Kanban University

KANBAN AT NEMETSCHEK SCIA

Balancing Collaboration between Cross-functional Teams

KANBANCASESTUDYSERIES

The phone rang in the office of Jean-Pierre Rammant, the CEO of Nemetschek Scia. He picked it up and heard a familiar voice: "Ivan and eight other developers have quit."

Ivan was a senior software engineer located in the Czech Republic along with the rest of the development group. He had been working on the company's most important product—Scia Engineer, calculation software used by construction engineers. Ivan was the creator and maintainer of the code behind the report application[1], which is such a critical component of any construction software that if it isn't stable or accurate, engineers may very well switch to some other construction software.

Immediately after the phone call, Jean-Pierre knew that most of the code behind the key reporting application had just gone to the garbage. The reporting function of Scia Engineer was already causing problems for engineers around Europe. It would crash often, many times losing data that had already been generated. Each crash meant starting all over again. The report was behaving slowly and had been flagged for rebuilding with highest priority. But Ivan had just quit. The year was 2009.

In May 2013, Nemetschek Scia launched its annual release with a complete redo of the report application. The new version automatically recovers data that has already been generated even if the application crashes. "I don't think Scia Engineer has ever had such a crucial release," says Patrick Steyaert, a trainer and coach consulting with the company. It has taken five man-years to accomplish the necessary amount of coding alone. That is an achievement that was deemed impossible before now.

This is the story of how Kanban transformed Scia Engineer from a dysfunctional product in 2009 to a progressive solution in 2013.

Company Background

Twenty-five developers and ten product development engineers (PDEs) are responsible for the development of Scia Engineer.

Almost all of them are situated in the Czech Republic and are organized into teams designated according to a particular structure domain—steel, concrete, solver, etc. Product managers and business analysts are all situated at the headquarters of Nemetschek Scia in Belgium.

Everyone on the Scia Engineer team has a civil engineering background and each has progressed into a more specialized role in the company. Whether it was development, product management, business analysis, or something else, each person has grown incrementally to fill the position he or she now holds. Currently, however, a few of the people who replaced the developers who left in 2009 are purely IT developers.

Patrick Steyaert has been consulting for Nemetschek Scia since 2005. Over the years he has grown to know the Czech developers pretty well.

Patrick says, "I have been to Prague to meet the developers many times. Each time I noticed the same thing: When we went to dinner with them, evenings would progress long into the early hours. The guys would continue to engage in conversations about the way we should do something in the software in terms of constructing it. They just can't let it go."

He says that because they were civil engineers prior to becoming developers, they were always energized by the mind-boggling challenge of how to make the product better.

Scia Engineer is design and analysis software for large construction projects that dates back to the mid '90s. It was introduced as the product Esa

[1]The *report application* is core functionality in software products for design and analysis of large construction projects, such as bridges and skyscrapers. The report needs to be detailed and absolutely accurate. It serves as proof to regulatory bodies for permits and to insurance companies that the construction will not fail.

Prima Win. Since then there were a few more versions before the current product was released in the early 2000s. Code for this product has never been fully refactored, meaning that older code is still used in the newer versions. Releases of features and partial improvements for Scia Engineer have been provided once a year to customers.

"You don't add a feature just like that and see if it crashes a building," Chris Van Loock, the Quality Process Manager at Nemetschek Scia explains. It is a scientific software product and it needs a certain process from analysis to testing.

The First Need for Change

Back in 2005, the management at Nemetschek Scia had a growing problem. Releases for Scia Engineer were coming in with huge delays of up to a year, if they came in at all. Management thought that continuing to use older code from prior versions might be the core reason for the delays.

Patrick Steyaert was called in to inspect the quality of the code. As he talked to people like Chris Van Loock and others around the company, it became evident that code quality was not the root cause of the problem.

What was in fact causing the delays was how product management and development interacted with each other: Their process for reaching decisions about the priority of what would be developed (or not developed) was slowing things down. So reviewing code was exchanged for reviewing process.

