flaws in the current perspective based on waterfall approaches. There is a big potential in creating waste, e.g. through administrative behaviour. Moreover, frequently chosen software development methodologies do not encompass certain much-needed processes.

Although the Kanban technique is not the subject of many analyses and was not promoted as much as the Scrum or XP ones, it is more and more frequently used in software projects as one of the tools of Lean thinking. It can be used with positive results in each project type regardless if it is an "Agile" or "Waterfall" style operation.

It should be noted that this basic and simultaneously intuitive mechanism is a powerful tool allowing for the easy optimisation of nearly every activity and process within software projects. In both cases, substantial profits were observed both on the side of our Team as well as on the Client's side over a relatively short period.

As mentioned above, the most important aspects of those undoubtedly are visualisations, process regular order and the creation of a cooperative platform, which can be easily modified and adapted to any given target group.

During the analysis of the consequences of Kanban deployment another perspective emerged. Taking into account the human factor, it can be observed that Kanban uses triggers as a tool for gradual self-improvement of each team, a sort of evolutionary step towards the reform of documentation and fabricating processes. Hence, unlike something forced upon the staff by the management or outside specialists, Kanban results in an all-natural, symbiotic and adaptive process.

7. About Capgemini and Software Solutions Center Wroclaw

With 180,000 people in over 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2014 global revenues of EUR 10.573 billion. Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organisation, Capgemini has developed its own way of working, the Collaborative Business

Table 1. Project B – reaction time

Period		Reaction Time		
year	quarter	average	median	issues
2013	Q4	6d 8h	1h 21m	73
2014	Q1	2w 5d 6h	3d 3h	156
2014	Q2	1w 4d 23h	19h 14m	195
2014	Q3	3w 14h 21m	16h 52m	200
2014	Q4	2w 8h 38m	20h 42m	171
2015	Q1	3w 7h 41m	6h 31m	176
2015	Q2	1d 15h	5h 30m	161
2015	Q3	2d 6h	4h 31m	179
2015	Q4	1d 15h	3h 2m	223

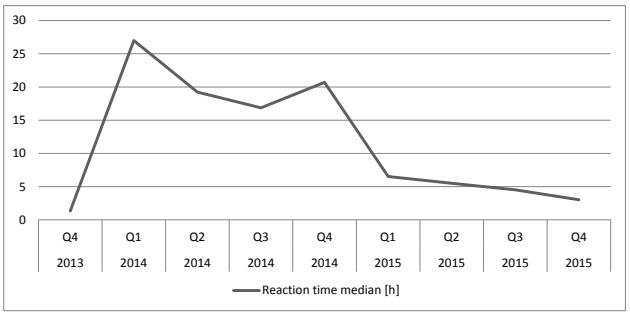

Figure 14. Project B – median of reaction time

Table 2. Project B – lead time (time line)

Period		Reaction Time		
year	quarter	average	median	issues
2013	Q4	6w 5d 20h	7w 3h 26m	25
2014	Q1	4w 3d 19h	2w 1d 10h	76
2014	Q2	8w 4d 4h	6w 2d 2h	97
2014	Q3	9w 6d 5h	5w 6d 4h	166
2014	Q4	16w 5d 22h	11w 6d 4h	173
2015	Q1	22w 1d 7h	15w 1d 19h	231
2015	Q2	15w 5d 16h	10w 6d 22h	148
2015	Q3	17w 5d 8h	10w 3d 6h	205
2015	Q4	16w 3d 5h	6w 1d 6h	427

Table 3. Project B – cycle time (in progress)

Period		Reaction Time		
year	quarter	average	median	issues
2013	Q4	3w 1d	2w 3d 1h	41
2014	Q1	3w 5d 5h	2w 3d 22h	62
2014	Q2	4w 6d 3h	3w 1d 5h	107
2014	Q3	4w 6d 12h	3w 5h 44m	130
2014	Q4	9w 21h 6m	5w 3d 2h	177
2015	Q1	7w 6d 14h	5w 10h 20m	200
2015	Q2	8w 1d 16h	3w 3d 20h	171
2015	Q3	6w 6d 8h	2w 6d 15h	191
2015	Q4	5w 1d 21h	1w 2d 23h	328

