Kanban University

KANBAN AT MOBILE.DE

Creating the Perfect Process, One Block at a Time

KANBANCASESTUDYSERIES

Apple's App Store users gave it a one-star rating. The first entry by the online search tool for buying and selling vehicles, mobile.de, on the portable devices arena was not receiving a standing ovation by any means.

Ralf Tomczak, mobile.de's CTO, stood in disbelief as he saw how many downloads were followed by user reviews far below the top score of five. "We knew our first mobile application could be better, but we never expected the users to be that harsh," he says.

The year was 2010; iOS application downloads had reached the billion mark, and users were already being spoiled by a growing talent force equipped to make their mobile experience fulfilling and meaningful. Mediocre applications were not simply being criticized; they were also tarnishing the images of the companies that released them.

A mid-September Friday in 2013 is Holger Hammel's last day as Development Manager of the mobile development team for mobile.de. He is moving on to his next assignment within the parent company, eBay Inc., leaving behind three mobile applications and a mobile version of mobile.de. The rewritten iOS app now has an average ranking of 4.5 stars. The team has created iPad and Android apps and a mobile website portal. The site carries an app-like look and feel and generates roughly 35 percent of the overall traffic for the service.

This is the story of how Holger and his team turned the odds with the help of the Kanban Method and Lean Thinking. In just two years they have managed to put the applications at the top of the mobile rankings for Germany.

Background

Holger first started working for the Berlin-based company mobile.de in 2007 as a freelance developer, and he later accepted a permanent position.

The Internet platform hosts more than a million offers for vehicles from both private owners and dealers. Used primarily in Germany, as well as in many other Central and Eastern European countries, it receives up to 2,000 queries a second.

The business was established in 1996 and was acquired by the giant eBay Inc. in 2004. Soon after the acquisition, the company grew and had to scale up to meet the consequences of increased user demand, as well as deal with the side effects of having a bigger team. The development teams realized as they faced these challenges that something in their process had to change.

Facing a major reengineering, mobile.de was advised by the German consulting company IT-Agile to adopt emerging Agile practices such as Scrum[1]. At the time, companies worldwide were reporting improved delivery rates for software and significantly greater commitment from team members through the increased involvement and empowerment such Agile practices provide.

Scrum helped the teams become more effcient, but a few things seemed to be missing in the new Agile reality: the flow of ideas, a picture of all ongoing projects, and the important combination of both a viable plan and predictable feature delivery.

In the search for something that could address all of these, Kanban Method practices were introduced on top of Scrum in the following years. With time and the help of the external coaches from IT-Agile, the practices of visualizing the knowledge work and instituting process changes to make it flow more smoothly—both on the team level and on the portfolio-management level—were successfully implemented.

By 2010, with an already reengineered browser experience, Ralf and the rest of the senior-level managers decided to pursue the mobile domain and extend the company's services in that direction. At the time, it made more sense to hire an experienced external agency to produce the application rather than build a team from in-house developers, as they would have had to learn the platform first.

In 2010, Apple's iPhone and its mobile App Store were far more advanced than any others in the market, so the choice to go for the iOS platform was obvious.

"We had objections toward the visuals and user experience of the

[1]Scrum is an agile software development model based on multiple small teams working in an intensive and inter-dependent manner. There are a few defined roles, such as product owner and Scrum master, and all the teams have to work in set timeframes, or iterations.

realize how weak the underlying technology of the proposed application was," Ralf recalls.

Experienced mobile users, curious and enthusiastic for the product, quickly found an array of glitches in the app. Ralf soon realized that no shortcuts were allowed in mobile.

"Seven million people use mobile.de. We had to face the fact that many of them were now wanting to look for their next car on their mobile devices. If we wanted them to continue using mobile.de, we had to provide a smooth experience for that," Ralf says.

The mobile application needed to carry the spirit and philosophy that made mobile.de so popular. Internal developers knew best how to do that. Learning and adapting to the mobile environment was the easier part of the equation.

"We figured out that the only chance for a good mobile application to pop up was if a team was created and nurtured in its own space with only that priority in mind," Ralf explains.

During this time, Holger was a software engineer on another team. Through the years he had developed a particular interest in the Kanban Method. While he was on a Scrum team he noticed other teams that visualized their work on whiteboards referred to as Kanban boards.

Those boards were divided into columns that represented the individual process steps that a piece of work underwent from input and ready-for-development to deployment and release. The work items were written on colored post-it notes and put on the board.

Holger saw his colleagues on the other teams attend stand-up meetings every morning. He overheard them discussing tickets, especially the ones that were not moving along the board as easily as others.

Holger paid attention and saw something particularly interesting: Nothing in the process stayed the same, and over time, many things changed. Names of the columns, numbers of tickets per column, people present at or moderating those daily meetings—nothing was ever a constant. He loved the idea of such freedom. His colleagues told him that with each change the process became better.

He was particularly interested in seeing that even to an outsider like him, all these things seemed easy to spot. It was the inherent transparency that Kanban provided that gave such detailed insight.

