
LEAN SOFTWARE DEVELOPMENT WITH KANBAN
DEVELOP SOFTWARE 3X FASTER
DIMITAR KARAIVANOV, CEO AT KANBANIZE

Why should you read this case study?

For the past 28 months, we were able to push out **26 new releases**, with just two gaps in December, when we celebrate Christmas. As a matter of fact, the versions were there, however, we decided against releasing them because it might have meant very few people would make use of the changes during the holidays.

With an average of six new features per month, we have released **150+ new features** to production. The living proof appears in our Kanbanize Blog (make sure to subscribe for updates).

As of this moment, at Kanbanize we have a total of **one (1) open customer issues.** Yes, this is not a joke, we have almost no open customer issues for a code base exceeding a million lines. Of course, there are issues that we are just not aware of, but out of the reported ones, we've fixed them all.

We have around sixty open internal issues, most of which are low priority. We are depleting these quite quickly, so in around four months we expect to have **zero open internal issues as well**.

We managed to create an eight-digit company in three years and effectively multiply the initial funding **by more than 30 times.**

The numbers show that we must be doing something right and this book talks about the engineering part of this modest success.

Okay, let's do this!

1. The Beginning

I co-founded Kanbanize because I envisioned the exponential adoption of the Kanban method some 6 years ago (the name is not a coincidence). Back then (2010), I became a Lean champion at a big German company. We were a team of Lean thinkers helping the RnD department (and to some extent the entire company) change course from Waterfall to Lean and Agile delivery methods.

To this day, I am not sure how I got this role and to be quite open, I have no idea why Christoph chose me before other much more experienced managers in the company. I guess it was a stroke of luck that would later prove even more significant than I anticipated at the time. Christoph came from the biggest German software company - SAP. He was famous for achieving outstanding results there and that's why our CTO invited him to take up the SVP position.

I met Christoph for the first time a couple of months after he joined the company. We talked about my team, which was responsible for the performance of the product suite. I remember showing off the good results that we had got during the past two years, but he didn't seem too impressed. I realized later that he was probably not that interested in what we were actually doing, but focused on assessing the people he could count on later. He had plans that none of us could have suspected.

His plans turned out to be simple but revolutionary for the situation that we were all in. He was on his way to establish a promotion process for the entire product suite (20+ products working together). This basically meant the following:

1. Each product team had to have a nightly build and as many automated tests as possible.
2. All product builds had to be integrated each night and tens of thousands of automated tests were executed against the entire product suite.
3. The teams were not allowed to work on other things unless their builds and tests ran smoothly. Even if a single test was failing, the issue was marked as a blocker for the whole suite and had to be resolved with the highest urgency.

Now, I want you to evaluate the chances of success of this approach, while keeping in mind that it took some of the teams more than a month to build their products. That's right - the last successful build for some products was more than a month old and these same guys had to not only build every day, but also run automation tests, none of which could fail. Quite revolutionary indeed.

In the beginning, people thought Christoph was not serious about the promotion process and even ridiculed his approach. However, they soon realized that he wasn't joking and started working hard to get this whole thing done. It took everyone a lot of time to get there, but after an

year things started to look better. We had a well-oiled promotion process of consistently producing stable and well-integrated suite builds, representing 20+ products working together.

A lot of success followed. We were able to release a super high-quality release on time and that had not happened for many years prior to the changes. We started adding more tests to the promotion process - performance tests, upgrade tests, installation tests, etc. In my team we were able to detect a performance regression on the next day after the release was introduced. Just think how many companies have performance tests in the first place and then think how many of those companies can detect regressions throughout the promotion cycle of 20 products. We were nailing it down every single day and, in fact, there were a lot of performance bottlenecks fixed without even raising promotion blockers for them. This was a sign that the product teams could actually spend time improving the product's architecture and not just put out fires, as they were used to.

Watching all of this from a first person perspective and actually being an active part of the transformation got me really fascinated. I was amazed by the effectiveness of Christoph's ideas. That got me further into Lean and I started to develop and test my own ideas in the teams with which I worked. That's how I learned about Kanban and this was the element that changed my life for good.

...

Meanwhile, one of the main initiatives that we had with the Lean Thinkers team was the "Feature Management" topic. The problem there was that there were many user stories distributed across 20+ teams and there was no reasonable tracking as to of what was happening on the feature level. In other words, only the corresponding product manager was partially aware of what was happening in the team and there was absolutely no way to figure out the progress outside this group. Apart from that, there were many cross-team features that were usually delayed, due to the lack of synchronization and focus holding back some of the teams.

What we accomplished was to establish a mechanism that would allow us to track the overall progress of all feature work for more than a dozen different products, split into five investment areas. This mechanism would allow us to report weekly on how many features have been completed, how many were being worked on, how many user stories were in progress, etc. all of this distributed by investment area (management wanted to know where money was being invested).