In 2009, when the coach heard that Ivan and eight others were quitting, he was baffled. What had gone so wrong, and why? Scrum, a popular process approach, had been adopted by the company in 2005. It had been used to improve the working relationships between product management and development and to decrease delays. Scrum was supposed to have prevented problems from boiling over, not create more of them.

Before the agile method Scrum was introduced, Scia Engineer was operating on a long-term plan for a release each year. Because everyone in the company had a civil engineering background, they all felt equally qualified to have a voice in prioritizing which features would be included in a release. Individual responsibilities had never been defined very clearly, and without completely separate and defined roles, product managers and developers each believed they knew what was best for development.

Culturally different and physically spread out, the two teams were contributing ideas and changing their minds about the product's direction more often than not. Involved actively in product decisions, developers would try to make time for the engineering as well. All along there had been no safeguard against deviating from the originally set priorities. Many ideas were started but never finished. Delays accumulated to such that they were a year behind on the initial release date. Jean-Pierre and the rest of management were looking for an explanation and a change.

The review of the company's processes showed a need for more agility. Planning a whole year in advance was ineffective. Instead of accomplishing improvement, it was distracting everyone and creating friction between individual team members. The people were behaving too much like civil engineers— in the hard and costly world of construction, rework and modification are prohibitively costly and time-consuming and safety is critical. The software development, a wholly different domain, was suffering firstly from this way of thinking and secondly the lack of attention it was receiving.

Major projects such as reengineering existing functionality were a nightmare for product and development team managers to estimate and follow. Those sorts of projects were in fact avoided because people were too insecure to pursue them. As it was, requests in general were already mixed in their scope and size. Tasks came from several sources—strategic direction communicated by leadership, improvements asked for by support, business-as-usual bug fixes, and routine tasks for maintenance of the product, such as assuring compatibility with operating system upgrades.

Scrum Comes In

The solution in 2005 was to introduce two major changes: a process that improved and structured the flow of work as well as healthy restrictions that clearly defined the individuals' responsibilities. This was meant to focus everyone's attention. It was hoped that as a result people would be able to set a realistic plan and keep a reasonable schedule for releases. The agile process Scrum, with its time-boxed iterations, as well as the clear distribution of responsibilities for prioritizing the features from engineering, seemed well fitted to these sorts of issues.

Probably no one could have expected that this holistic product

development strategy would hide the ominous potential to cost the company some of its most talented and engaged developers in 2009.

With the Scrum process, the development and product teams' responsibilities were split. Prioritization and product decision-making were left solely to the product team in Belgium. That restriction was inevitable. If too many people had direct input to priorities and targets, the product development would have continued losing focus. There was serious danger that in the long run it would have been out-gunned in the market by superior niche products.

The clarification of roles was pushed in an attempt to organize faster delivery with higher quality. Product managers were deemed to be the ones who could determine what was best for the product and transfer that information to the developers. They would set the targets and the direction and turn those into user stories and tasks for development.

From there, the development team could simply take what was in the input queue and deliver it. Contributions toward the product's content would no longer be part of the development team's duties.

This segregation of responsibilities was implemented. Next was the introduction of time-boxed iterations. These time-boxed increments of functionality were intended to set a stricter timeframe within which developers needed to execute their tasks.

The sprint, as such a timeframe is called in the Scrum process, was to be six weeks long. Tasks would be discussed and pushed for development during a meeting in the beginning of the period, called a sprint kickoff meeting.

In the following six weeks, the developers had to accomplish the tasks that were assigned by the product team: They would produce code for four weeks and then fix bugs for two weeks.

Each morning the PDEs met with the developers by way of a conference call. During these daily stand-up meetings, developers reported on what they had done the day before, what they planned to do that day, and whether they had come across any impediments or stumbling blocks.

In that way, each individual team manager got a clear sense of what was happening on a daily basis and how the team was progressing. And since what was scheduled for the sprint was delivered more or less within the timeframe, many had hoped that this trend would continue for the bigger release as well.

By its definition, Scrum is intended to instigate collaborative teamwork and bind together a group of people to work as a unit to reach a common goal. Implementing Scrum at Scia Engineer was intended to keep the product team engaged in the development part of the process, as well as the planning. Even though their duties were mainly about setting targets and priorities, they also needed to follow up and see what was happening after distributing the tasks. It was believed that only through observation of the end-to-end workflow could a good quality result be accomplished.