He began reading more about the Kanban Method and became fascinated with another aspect of it: Some argued that estimating delivery time might not be necessary. Delivering features in a reliable and constant flow seemed to be much more vital than estimating. How much time that actually took was a matter of measuring the progress of tasks in real time. He had tried to convince his team leads to try the approach, but they never agreed, thinking that it interfered with the underlying principles of Scrum.

The Beginning

The mobile development team was established in 2011. Holger was assigned as the Team Lead. Initially there were two developers, a Quality Assurance engineer, and a Product Owner.

"We were all very experienced in building a high-traffic e-commerce website, but we knew little about how to do mobile applications, and it was certain we would run into pitfalls and issues along the way where the answer could not be found in the codebase. We might have had to spend the first few months outlining ideas and preparing a solution, but such preparation felt like a complete waste," Holger says.

The project had strategic value for the company, not only in economic terms, but also in terms of proving mobile.de's ability to be an innovative and fast-moving company. The team had to start developing immediately.

To ensure their endeavor, they decided to rely on the Kanban Method. They believed the right process would stem from the help of transparency—visualization of invisible work and the workflow process. Such transparency would also help in identifying blockers and dealing with them in a timely manner.

"I believed we were capable of building the mobile applications without too many plans and documents in advance. I saw these as constraints and overhead, so we removed them from the start," Holger says.

All the communication that is woven into Kanban—the daily stand-up meetings, the planning sessions, and the consequent retrospectives—would give the possibility for quick resolution of problems, technological or process related.

"This was our chance to experiment with Kanban and we took it," Holger says.

Such an approach had its risks. Holger and the team were ready to take responsibility for those risks. The question was, would the managers above agree?

Projects in mobile.de were usually executed with clear milestones and accountability for the investment of resources. Previous Kanban Method adoptions had improved delivery, but they came on top of existing release plans and processes. The mobile team was the first ever to start from scratch with Kanban; they insisted on no iterations, no estimations, and no deadlines, just a constant flow of communication, learning, and delivery.

"It did require a leap of faith that this team, with no track record, would succeed from the start, without metrics to hold them accountable, using just a Kanban board," Ralf says.

The team made a commitment to experiment with various solutions to each of the design and software architecture problems they came across until they found the best solution.

To help the flow and reliability, individual work items were small, taking between half a day and two days to complete, and were releasable to the end users. The team's Kanban board had just two columns—planned and ongoing. They all agreed to spend five minutes together in front of the board every morning. There was an additional one-hour planning meeting every week to discuss and roughly prioritize upcoming user stories.

In the first year, the focused team solved many problems such as what backend engine should drive the future mobile applications and whether or not existing APIs should be part of the framework. The bitter taste of the low rankings for the initial iOS app remained, and the backbone technology had to be impeccable. With such a small team, it was easy to have focus, discussion, and consequent resolution. In the case of the backend technology, the choice was obvious.

"It might have taken us less time if we had selected an existing framework and made it work for us. We all shared high expectations that the user base would grow exponentially. Without a tailor-made backbone to meet that, chances were that the servers would crash and annoy users," Holger explains.

Within a few months the backend was built.

With the help of external developers from IT-Agile, the team tried to salvage the iOS app. They realized it had to be rebuilt from scratch to make it user-friendly, stable, and maintainable in the long run.

Part of the team took on rewriting the iOS app and a little over a year after the team was first assembled, many iPhone owners in Germany had a much better way to look for their next car. But the endeavour was far from over: a new mobile platform was growing hungry for apps.

The Android App and the Epic

"Do you actually think this is enough functionality to release the Android app?"

The derogatory question from the business people stung Holger and caught him unprepared. By 2012 the Android OS had already gained significant momentum in the marketplace. More and more smartphone brands were adopting it, and the user base was substantial. It was the next logical step for the mobile team, so more developers were added to the team.

They were given time to learn the Android environment while the user stories were added to the team's board. So, when the team presented what they believed was the completed first version of the Android app to the business people, they were taken aback by the negative reaction and dissatisfaction.

"But you could observe what we were building all along and you never said anything to indicate it was not enough," Holger replied.

From the very beginning, the team tried to avoid the label "project." A project meant plans, documents, and expectations, all of which was unnecessary waste in a system whose main goal, after all, was to create a working mobile application. They believed that a user story on a ticket gave the "who," "what," and "why" of a requirement in a simple and concise way and that it ought to be clear to everyone. No additional layer of organization was necessary.

Now, for the first time, they began to doubt their assumptions. The team was coming across a blockage in the system that was not the usual pink ticket, like a missing requirement. The problem was bigger, and they wondered if the team's isolation from their internal clients had something to do with it. In their detachment from the business people, the team had lost one of its main sources of validation that what they are doing was right.

After some discussion, a new definition of the Android app's first version was cleared. The necessary additional functionalities were turned into user stories, which were written on sticky notes. Everyone wanted to make sure that such misunderstandings didn't occur in the future.

More meetings or definitive plans were certainly not the answer in an industry where technology developments changed constantly. To find a resolution, Holger went to the one place that he believed held the answer— the Kanban board.

He looked at the columns, he looked at the colorful array of sticky notes arrayed in those columns, and he wondered about the answer. He remembered how he had looked at other teams' Kanban boards, studying them with curiosity. And then it occurred to him. What the team was working on was there; nothing was hidden, but it was not graspable to an outsider.