All the data was presented using a monstrous excel spreadsheet, which was generated once a day from a central database. There was a working student whose primary job was to get the data extracted from a tool starting with J and then get it imported into the central database, from which the spreadsheet was seeded. Then, we would use this spreadsheet to run our weekly meeting, during which we would come up with observations and suggestions for improvement.

There was also a separate meeting of all VPs of the corresponding investment areas to discuss the feature status and take actions to unblock given features, if they appeared stuck.

This solution worked really well compared to the chaotic environment in the past, but, of course, some things were far from perfect:

- Product managers lacked the visibility into what was happening in RnD and were always unhappy, because “the development teams were very slow”.
- There was no mechanism to stop teams from starting new work, which caused a dramatic increase of the features that were started, but not finished. At one point we had to forbid starting new features so that we can get at least something out of the door.
- The features that had to be worked on by more than one team were frequently left behind and special synchronization efforts were always necessary.

Being involved with the Feature Management initiative, while at the same time experimenting with Kanban in my teams, made it really easy for me to see that using a Kanban system on the global feature level would solve a lot of problems (and expose a lot of new ones). Had we implemented such a system, we could have:

- Ways to manage work in progress limits so that we achieve better flow and eventually better throughput from the same people and resources.
- Transparency into the current progress, thus reducing status reporting and actually making the reports much more accurate.
- The cycle time and block time metrics to support a Kaizen culture. Kaizen is only possible if you have baselines and ways to track changes, be it positive or negative.

Pursuing this idea further, I started searching for better alternatives that would allow us to map this whole management process. Looking at the different tool offerings, it became evident that tooling support was years away from what I envisioned my organization needed. That was understandable. After all, we were quite innovative with what we were doing and tooling had not yet caught up.

That’s how Kanbanize was born – out of necessity. I was lucky to have Christo, a childhood best friend of mine and now CTO and co-founder of the company, working on the first version. He came up with a prototype after six months of work and I went ahead to demo it to the Lean Thinkers team. I actually proposed Kanbanize as the official feature management instrument in the company, but the project didn’t fly. It’s just that the company needed much more than we could offer with the first immature version of the product (this was more than 5 years ago).

However, Christo and I could recognize the potential of this solution and we could probably see what others didn’t, so we decided to quit our jobs and completely dedicate ourselves to the dream to “Kanbanize the world”. This was, and still is, the best professional decision I have ever made in my life.

Getting an investment from a local venture fund (thanks Eleven!), on-boarding Biso as a third co-founder and quitting our lucrative jobs was the easy part. When we found ourselves in the situation with an ugly beta-version, just $125,000 in the bank and a total revenue of $3,000, we realized that we needed to act fast. We just had to find ways to make maximum impact with the least amount of money possible and make the company profitable. Oddly enough, this looks pretty much like the situation which Toyota were in when they started their automotive projects. They were a small Japanese company that wanted to build cars, but didn't have a lot of money and know-how. That is how they created Lean - they just didn't have the luxury not to.

Full of fear and hesitation we continued with our endeavor to Kanbanize the world. We became a team of five guys, the three co-founders and two employees. We were getting some traction, but it was far from what we needed. That's when Christoph came back into the picture.

One day, I got an unexpected call from the HR manager of the office in Sofia. She told me that Christoph wanted me to take the position of a Managing Director (the position from which Biso had resigned in order to join Kanbanize). I declined right away, because I was fully dedicated to Kanbanize, but I felt I should thank Christoph for this generous offer. I sent him a short "Thank you" email. He replied. I replied again, and a couple of months later he became an investor in the company. After all, it was a company conceived on the basis of his ideas, so his place on our board was a well-deserved one.

Shortly after that, things just started to happen and although it was not easy, the world is a much better place now. But if there is one thing that made us succeed, it is the Lean Thinking that we've mastered so well. My goal for the case study is to convey as much of this "Thinking" as possible. Learning through experience is priceless, but if I could save you even a month of mistakes, I would consider my mission successful.

2. Literature

Reading "Lean Thinking" by James J. Womack and Daniel T. Jones was how I initially became acquainted with the concepts of Lean. The ideas it presented made so much sense that I started reading book after book on similar subjects. Many of the things we do today would not have been possible without some of the books I have had the privilege to read. That is why I would like to share a distilled list of "must-reads" with you right away:

- The Goal by Eliyahu M. Goldratt and Jeff Cox
- Lean Thinking by James J. Womack and Daniel T. Jones
- The Toyota Way by Jeffrey Liker
- Kanban by David J. Anderson
- The Principles of Product Development Flow by Donald G. Reinertsen
- Lean Product and Process Development by Allen C. Ward and Durward Sobek
- Lean Software Development by Mary and Tom Poppendieck

Find the time to read these books. They will be well worth your time.

3. Case Study: Lean Software Development at Kanbanize

Mastering Lean has been our task at Kanbanize for the past six years and, in this chapter, I am going to share what we've learned and what is still a challenge. This is going to be a very practical course and I hope you will find useful tips that you can directly apply to your work.