During the first few years after the agile process was introduced, Patrick saw a lot of resistance to Scrum within the company. However, metrics were showing that lead times were improving. Or so it had seemed.

The truth of the matter was that deep down, the developers in the Czech Republic had become increasingly unhappy. After being segregated from product management, the developers were stripped of their ability to have a say in the product's content. The very same people who cared so deeply about the product that they would debate about it late into the evening were disenfranchised. They had lost their grip on the content and instead had been occupied with mundane tasks like reviewing their own code or unit testing[2]. And they weren't even sure why. Where their passion used to exist, embitterment and disappointment had bloomed.

According to Chris Van Loock, "There has always been a drawback with the developers and the late night conversations: what if someone from the product decision making team has not been present during the passionate discussions and product-altering visions? Have the discussions been made tangible, is the information retained somewhere, are the conclusions and ideas actually realistic and marketable? The Czech developers really have always been quite energetic, fueled by their enthusiasm for the overall product. That enthusiasm has withered by being isolated from it along with the tedium of daily tasks. If the energy builds up too much without somewhere for it to go, there might be lightning that follows."

Scrum had somehow ignited a lightning storm.

Change Comes in Once More in 2009

Once the nine developers left to create their own company, it

[2]Unit Testing is a technical term that refers to testing individual functions or methods within the code on a very low level.

was clear that something in the process or the organization needed to change.

Delivery delays had been fixed with Scrum, and segregation had been implemented, but, obviously, at a high price. The whole product and development team had fixated themselves on their individual tasks and deadlines, but had failed to learn to collaborate with one another.

The passion people used to carry with them was exchanged for busyness. The lack of collaboration resulted not only in nine people leaving, but also in a product that had ceased to improve as much as the market required of it.

"I think we should try Kanban," Patrick said to Jean-Pierre one day in 2010.

"It will provide a big picture for everyone and it will get people talking to each other without changing the way they work in any way," he explained.

He had come to the conclusion that the staff members needed a certain avenue to express their opinions. Kanban, with visualization boards—where everything is transparent and subject to discussion—could be the right one.

If anything was going to pull the company out of the dilemma it was facing, it was going to be its own talent realizing its full potential and capacity. That would happen only if people felt they were emotionally attached to the product.

Emotional attachment begins when everyone feels they are part of the decision-making process. So if a new approach was to be tried, everyone had to agree to it. The very fundamentals of Kanban called for such an incremental adoption.

Patrick organized a kick-start workshop for a selected pilot team. The main goal of the workshop was for each individual to identify the problem before seeking a solution. Participants were asked various questions to find out why they felt unsatisfied with the current status quo.

"We said out loud everything that bothered us," Peter, a team leader, recalls. "I remember the click during the workshop. Patrick showed us different Kanban visual boards and how tasks were nicely spread out and organized on them. It was so clear. Through that example board, I clearly saw the benefits this would have for us, seeing what is in progress, being able to identify bottlenecks or any other issues for that matter. Also, besides the transparency, it showed simplicity: Kanban is not complex; it is in fact a quite simple, natural way to understand a process. It's funny that we never considered this way of thinking before."

Witnessing the progress of the pilot team, more teams were kick started with Kanban. But the biggest test for Kanban was yet to come. Jean-Pierre and Scia Engineer had to get the reporting application reengineered. The project needed a devoted team with a common understanding of the goal and how it was to be accomplished. Perhaps Kanban's visual boards held the key for success of this major project.

At the end of 2010, a team of five developers, two product managers, and a business analyst gathered to make a plan for the Reporting rewrite project.

"You don't do a reengineering such as that incrementally and see if it works or not. You need an understanding by everybody that the choices here are top priority.

"Kanban is not complex; it is in fact a quite simple, natural way to understand a process. It's funny that we never considered this way of thinking before."

It is all about discipline," Patrick says.