User stories alone were small enough to create the sort of constant flow he had wanted all along, but they were not big enough for someone outside to get a big picture of what the team was working on. In his drive for flow and freedom of process he had sacrificed context.

The team never stood a chance to receive timely feedback on key functionalities for the Android app, or for anything else for that matter, because the business stake-holders didn't understand what was going on.

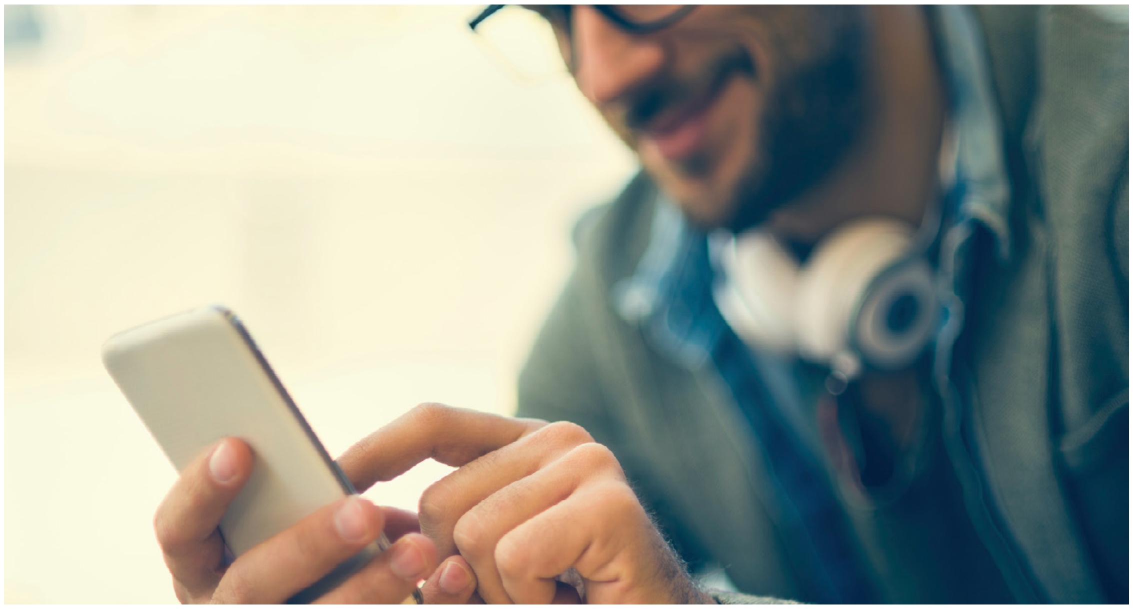
Soon after, the notion of an "Epic" was introduced and a place for it was designated on the board (Figure 1). It represented a bundle of features, all connected by a common idea, and was named such that everyone could understand what it was and what to expect of it.

When formulating an Epic, the team tried to make sure that building it would not take more than two or three weeks. The user stories within it would still be in a developer's language, such as "Input field for first registration date," but the Epic would be called something like "Create form to insert a car by private clients," which was language universally understood by anyone at the firm.

"Initially, we did it to help management understand what we were doing, but with time, the Epic helped our thinking on how to drive demand," Holger says.

Eventually, the Epics also proved a way for the team to control their work-in-progress and multitasking—the rule was that a developer should be involved with user stories only in the context of one Epic.

The Epic, in addition, became a success and validation metric and a way to evaluate the return on investment, as it represented a foreseeable addition to the product.