Before we start, I would like to quickly go through the reasons why Kanbanize was created in the first place. The main idea for the tool was to ease the process of breaking down bigger chunks of work into smaller pieces and to ensure seamless tracking not just on the user story (task) level, but on the feature/project level too.

The issues that Kanbanize was meant to tackle were and still are:

1. Lack of visibility into the progress of individual features (or projects).
2. Too much context switching between multiple projects or tasks.
3. Slow development processes and overall poor productivity.
4. Lack of mechanism to prevent teams from uncontrollably starting new work and therefore harm the overall performance of the company.
5. Inability of the product management to proactively work with the engineering teams due to the lack of real-time status updates.
6. Lack of actionable metrics to be used for continuous improvement (Kaizen) initiatives.
7. Lack of consistently up-to-date status reports available not just for the management but for everyone involved in the project.
8. Hard to use tools that were never meant to be used by product management, but just engineers.
9. Show what engineering is working on so that product management won't always think that developers are slow and lazy.
10. Product management engaging with individual engineers directly, due to the lack of visibility of the actual progress.

It took us more than ten years at corporations like SAP, Johnson Controls, Software AG, ProSyst and six years at Kanbanize to get to where we are today and it won't be exaggerated to say that we are still in the beginning of the journey. However, knowing how much effort and knowledge it took us to get to that stage, I would like to share some of our experience, so you can make the transition not in six, but in probably one or two years.

Quick Company Overview

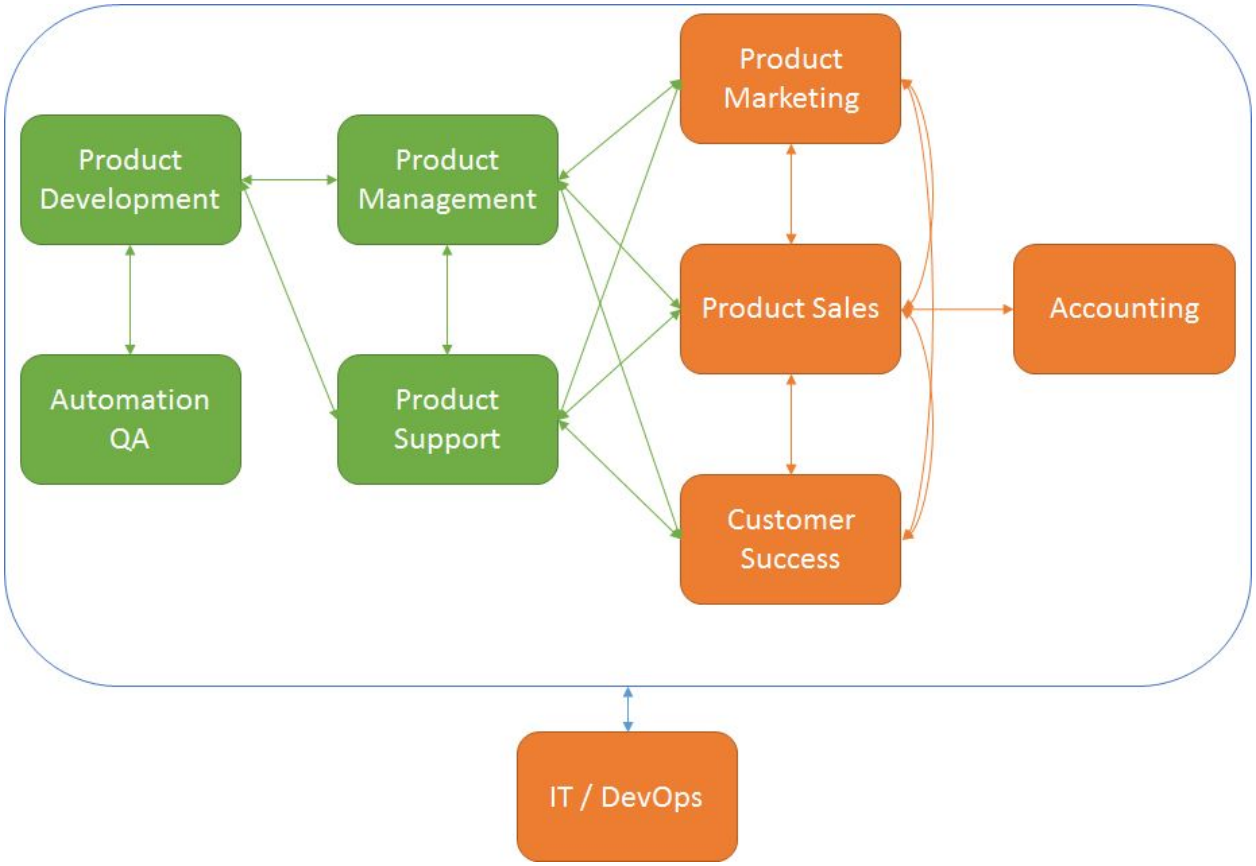
We are a product company and for the moment we are working on a single product, but we may change this status really soon. Our release takt time (how often we deliver value to our customers and how often they are ready to accept it) is one month, which means that we roll-out

new features to all our users every four weeks or so. Some of you may recognize a Scrum sprint here, but what we do is actually quite different. Instead of planning what we will be able to accomplish in a month, we do our best to accomplish as much as possible and, when the takt time comes, we just release what has been completed. This is a fundamental difference between Agile/Scrum and a Flow-based method, such as Kanban. I urge you think more about that, it may sound like a minor difference, but it is, in fact, a huge gap in productivity between these two approaches.

