CODEWISE CASE STUDY: SCRUM IN CUSTOMER SUPPORT

About Codewise

Codewise (https://codewise.com/) is dynamically growing its international presence in internet marketing. Over a short period of time, it has grown to way above a hundred of people and found its previous working methods ineffective. Trying to maintain its non-corporate culture, Codewise provided teams with Agile Coaching but decided not to enforce any approach.

Codewise is strongly IT-oriented company operating in the online marketing business. The company came up through quite a well-tread path. As a small start-up, they've been able to conquer the market by quickly delivering products that address their customers' needs, but over the time, as they have grown, agility has become a challenge. For a company with a couple of products and over a hundred employees, the delivery of new functionality or solving customer requests was taking weeks instead of hours. The organizational culture had also created a cult of technical perfection. The product was not seen as shippable until all details were implemented. Our proposal was to introduce an iterative approach to reducing the risk of over-engineering solutions. However some of software engineers had had bad experiences with pseudo-Scrum implementations from their previous jobs and opposed any changes to their current ways of working. Interestingly, it was the Customer Support group who volunteered to run the experiment.

The team was in difficult situation. There were two main problems they had been struggling with. First, the team was overloaded with requests, which was leading to burnout of employees and high rotation in the team. Several experienced team members had left the team. Second, with growing usage of the product, the number of customer request was increasing, and thus the service level agreements were difficult to meet. As a result, the team was trying to figure out an optimal group size which would be able to handle all the work.

Step 1: Identify current state

The first step was focused on visualization. We noticed that there are 2 types of work items:

- Customer requests – hundreds of tickets logged daily. The team could neither control their arrival time nor amount at the moment. There were established SLAs for these items and therefore work should be started immediately on arrival.
- Internal requests included those generated by the software engineers, the marketing team, the management or the team itself. There were no SLA for these items, and the team could decide independently when to start working on them.

The team decided to use a physical board for internal requests and keep customers tickets in an electronic tool. They chose a weekly Sprint duration. The first iteration gave us some interesting insights. First, the items on the physical board were not moving at all due to dependencies, missing knowledge or just simply lack of time. Second, we noticed that the electronic tool was neither allowing us to answer the question of whether we had met the SLA, nor giving us the possibility of analyzing gathered data. As a result, the team decided to take two actions:

- reduce the number of items on the physical board by saying "NO" to some of them,
- identify and introduce a tool that would help us to analyze trends and patterns in customers' requests.

Over the next weeks the team was pulling one or two internal request per Sprint, which allowed them to focus better and deliver those items faster. They also independently chose a new tool which meet their needs and implemented it. As it was easier to use, they were able to react faster to customer requests. Additionally, they were able to identify patterns and create a plan to address most common problems. This was a great sign that the team was taking responsibility for their own work, - a significant change from the previous way of working, where their tasks were assigned to them in the middle of the night.

Step 2: Change group of people into a team

The team at this point consisted of 2 experienced people and 6 newcomers. Most work was done by individual team members, which resulted in lack of knowledge sharing, new team members being blocked by lack of information and duplication of work as the same problem was solved by several people. We wanted to improve the collaboration between team members, so the team started running Daily Scrums. In the beginning, meetings were pretty long as many blockers and issues were discussed. Over time, as the habit of collaboration outside the Daily Scrum evolved, they reduced the length of the meeting to under 15 minutes. Additionally a form of pair-programming, called "shadowing" was established – meaning that the team solved difficult issues in pairs. This gave the team members more experience and sped up the work of the whole team.

Step 3: Stay focused

Due to the nature of Customer Support work, it is impossible to predict or control the number of raised tickets. In addition constant context switching was impacting effectiveness of team members. Focusing on a single task was difficult when work was interrupted by new requests. To address that, the team introduced "no ticket day". Each Sprint the team selected a couple of important items to be done. At Daily Scrum they decided who was going to work on this important improvement or task and that person was able to focus on this work, being protected from interruption by other team members who swarmed to solve upcoming customer requests.

Step 4: Automate repetitive tasks

Every Retrospective, the team analyzed data from the new tool, such as types of requests from customers, systems breakdowns, or new functionality delivered by developers. The team selected the most time consuming, most frequent or most hated type of requests and planned improvements to address them. Thanks to a "no ticket day" policy, some of the team's work has been automated by creating tutorials, video guidelines, a self-study knowledge base and a repository of canned responses. This gave the team space for new experiments and freed team members from some of the repetitive work. From being full-time on answering customers the team achieved a balance of around 60% of time spent on customer requests and 40% for improvements.

Result: Great Customer Support Team

A year since we began, the team has grown and split into more focused groups. All teams still use a weekly cadence, task board, and electronic tool, and the Daily Scrum is completed in less than 15 minutes. The practices of working in pairs and having a "no ticket day" policy have propagated to most of the other non-IT teams in the organization. Retrospectives and Planning have been combined into one meeting as our plans are mostly focused on improvements. The team decided that we are not only meeting our SLAs, but we are ready to extend our commitment to customers. New team has been established that is going to provide support 24/7.

The team has prepared an introduction program not just for new members, but also for other teams and customers. New employees are introduced significantly faster than before the transformation.

Retention has improved significantly. During the last year no one has left the team and when team members got the opportunity to change departments, almost everyone decided to stay in Customer Support. In other departments this group is perceived as effective, delivering a good job, having most impact on product development and fun to work in. Other people are asking if they can join.

Finally, both management and the IT department started noticing the value of Scrum practices and any negative perception was very limited. IT teams working on the same product as the Customer Support decided to join the Scrum side. They began running two-week sprints and all teams, together with Customer Support run joint Reviews. The Customer Support team is invited by IT teams for Planning and is involved in the daily work of the software development group. The company is still growing and introducing new products. There is pull from other employees to move next products into similar synergy.

Acknowledgements

We would like to thank all team members and managers that were working with us for being constantly attacked by IT vocabulary such as "refactoring", "automation (build and test)", "pair programming", and using their creativity to translate these software patterns to the non-IT world.

We especially would like to mention Tomasz Kaczanowski, Alek Fronczek and Maciej Szlachta from Codewise for inviting us to this experiment.

That is not the end of the story. We continue our cooperation so stay tuned.