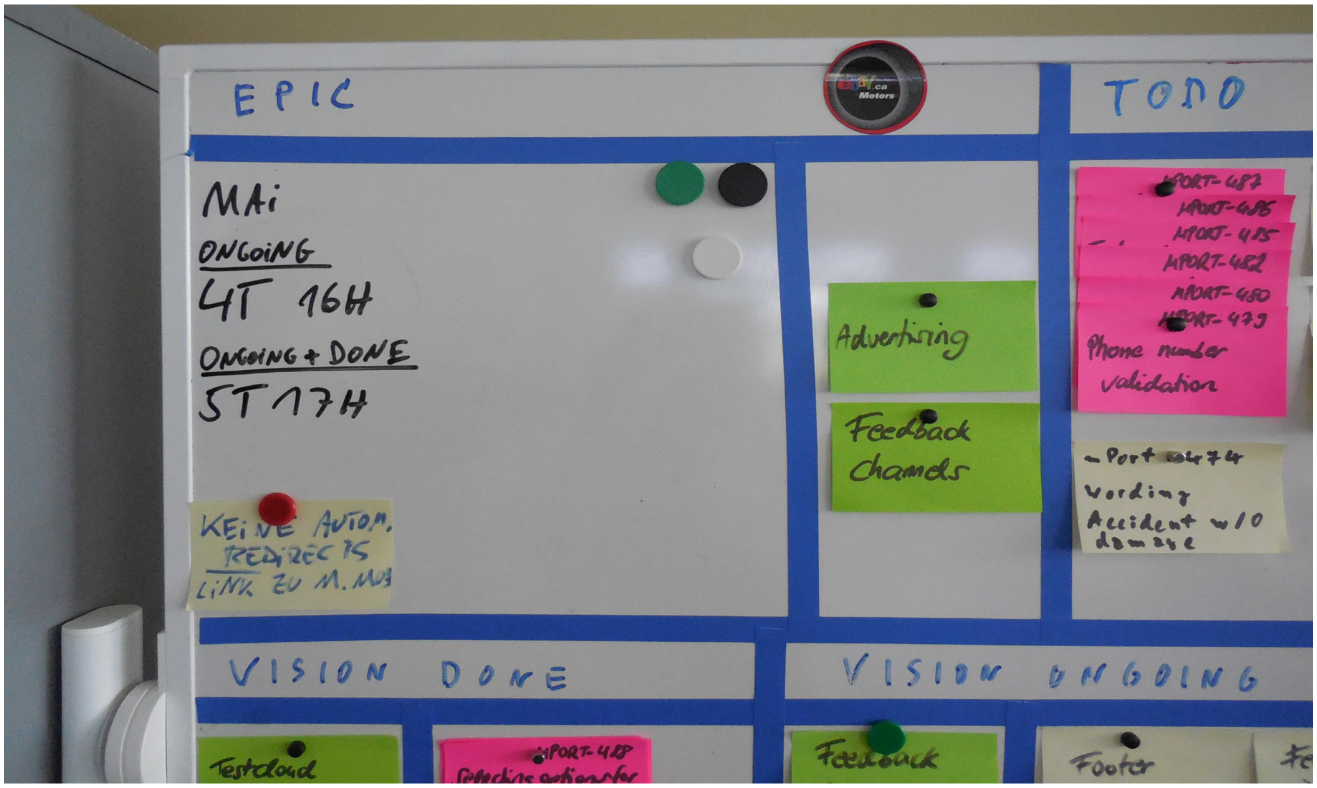


Figure 1 - The column designated for Epic tickets. Once a ticket is here, it has to be split into user stories or have the already collected user stories visually associated with it. The next time the Epic ticket moves is when all of its user stories are completed; then the Epic goes up for Validation.

Torsten and the Vision

"You should feel the pain of the problem and work to resolve it."

That piece of direction marked the first few months of Torsten's tenure at mobile.de. A front-end developer, he was the first who specialized in developing for mobile to start work on the mobile team.

"I came from advertising agencies and startups and had been used to the notion of just being productive. If something blocked my current assignment, I always put it aside and moved to the next thing on the to-do list," Torsten says.

That naturally had resulted in a habit of working on multiple tasks at once, some of which were forgotten in the haze of heated activity.

"During the first months I found it really hard to focus on one user story only, especially when it was blocked. Fighting the urge to break that rule was difficult," Torsten says.

Torsten's previous experience stood in the way of adapting to the principle of not leaving any tasks behind. Coming from a world that used the traditional software development lifecycle (SDLC) model, he was used to a phase-gate approach to progressing through the lifecycle stages of a product: First, the entire framework would be finished, then the whole UI; afterward, all the functionalities would be added, and in the end, all of this would be merged with the backend.

"At mobile.de, my teammates and I would work on a single functionality, ensuring it worked on its own, before moving to the next. In the big scheme of things, it did not make much sense, especially when I had to focus on a single user story even when it was blocked," Torsten recalls.

As the mobile team scaled up, the overview of the various applications and features in development was not as easy to follow as it once was. Without the larger picture, dealing with blockages first hand was becoming harder, too. As good as it had sounded just to develop in a flow, Epic after Epic, people needed a sense of direction and of what to expect to feed their motivation.

Holger went again to the place where the answers lay—the Kanban board. The issue was addressed during a team meeting.

"What do we need to do to improve?" he asked. "It would help if we had the perspective of where we are going, side by side with the Epics and the user stories," they agreed.

Soon after, the Vision—a higher-level, concept-phase column—was introduced.
By that time the team already had a third domain where it focuses its attention: the Mobile Web browser. Each of those domains had a different

vision because it was at a different stage of its development, it became evident the common board had to be split into three separate Kanban boards. The iOS (Figure 2), Android (Figure 3), and Mobile Web (Figure 4) boards all had a Vision Ongoing column that served as a preplanning phase.

By that time, the team already had its first product owner, who defined the creation of the Vision as his territory. Only Epic-sized tickets were put in the Vision Ongoing column, and after careful evaluation from the product owner, these were moved to Vision Done. From there they would be moved to the Epic column when a slot opened up.

The Vision helped Torsten stick with a user story, but so did the constant communication.

"I never felt left alone with the struggle of a blockage. We solved blocks together, discussing them in meetings. I developed a responsibility to show progress the next morning to the same people who had helped me with my problem the day before," Torsten says.

As he slowly changed habits, he realized that with the Kanban system he saw the fruits of his work much sooner. He did not have to wait six months for a product to be completed fully and delivered. Building the product in this leaner way, functionality after functionality, made the applications more stable in the long run.

Due to the fragile nature of mobile applications and websites, Torsten had often been frustrated with products crashing after changing and refactoring the code base.

"When I knew that I could add a feature without the chances of breaking the product, I felt more compelled to finish it, even if I was blocked," Torsten says.