It took about three months, many meetings with clients, and several mock-ups of the report application before the reengineering could even begin. The product managers outlined a special charter for the project. The vision document gave a rough overview of how the next two years' worth of prioritization, coding, testing, analysis, deployment, and merging had to run if the reengineering was to succeed.

Of course, there was risk involved. The stream of requirements that had to be accomplished was huge. But the team seemed committed and viewed the risk as a driver, not as an impediment. A devoted project manager was following the entire process.

In March 2011, the reengineering of the report application began. The team had a huge stream of requests and requirements to be estimated and evaluated on equal terms with the rest. They were all put in the product managers' backlog. Everyone, including developers, could pitch their suggestions.

Figure 1.1 The Discovery Kanban board. Each request is put on it and tested through various stages in order to meet the product managers' acceptance criteria. A column on the board represents a stage, which in turn consists of "In Progress" and "Done" states. The system gives out signals if a work item stands too long in the "Done" column without someone picking it from there to move it along to the next step. In this Kanban board a card might be rejected halfway across because the requirement has been deemed unfit.

The reengineering work process was organized in three main Kanban boards—a Discovery Kanban board, an Expert System Requirement Kanban board, and a Delivery Kanban board.

The Discovery Kanban board was about creating and developing ideas for the report application. Product managers picked a request from the backlog (where everyone could have an input) and placed it on the Discovery Kanban. Each request became an individual card on the board and as it was evaluated it was moved along.

"The Discovery Kanban is a sort of triage. It needs to be, because demand will always be much higher than what can be accomplished," Geert Adriaenssens, the Product Manager at Nemetschek Scia, explains.

Many of the ideas had to be thrown away—casualties to the fact that engineering was limited—and only features that were the most suitable for the product and the client could make it.

The Discovery Kanban—and the transparent journey of each request on it—helped people see which ideas made it, which ones did not, and why. With this sort of information available to everyone, the sense of lost influence began to disappear.

Once a certain card had succeeded and passed through each stage of the Discovery Kanban, it was transferred to the Expert System Requirement Kanban board. By that time the product managers had already recognized it as a vital requirement for the product.

The Expert System Requirement Kanban board was the developers' opportunity to have a say on the destiny of the requirement. This board provided a platform for conversation between product managers and developers. They elaborated on how each requirement fit into the overall product. Through this collaboration, each side agreed on the requirement so that developers would stick with it during the follow-up delivery, even if the delivery eventually required some of the more mundane tasks such as testing, analysis, and bug fixing.

"I remember seeing for the first time that everyone was actually focused on what was on the right side of the board and not on the left," Patrick says.

What seemed to be happening was that finally the focus on finishing a task was preferred over starting a new task or the loss of focus on a partially completed one. As that was becoming the trend, the successful report reengineering stood better chance.

Once a card had successfully passed this round of acceptance, it was automatically transferred to the input queue of the Delivery Kanban board. Once on that board, all requirements would be sliced into smaller work