Our monthly release goals are:

- Zero open customer defects (achieved consistently)
- Zero high severity internal defects (achieved consistently)
- Less total internal defects than last release (achieved consistently)
- At least two major new features and at least two minor (achieved consistently)
- Zero regression defects in each release (work in progress)

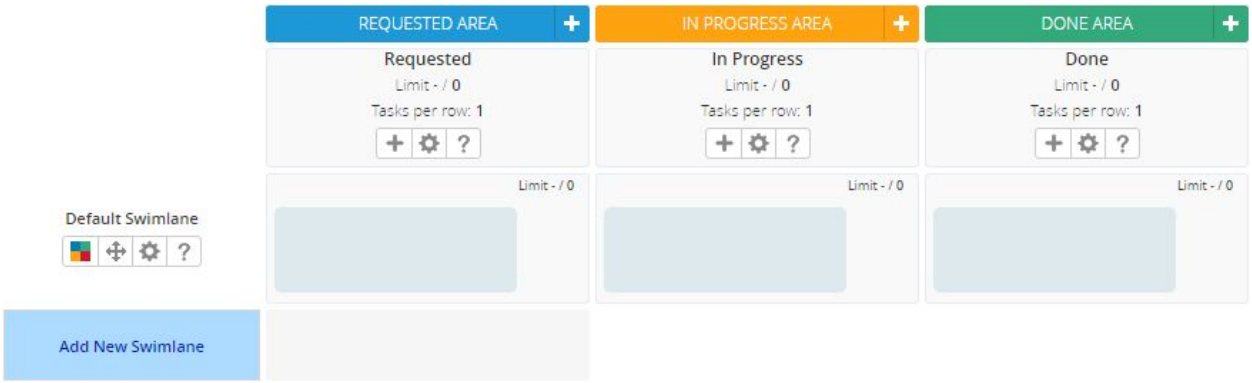
To give you a better perspective about the company structure, let us explore what service teams (or also services) we have in the organization and how they interact with one another.



Service teams (at Kanbanize)

The use of the word “services” is intentional. We do have teams in the company, but we think of them as services. A service has an input, processing steps and an output. In order to have all services working well together, we have established rules to control how services take input from other services, how work is actually being completed and what the output should be. We can compare our company to a computer program that has the different modules talking to one another via predefined interfaces. We do this despite the fact that we are all in the same office. The support person may be sitting next to you, but instead of asking them to do something for you, we insist that people create tickets in the Support Kanban board, which is essentially the input for support. The same principle is with each and every service in the company, be it marketing, sales or product management. This does not mean that we discourage face-to-face communication. There are many situations when filing a ticket and discussing it personally is the right thing to do and we do it quite often.

One feature we have in Kanbanize, which many users consider a limitation (at first), is the predefined sections of the Kanban board. If you try to edit a board in Kanbanize, you will see that there are three sections on that board - REQUESTED, IN PROGRESS, and DONE.



Requested, In Progress and Done areas in Kanbanize

Why would we force everyone that uses our software to have to choose which column belongs in which area? The answer is quite simple - each Kanban board is meant to be a service and each service has an input (REQUESTED AREA), processing steps (IN PROGRESS AREA) and output (DONE AREA). Having this separation, we can perform a much more sophisticated analysis of the data in the board and provide feedback to the users.

Unsurprisingly, all services within our organization use Kanban (and Kanbanize) to do their job. It is our universal delivery tool, which, integrated with email, turns out to be a really powerful system that can be employed to serve various goals. However, this book is about product development, so we will mainly discuss the services marked with green in the image above and only partially touch on the orange ones. Let us get started...

Product Management

As shown on the Figure above, product management is one of the central services in a product company. It is practically the bridge between product development and sales/marketing. Being a central service, product management has to take care of multiple initiatives such as:

- Articulating the product vision and strategy to all other services
- Capturing customer feedback and converting it to meaningful features
- Reporting on the current status to all interested parties
- Actively researching new technology to stay up to date with the new trends

So how does a product manager come up with the product strategy in the first place? How does she capture ideas, in a way that is easy to work with, without spending too much time maintaining huge backlogs? What can be done to easily report on the progress? What metrics are important?

These are all important questions and we will try to answer them all with concrete examples from our daily work. First, we will go through the backlog management, then we will cover the actual delivery model, which is in essence an Upstream Kanban implementation, and we will finish with a set of important metrics to monitor.