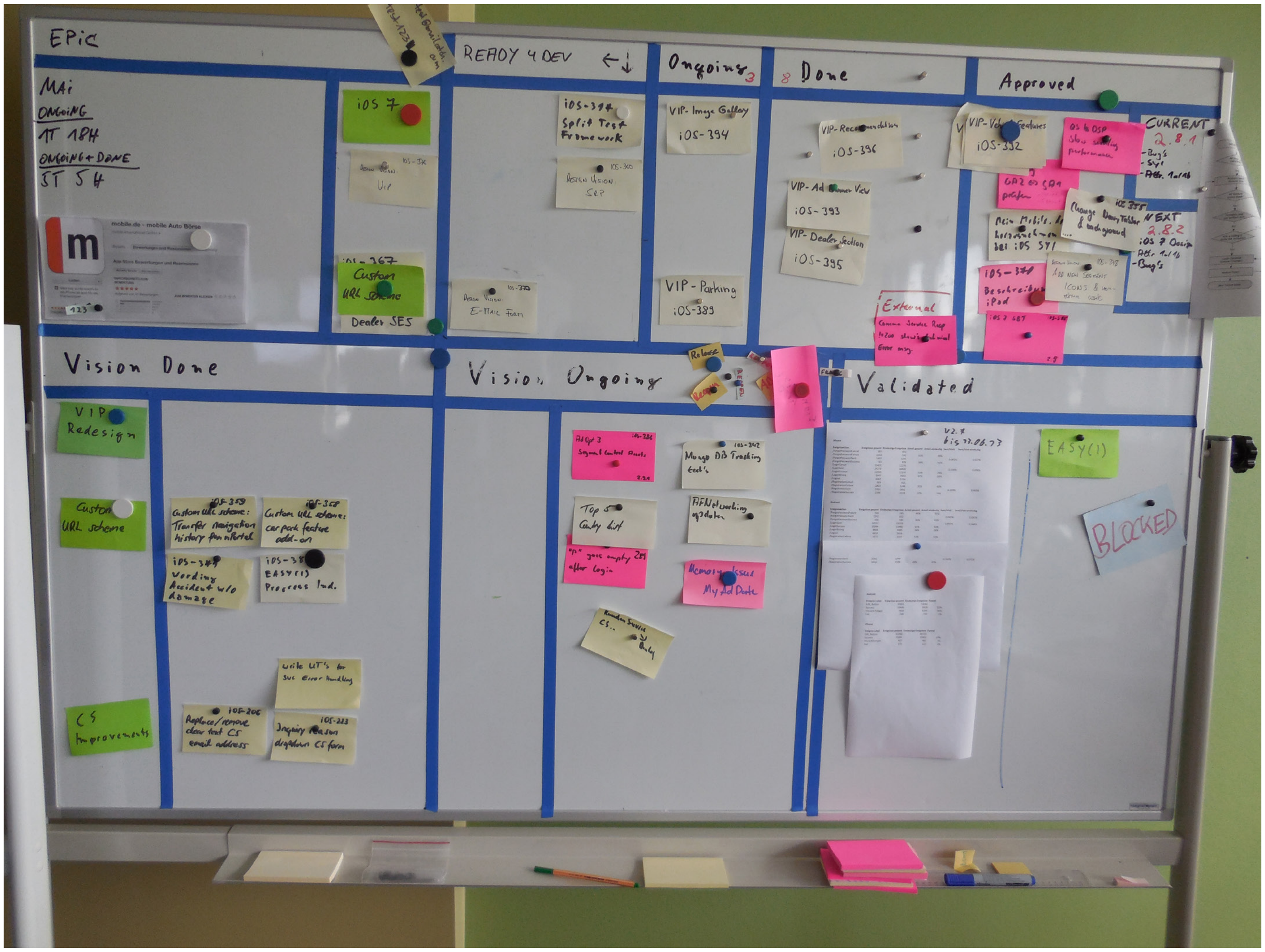