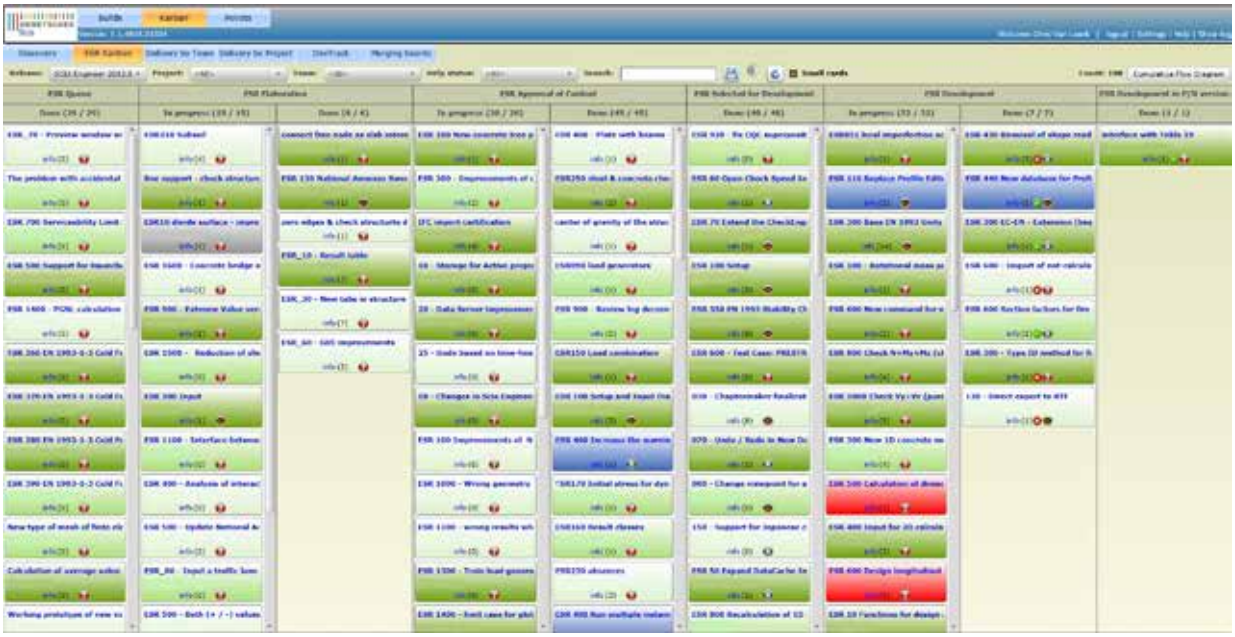

Figure 1.2 The Expert System Requirement Kanban board. Both the product and development teams elaborate requirements on this board.

items. Work-in-progress limits were introduced on the Delivery Kanban board that developers handled. As developers pulled a work item, they were encouraged to finish it before taking a new one.

The Development Manager ran operations review meetings together with the software development teams after each two sprints (each sprint was still six weeks). Together they observed the cumulative flow diagrams (see Figure 1.4) that were generated from the information each Kanban board provided, reevaluated the situation, and reestablished priorities if necessary. The big picture gleaned from the operations review meeting enabled vital improvements.

"Product Managers didn't have to waste time investigating whether everything was going smoothly and on time. They would just look at the Kanban board and instantly know. Instead, they could spend time figuring how to improve the product,