The Product Backlog Board

According to Lean, maintaining an ordered list of ideas (backlog) is waste. It is waste because working with hundreds of different ideas, always trying to refine them and prioritize them, takes time. If they never get implemented, this time is simply wasted. However, you still need a way to capture your ideas and customer feedback.

We do that with the so called “All Features” Kanban board. This is where we capture all the ideas that we come up with as well as the customer feedback we receive. The board structure looks like that:

Proj Mgmt and Analytics	Run-times and Automation	Links and Dependencies	Email and Collaboration	Ease of Use	Administration	Integrations	Process Governance	Mobile	Misc
High Priority									
Medium Priority									
Low Priority									

The Structure of the Product Management Backlog in Kanbanize

The explanation of the columns listed above:

- Proj Mgmt and Analytics - this column contains ideas or requests related to how managers track status and report on it. Also, this column covers the requests related to the analytics module.
- Runtimes and Automation - if you have used Kanbanize you should know that the runtime automation policies are the heart of our software. This is the way to automate your processes with the help of super-flexible business rules. This column captures ideas/requests regarding potential improvements in that area.
- Links and Dependencies - requests related to card linking and card dependencies. These requests usually come from project managers trying to implement a portfolio board (we will talk about that in a minute) or some sort of a schedule.
- Email and Collaboration - as the name suggests, this column covers the collaboration part of the tool. This is mainly the Email Integration piece and the Moxtra collaboration module (chat + screen sharing meetings) with which we are experimenting.
- Ease of Use - this one is easy to figure out. Here, we file requests that do not affect a concrete area of the software but, rather, the overall user experience and usability.
- Administration - again, the name speaks for itself. This is where we keep all requests related to administration.
- Integrations - a lot of companies integrate Kanbanize with their own systems via the developer's API and this is the column for such requests.
- Process Governance - everything related to permissions or implementing concrete behavior in the system is kept in this area.
- Mobile - work related to the iOS and Android Kanbanize apps.
- Miscellaneous - anything that does not fall in some of the categories above.

This is how the actual board looks in Kanbanize (this is a huge board. In order to get a sense of it in its entirety, it has to be minimized to the point where nothing is readable, sorry).

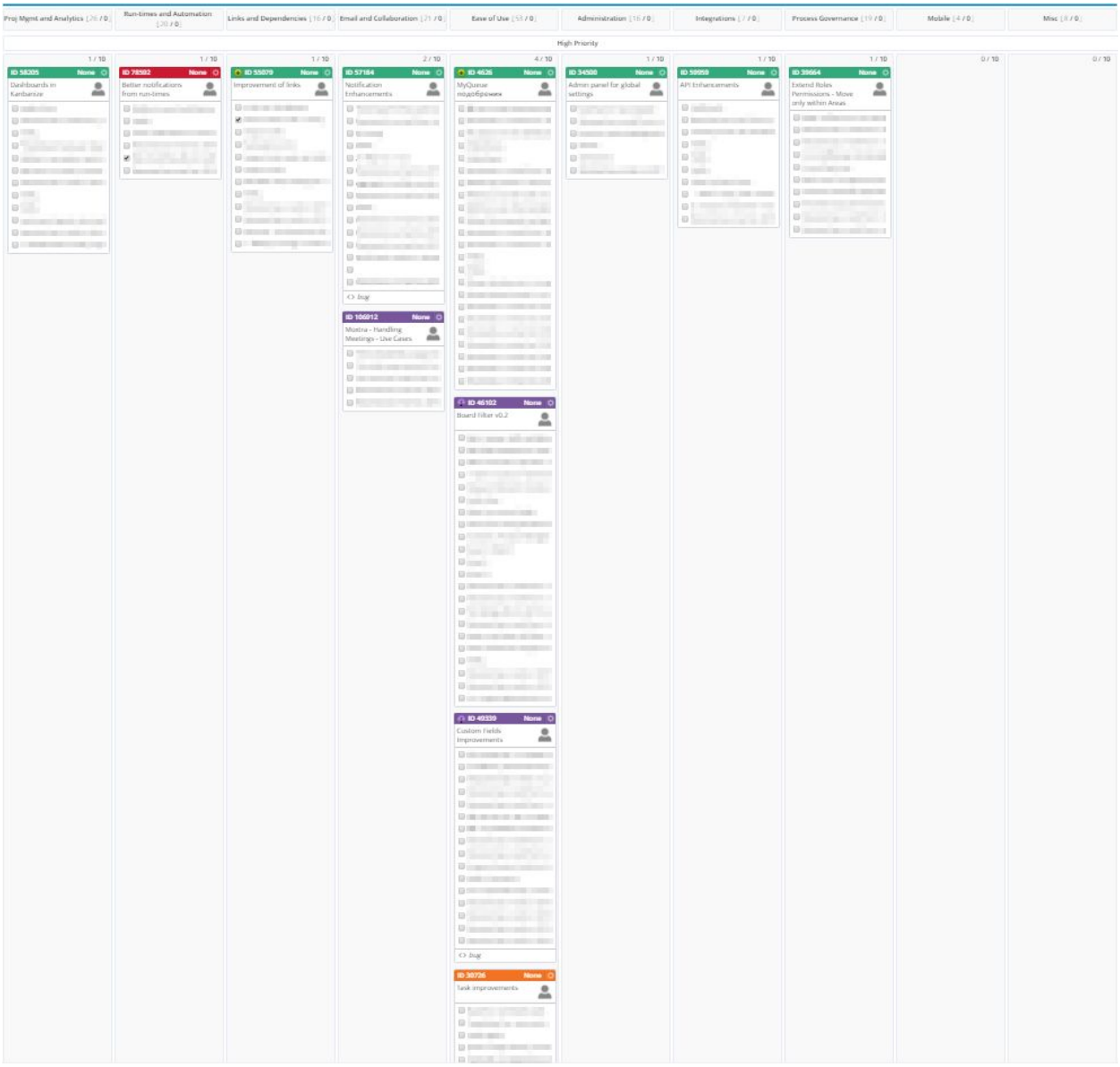
The Product Management Backlog in Kanbanize