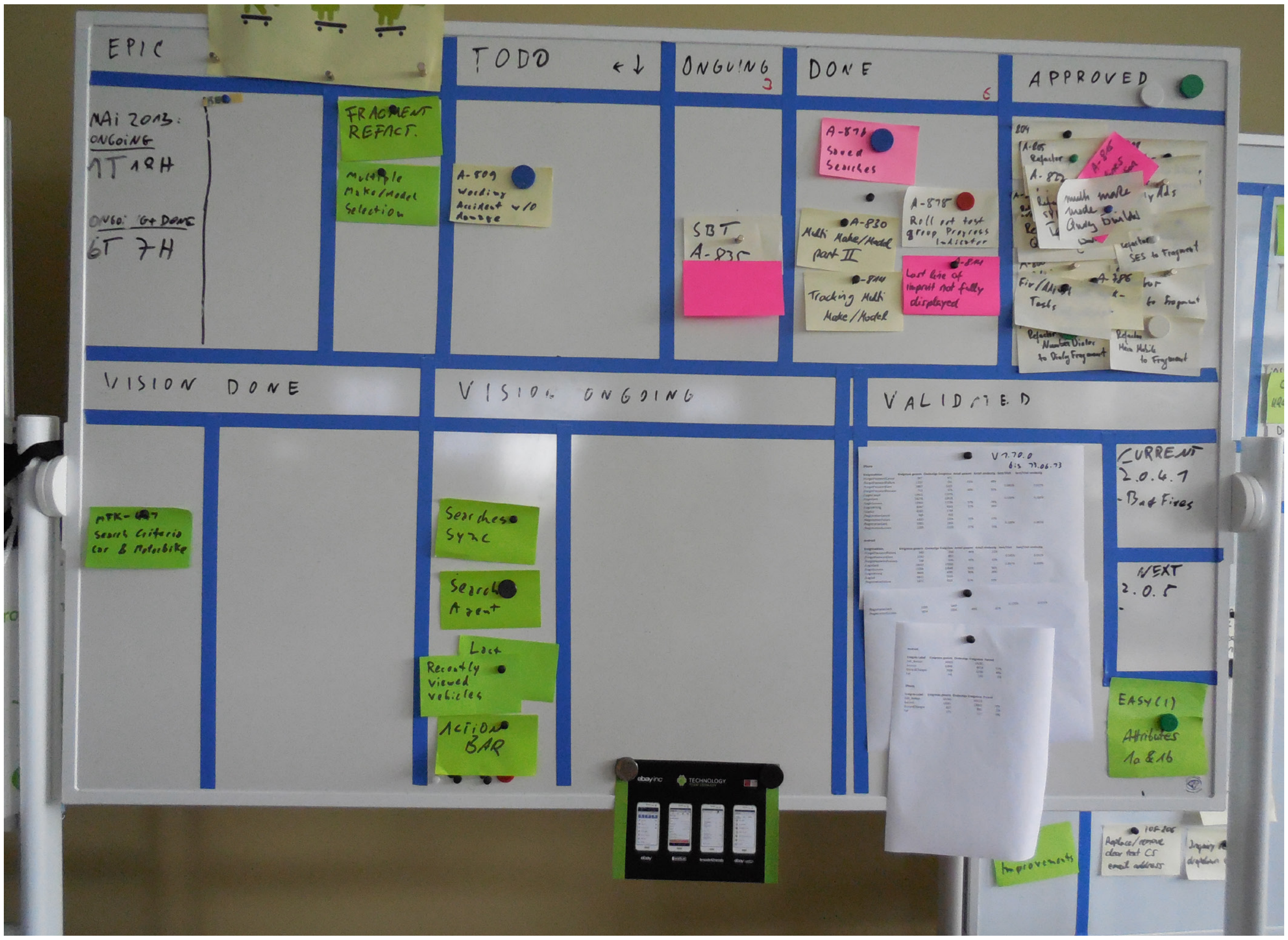