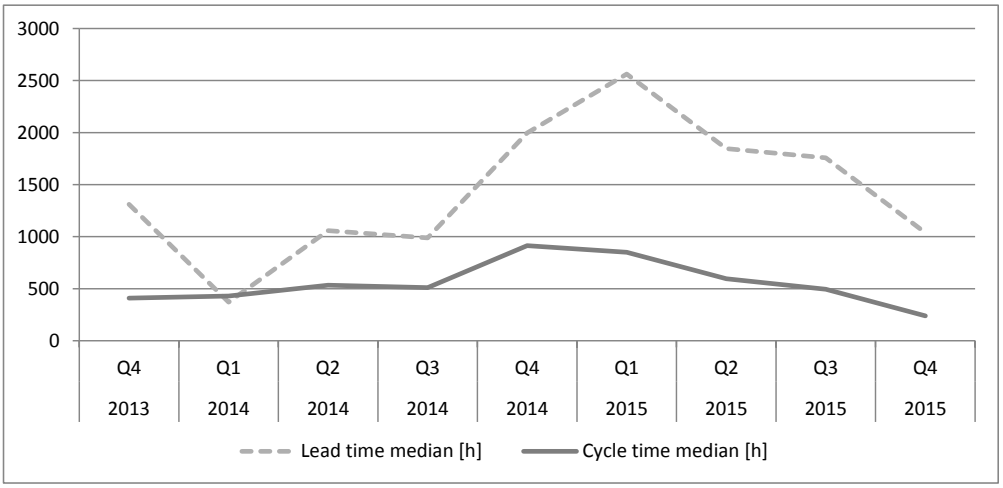

Figure 15. Project B – median of lead time and cycle time

Experience^{TM}, and draws on Rightshore®, its worldwide delivery model. Capgemini in Poland employs 6500 consultants and IT as well as business process experts. Centres for IT and business process outsourcing services has operated in Wrocław, Poznań, Kraków, Katowice and Opole with the main office serving the Polish market based in Warszawa. Capgemini Software Solutions Center exists in Wroclaw since 2004. More than 800 IT experts currently work in Wroclaw delivering high-quality services in the areas of software development, software package implementation and application lifecycle services to German-speaking clients.

References

[1] T. Ohno, *Toyota Production System: Beyond Large-Scale Production*. Cambridge, MA: Productivity, 1988.

[2] J. Koplin, S. Seuring, and M. Mesterharm, "Incorporating sustainability into supply management in the automotive industry – the case of the

Volkswagen AG," *Journal of Cleaner Production*, Vol. 15, No. 11, 2007, pp. 1053–1062.

[3] J.P. Womack and D.T. Jones, "From lean production to the lean enterprise," *Harvard Business Review*, Apr. 1994.

[4] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Agile Toolkit*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[5] D.J. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, Apr. 2010.

[6] H. Kniberg, *Kanban and Scrum – Making the Most of Both*. Lulu.com, 2010.

[7] K. Petersen and C. Wohlin, "Measuring the flow in lean software development," *Software: Practice and Experience*, Vol. 41, No. 9, 2011, pp. 975–996.

[8] M. Host, B. Regnell, J.N. och Dag, J. Nedstam, and C. Nyberg, "Exploring bottlenecks in market-driven requirements management processes with discrete event simulation," *Journal of Systems and Software*, Vol. 59, No. 3, 2001, pp. 323–332.

[9] M. Staron and W. Meding, "Monitoring bottlenecks in agile and lean software development projects," *Product-Focused Software Process Improvement*, 2011, pp. 3–16.

[10] M.J. Michael Prokop, "Use of kanban in the operations team at spotify," *InfoQ*, Sep. 2010.

[11] H. Kniberg, *Lean from the Trenches: Managing Large-Scale Projects with Kanban*. Pragmatic Bookshelf, 2011.

[12] M. Ikonen, E. Pirinen, F. Fagerholm, P. Kettunen, and P. Abrahamsson, "On the impact of Kanban on software project work: An empirical case study investigation," in *16th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE, 2011, pp. 305–314.

[13] P. Middleton and D. Joyce, "Lean software management: BBC worldwide case study," *IEEE Transactions on Engineering Management*, Vol. 59, No. 1, 2012, pp. 20–32.

[14] P. Middleton, A. Flaxel, and A. Cookson, "Lean software management case study: Timberline Inc." in *Extreme Programming and Agile Processes in Software Engineering*. Berlin, Heidelberg: Springer, 2005, pp. 1–9.

[15] M. Majchrzak, Ł. Stilger, and M. Matczak, "Working with agile in a distributed environment," in *Software Engineering from Research and Practice Perspective*, L. Madeyski and M. Ochodek, Eds. Polish Information Processing Society Scientific Council, 2014, pp. 41–54.

[16] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed. Addison-Wesley Professional, 2004.

[17] M. Robert C., *Clean Code: A Handbook of Agile Software Craftsmanship*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2008.

[18] Atlassian, JIRA documentation, (2016). [Online]. https://confluence.atlassian.com/display/JIRA/JIRA+Documentation

[19] Atlassian, Specification – confluence advanced editor, (2016). [Online]. http://confluence.atlassian.com/display/DOC/Specification+-+Confluence+Advanced+Editor

[20] E. Shihab, C. Bird, and T. Zimmermann, "The effect of branching strategies on software quality," in *2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM*, 2012, pp. 301–310. [Online]. http://doi.acm.org/10.1145/2372251.2372305

[21] Atlassian, JIRA service desk documentation, (2016). [Online]. https://confluence.atlassian.com/servicedeskserver030/

[22] Go2Group, Connect all, (2016). [Online]. http://www.go2group.com/connectall/

[23] K.S. Rubin, *Essential Scrum: A Practical Guide to the Most Popular Agile Process*, 1st ed. Addison-Wesley Professional, 2012.

[24] M. Cohn, *Agile Estimating and Planning*. Upper Saddle River, NJ, USA: Prentice Hall, 2005.

[25] E.M. Rogers, *Diffusion of Innovations*, 5th ed. Simon and Schuster, 2003.

[26] K.L. Jeffrey, *The Toyota Way: 14 Management Principles From The World's Greatest Manufacturer*. McGraw-Hill, 2004.