Let me ask you a question. Can you point out the area where we get most requests/ideas for? I bet you can. Can you point out the second and the third? I think you can do that quite easily too.

You must be **seeing** the benefit of mapping your backlog in such a **visual** way. The keyword here is seeing, because you can get a lot of meaningful information out of the board, despite the fact that you cannot read a single character. Visualization of work is the first core practice in Kanban and this example is just further proof that visualization is a really powerful tool.

Another effective approach, which is quite specific to the use of Kanbanize, does a really good job for us too. It is a sort of voting system that allows us to immediately see which requests are

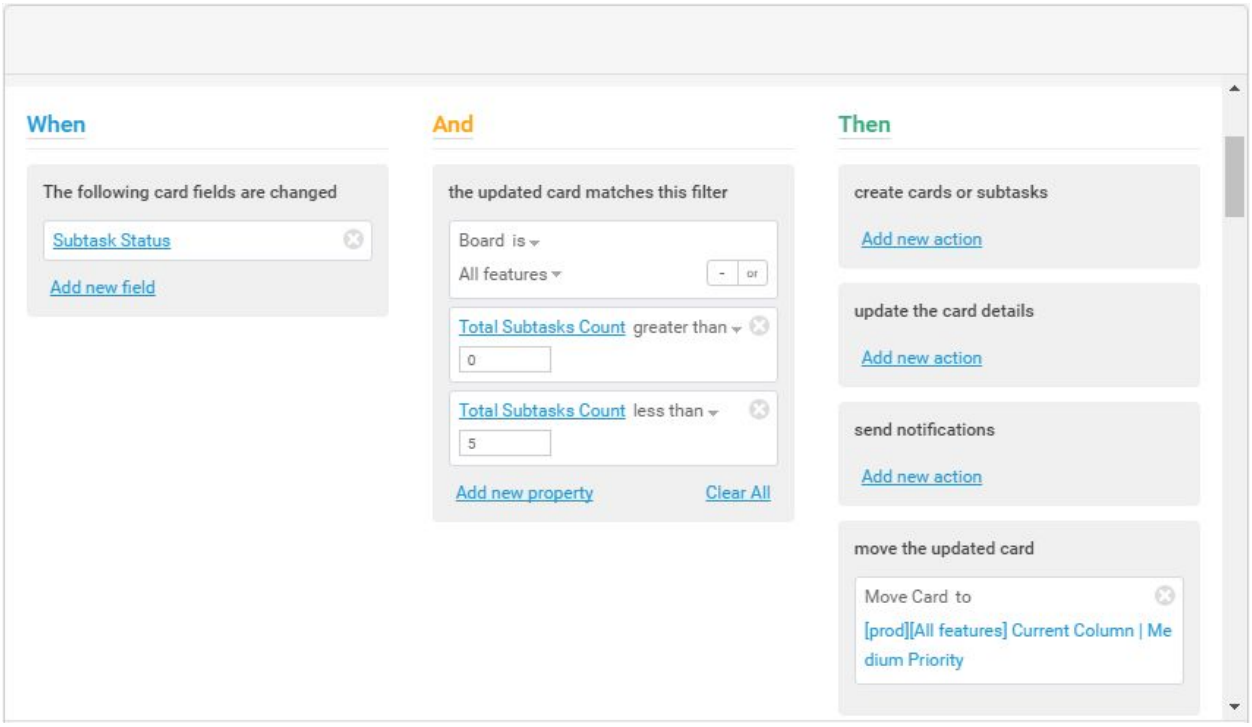
most popular. The realization is really simple - whenever a customer requests a new feature, a new card is created on this board. If the feature has been requested before, this means that such a card already exists on the board and, in such cases, we just create a subtask in that card. In other words, a card with no subtasks has been requested just once, but a card with four subtasks means that the feature has been requested five times in total. Here is how the first swimlane (high priority) of the board looks in reality:



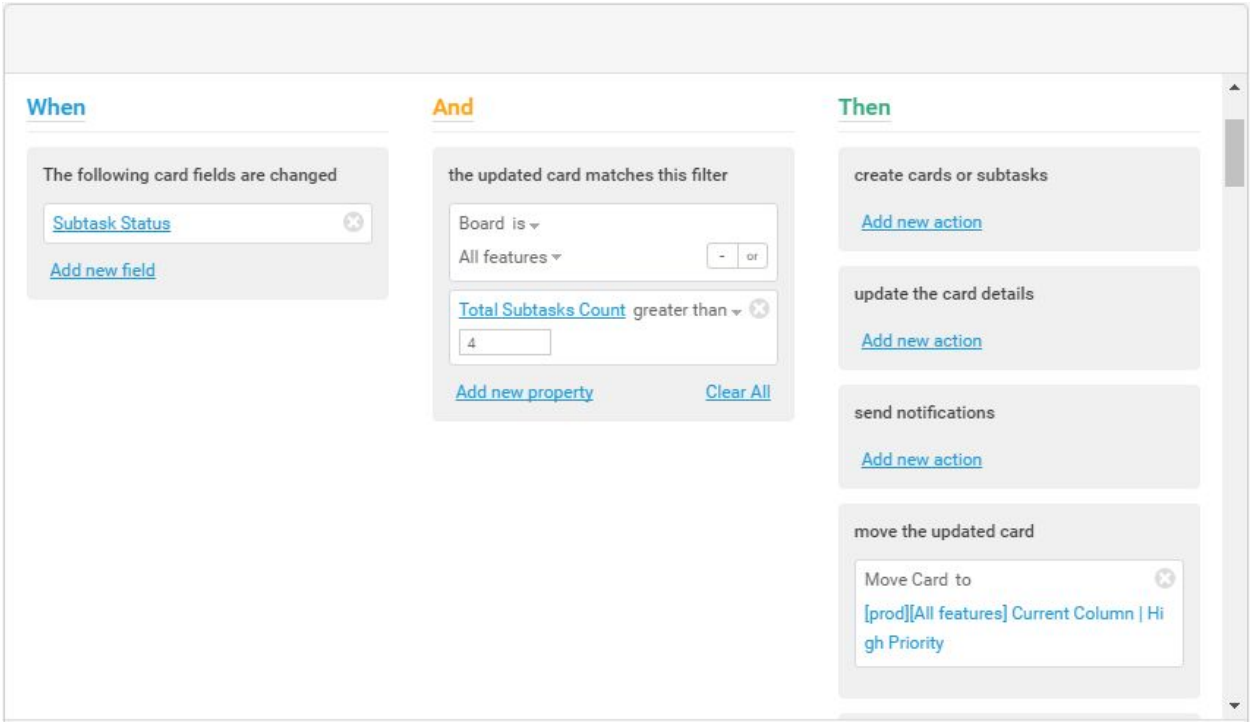
The Product Management Backlog in Kanbanize with Subtask Details

Visualization is, once again, very helpful here (I have obscured the subtasks text, because this is real communication coming from customers). We can very quickly see which of the requests

are most popular by just comparing the height of the different cards. The cards with more subtasks are larger and this directly means that more people have requested this enhancement. Now comes the “sexiest” part. We have a runtime policy which counts the number of subtasks on all cards and if a card has up to 5 subtasks, the policy moves the card automatically to the second swimlane. Another policy is configured to move the cards with 5+ subtasks to the first swimlane (high priority). This is how we automatically maintain a list of the most important features to work on. Here is how the policies look:



Subtasks automation policy in Kanbanize



Subtasks automation policy in Kanbanize

The only thing needed here is the discipline to consistently capture requests as subtasks and attach them to the correct card. It takes some time, but unless you use an automated voting system, there is probably nothing more effective than what is suggested here. We are not big fans of automated voting systems because feedback is only meaningful when you know who gave it. With automatic voting systems you rarely know this information and, at least at Kanbanize, we are hesitant to take decisions based on such feedback.

One potential issue with this approach is the fact that information may be duplicated if more than one person works on such a board. That is why you should always use keywords in the title of the card in order to make it easy to filter. What we always do is filter by a given keyword, check if such feedback already exists and, only then, decide whether a new card should be created or a subtask should be attached to one of the existing cards.

The Feature Management Board

In the previous section, we explored how ideas and feedback can be collected and prioritized in a visual backlog. In the current section, we will see how we actually run the service that feeds the development teams with work.