Figure 2 - The iOS team's board.


EPIC
MAi 2013:
ONGOiNG
1T 18H
ONGO. + DONE
6T 7H
FRAGMENT REFACT.
Multiple Make/Model Selection
TODO
A-809 Wording Accident w/o Damage
ONGOING
SBT A-835
DONE
A-878 Saved Searches
A-830 Multi Make/Model part II
A-878 Roll out test group Progress Indicator
Tracking Multi Make/Model
Last line of Impressum not fully displayed
APPROVED
VISION DONE
MFK-497 Search Criteria Car & Motorbike
VISION ONGOING
Searches Sync
Search Agent
Last Recently Viewed Vehicles
Action Bar
VALIDATED
CURRENT 2.0.4.1 - Bug Fixes
NEXT 2.0.5
EASY (1) Attributes 1a & 1b
Improvements
Figure 3 - The Android team's Kanban board.

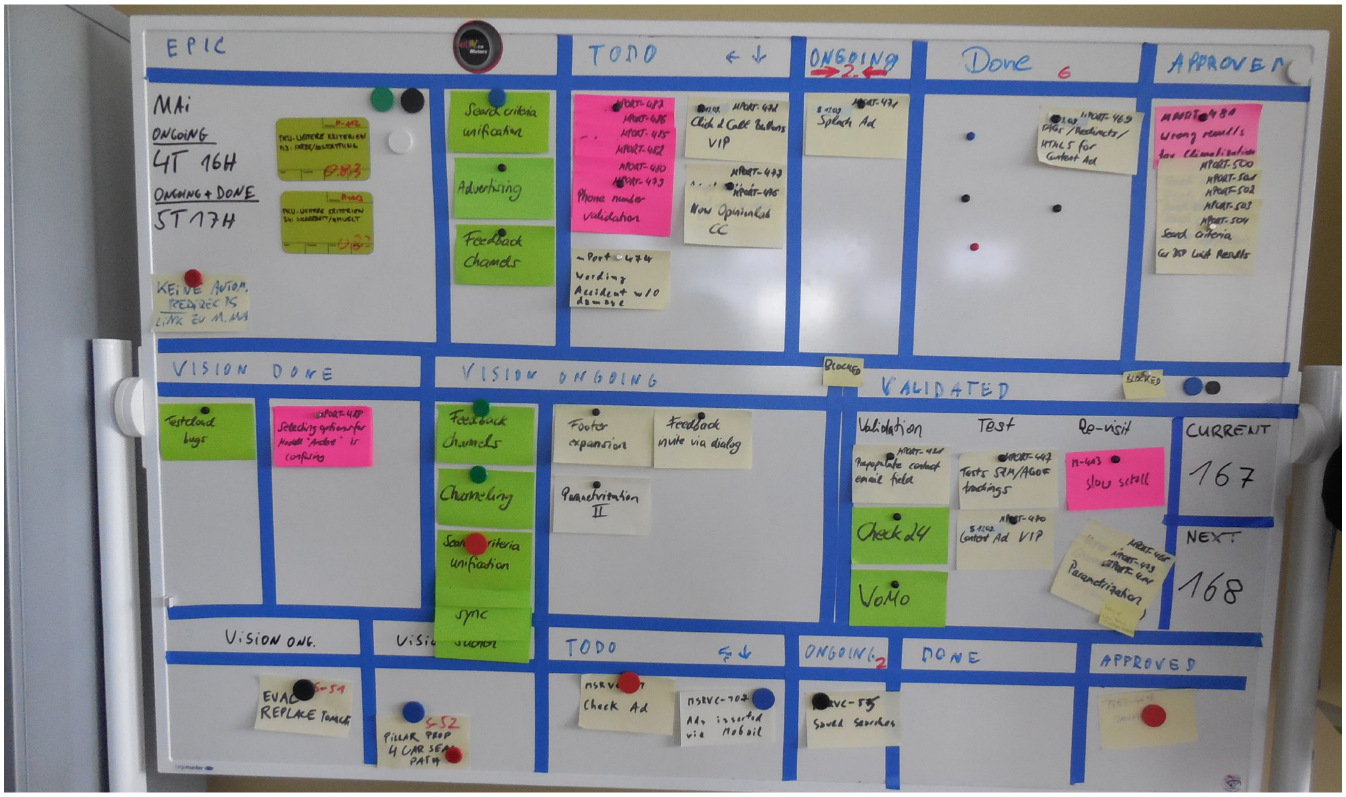


Figure 4 -The Mobile web platform's Kanban board.

The QA and the Retrospective

“We have always had this issue—more developers than Quality Assurance engineers. When the ratio became twelve to one, we were in trouble.” Holger says.

Although development had increased substantially, the mobile team still remained with just one QA engineer. As the team scaled, testing and quality assurance was on the verge of becoming a serious bottleneck. Developers were used to handing in their user stories to QA for testing, but soon the flow was disrupted and the team could no longer achieve the continuous deployment they desired. The question was whether to push for more QA engineers so they could go on the way they had been working before, or to make some other kind of change.

“We sat down during a retrospective and talked about our understanding of the QA role,” Holger says.

Retrospectives had proven to be a valuable source for addressing issues that hindered the team. They got people to think about the problems and come up with solutions.

During consequent retrospectives, the team began to reevaluate the notion of shufflng work to the one QA guy. How fair was that? The developers were the original writers and creators of code, so how could they help resolve this bottleneck? As the team pondered on these questions, they agreed that something drastic had to change.

And it did—little by little, Harald, the Quality Assurance engineer, began to consult with the team about how they themselves could test more and therefore need less actual QA testing. He established practices for the team to be able to test their own work.

“Test cases[2] are not something that I as a front-end developer could create. I do not know the appropriate programing language. But what our QA did was to come up with frameworks that made writing a test very easy. Running them, I could see immediately whether I had engineered my user story well.” Torsten says.

Manual testing sessions were established, during which developers from one team tested the stories of another. Furthermore, the QA consultant was motivated to look for innovative, automated ways of testing.

With time, something else developed as well—the sense of responsibility for the quality of the code from the developers themselves. Whenever developers were unable to perform a test, they proactively collaborated with testers from other teams. This evolutionary change of the QA’s role helped to return the flow to where it had been previously.

In hindsight, the retrospective meetings resolved many other glitches the mobile teams faced. The daily stand-up meeting’s quality had deteriorated when the team scaled up. Without set policies, it often was chaotic and unfocused, oftentimes shifting to non-development-related matters.

After the problems with ineffective daily meetings were escalated during numerous retrospectives, new rules were put in place. The only people who were allowed to speak during the daily were the developers. Questions from other participants such as the product owner, Holger, or anybody else had to be taken privately afterward.

[2]A test case represents a set of conditions or variables under which a tester will determine whether his or her work item is functioning as it was originally intended. Its components describe an input, an action or event, and an expected response, will help to determine if a feature of an application is working correctly.