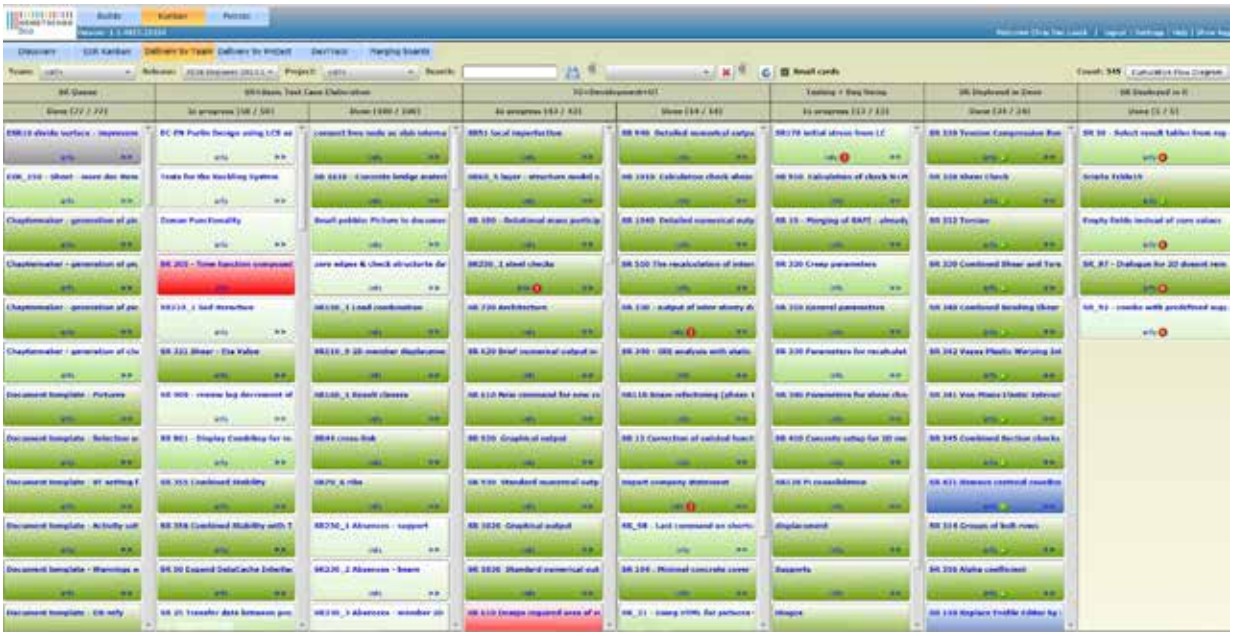

Figure 1.3 The Delivery Kanban board. It shows the work items in software development at different stages.

knowing how the code performs. This sort of first-hand information is invaluable for product managers. It can determine the destiny of a requirement," Patrick explains.

All these simple additions to the Scrum process were fixing the most urgent problems of Scia Engineer.

The Kanban boards had turned into the source for information through which smart, timely decisions could be made. The charter vision document had acted as the shield to continue improving by using the data from the boards. Both gave everyone the comfort to keep calm and focused and carry on with the reengineering.

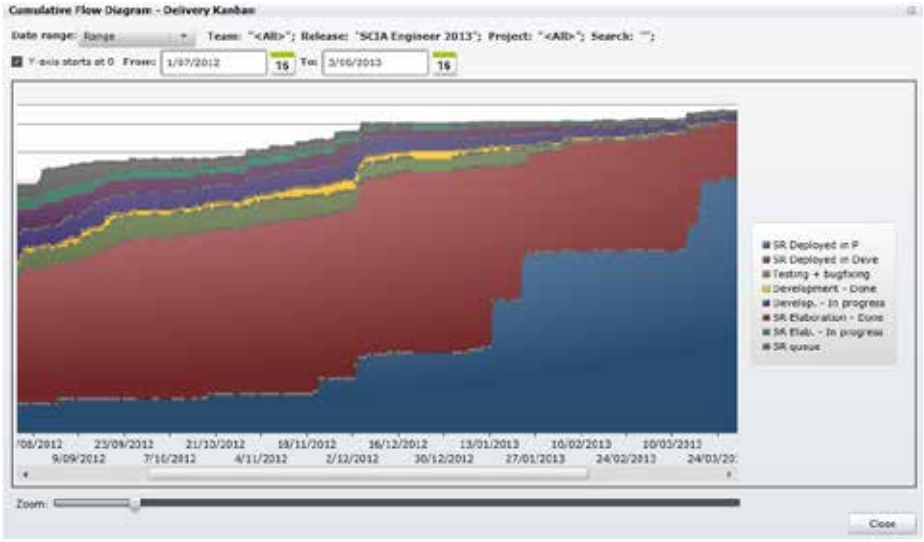

Figure 1.4 An example of a cumulative flow diagram that can be generated from the Kanban system. The unevenly rising red part represents the distribution of work that has been performed for the merging of a feature into the larger product. The merging has not been happening concurrently with the delivery but rather in large chunks, and only when it was critical.

Success!

In May 2013, the new reporting function was released to the customers of Scia Engineer. The incredible amount of work involved in this major commitment has been worth it. While the reaction of clients has yet to be witnessed, one thing is already certain. Scia Engineer and its teams have regained the sense of ownership of their product. They can take pride in their own abilities. They have regained the confidence that it is within them to be great and deliver a whole new Scia Engineer product. It is just a matter of time and a few more Kanban boards.

To learn more, contact Patrick Steyaert at patrick.steyaert@okaloa.com and follow him on Twitter at @PatrickSteyaert.

About Kanban University

Kanban University works to assure the highest quality coaching and certified training in Kanban for knowledge work and service work worldwide. Our Accredited Kanban Trainers™ and Kanban Coaching Professionals™ follow the Kanban Method for evolutionary organizational change.

Kanban University offers accreditation for Kanban trainers, a professional designation for Kanban coaches, and certification for Kanban practitioners.

Kanban University

https://www.edu.leankanban.com