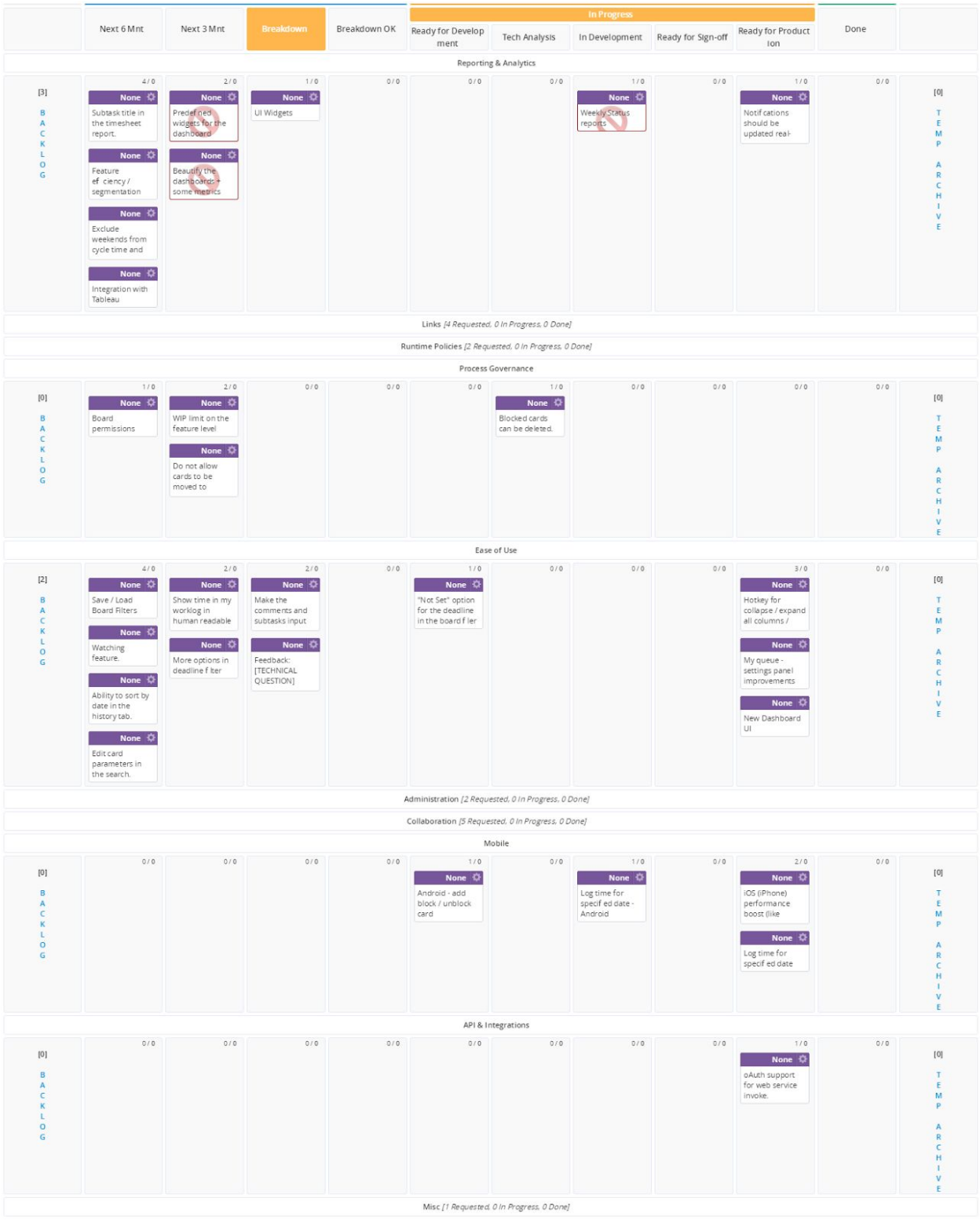
The board where we track the progress of features is called “Kanbanize-features Roadmap”. It is not a roadmap in the real sense of the word, but provides some generic overview about when features can be expected on production. **It is important to mention that this board is a sort**

of MASTER BOARD and it works in combination with other boards. In our particular case, the other board that is involved in the product development process is the board of the development team. Here is a scheme of the roadmap board structure:

Next 6 Mnt	Next 3 Mnt	Breakdown	Breakdown Done	In Progress					Done
				Ready for Dev	Tech Analysis	In Development	Ready for Sign-off	Ready for Production	
Reporting and Analytics									
Links and Dependencies									
Runtime Policies									
Ease of Use									
Administration									
Collaboration									
Mobile									
API & Integrations									
Miscellaneous									

The Kanbanize-features Roadmap Board Scheme in Kanbanize

And here is how the actual board looks:



The Kanbanize-features Roadmap Board in Kanbanize

It is obvious that the swimlanes represent the different categories the features can fall into, but it is more interesting to take a look at the columns and their purpose.

Thank You for previewing this eBook

You can read the full version of this eBook in different formats:

- HTML (Free /Available to everyone)
- PDF / TXT (Available to V.I.P. members. Free Standard members can access up to 5 PDF/TXT eBooks per month each month)
- Epub & Mobipocket (Exclusive to V.I.P. members)

To download this full book, simply select the format you desire below