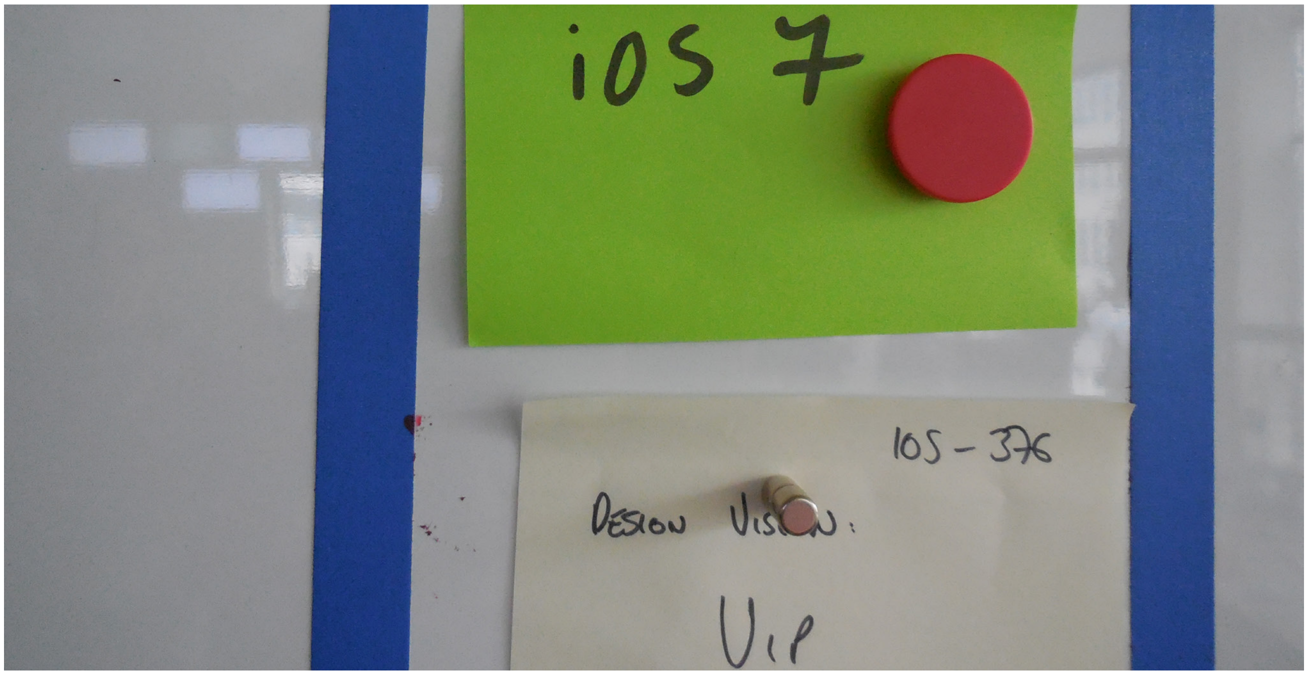


Figure 5 - The Epic ticket for the iOS 7 version of the application is already on the iOS team's board.

The Lean Experiment and the Café Tests

"The next project should be an iOS app for mobile.de car dealers," the business people said one day.

It was February 2013 and the team was skeptical that the need existed. At that time, the mobile development team already had an iOS app for iPhone and iPad, as well as Android apps in German and in English.

"We are sure the dealers need a service to help them keep track of inventory," the argument went.

By that time two-thirds of all car dealers in Germany were active users of mobile.de. The mobile team's process had smoothed out to a large extent, and Holger had more time to focus on non-process-related issues.

Reading a copy of The Lean Startup,[3] which he and all of his colleagues were given by mobile.de, was one such thing. He was fascinated with the argument for early validation of the hypothesis of customer needs. The process to do that was simple—build something as minimal as possible that can still function as a viable product, or a minimum viable product (MVP), as it is known.

This MVP should be taken out into the field for testing with example target customers, even if the product has many defects. The purpose of the test is to see whether these target users find the application interesting and valuable.

Instead of arguing with his colleagues from the business side, the product development department cleared some budget and time for a field trip to Frankfurt to test the proposed new application and validate the business model.

A group of user experience (UX) experts, product owners, and developers volunteered to do the experiment. Spending just four days and starting off with a paper prototype, the five explorers visited the car dealers' capital. They knew this timeframe would be enough for the necessary iterations to create the MVP on the spot. They split each day into four iterations and each time presented an improved version based on the feedback they had received from the previous iteration.

After two days, a click dummy was created. After a few more iterations and some pivots, it became clear that they had found valuable features, but that most likely the business model would not work. In those four days, all participants received so much valuable feedback about the existing application that they came back to Berlin with a bagful of user stories.

Inspired by this experience, team members Lars and Max (product owner and interaction architect, respectively) set up lean and frequent validation experiments.

For instance, once a week Max takes his laptop and works from cafes around Berlin. He talks to strangers there and asks them to play around with a prototype. The quick user feedback, proper metrics, and consistent validation of features fundamentally changed the way the team develops their products.

[3] Ries, Eric. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.* New York: Crown Business, 2011.

Conclusion

In 2013, three years after the initial mobile application was created, the mobile team's developers have learned not only the technology of mobile, but they can now focus on experimenting with their ideas and seeing users' reactions. Developing what people want has become the driving force in the ideation phase of their process. The flat design of iOS 7 is one of their next challenges.

"We want to find out if we can design the new, flat look of our application iteratively, rather than as a full redesign that we release as a big bang. This will create a visual inconsistency, which users may not tolerate for some time, but it will give us an early idea of what users want to see and use. The risk is worth pursuing," Ralf says.

He has learned the lesson from three years ago, and he wants to make sure that the new mobile.de iOS 7 application is an exquisite actor on the mobile stage where nowadays almost a million other applications seek time and attention from mobile users. He feels safe because by now he knows very well what to expect from the team.

With the Kanban Method's principles and practices in place with the mobile team, the next great feature is just a few experiments away.