5. 3. 2024 11 MIN READ

Startup scaling: From MVP to E-commerce

When helping one of our long-term clients scale up their product—a pivotal shift from online consultations to the strategic development of an e-commerce solution — we experienced both challenges and growth opportunities for our team dynamics. How did we adapt, restructure teams, implement Scrum methodologies, and employ a collaborative problem-solving approach to address difficulties that arose during the process? Let's take a closer look at more details.

Martin Bórik
Technical Analyst

A product that originated as a marketplace platform for interior designers to offer video consultations has found its market during the COVID-19 pandemic. It went through an MVP phase and not only secured an established position in the market but became a topic of widespread conversation. It found its customers and also with the help of promotion by top designers, the product made millions in revenues and thousands of users a day. It became an exclusive space for interior designers - with lower revenue but high margins, it was more or less a premium product.

In this part of the product, we were focused on improving the experience for clients with smaller iterations and features, optimizing the site for better conversion, and most importantly collecting data to analyze it further for possible marketing purposes.

Where to grow now - the E-commerce shift

It became evident that our current setup, primarily reliant on online consultations, had its limitations and couldn't sustain indefinite growth. The effectiveness of online discussions about interior design was lacking, particularly due to their non-repeatability. Clients typically seek interior design solutions as a one-time need, with minimal likelihood of returning for consultations. Furthermore, as the Covid pandemic ended, clients were once again able to engage with interior designers in person.

The strategic next step was therefore to sell other services or products to our clients using the strong acquisition potential of consultations and the audience of interior designers on our platform.

Through initial attempts with so-called *Trade-Only products*, which designers could recommend and sell to clients during consultations for a commission, we eventually arrived at the need to transform the product into a full-fledged e-commerce solution - Showroom.

Splitting into domain-based teams

During that period, our team comprised 7 developers and 2 members in the hybrid role of Project Manager, Product Owner, and partially Scrum Master. Our approach to delivering features involved using Release Candidates. This meant that features were gathered every week or two, deployed, and rigorously tested on the Staging environment before making their way to production. This led to splitting the team according to domains. *The first team* kept 4 developers and continued to maintain the platform with consultations for interior designers. Meanwhile, *the second "Showroom" team*, consisting of 3 developers, embarked on a new path towards integrating an e-commerce solution into our product.

The problem of a divided society

The "Showroom" team delivered an MVP e-commerce solution just a few weeks after the division. Members of this team expressed their satisfaction and enthusiasm during retrospectives, and they enjoyed their work while making

rapid progress. The team tasked with maintaining the consultation platform faced challenges from the outset. Frustrations arose from technical debt and a palpable "bad mood" stemming from the team separation and disrupted transparency.

Delivery caused further problems. While the "Showroom" team needed to modify and expand the e-commerce platform with features, complex payment system reworks, and the need for fast production delivery, the "Consultations" team worked on improving and lengthy refactoring of the platform itself. The clash in priorities resulted in merge conflicts and technical blockers, causing some Release Candidates to linger for weeks.

Team expansion and delivery setbacks

As e-commerce required fast delivery of complex features, the "Showroom" team had to be strengthened - in the following six months, it grew to 7 developers, plus a tech lead and one Project Manager combined with a Product Owner. A dedicated Product Designer was also added, followed by a Product Manager, who began to take control of the direction and quality of the product.

The "Consultations" team remained the same in size, with the addition of a specialized senior development engineer tasked with managing improvements to the technological platform of the entire solution.

However, the growing Showroom team faced challenges, with stakeholders' escalating demands leading to frequent overtime and tense releases, resulting in team burnout after each launch. To cope with the rising pressure, even the Consultations team began contributing to Showroom features. Unfortunately, this effort didn't boost overall velocity, as substantial time was spent on onboarding and explaining fundamental concepts, features, and code to the additional team members. The cumulative effect of these challenges reached a critical point, culminating in the failure of a release intended to launch an e-commerce beta.

Time for reconstruction and the approaches we adopted

What challenges did we face in the engineering operation, and how did we address the need for team organization and methodology changes?

- Recognizing the inefficiencies and challenges in the engineering operation, the team underwent radical changes. A stronger implementation of Scrum development methodologies became essential. This involved the wise division of the team, the establishment of efficient processes, and the appointment of individuals dedicated to guiding the team through these transformative changes. The goal was to address bottlenecks and previous challenges by fostering a more effective and organized approach to team dynamics and methodologies.

What role did the dedicated Scrum Master play in the team's functioning?

- The dedicated Scrum Master played a crucial role in facilitating team meetings, providing support to team members, addressing blockers, managing dependencies, and resolving conflicts. His responsibilities include fostering team collaboration and ownership, promoting transparency and cooperation, and empowering the team to address impediments autonomously. Additionally, he adeptly managed stakeholders' expectations, ensuring effective and motivational communication to prevent any potential frustrations among team members.

How did we address the need for cross-functional work and self-management within the team?

- To encourage cross-functional collaboration and self-management, a fundamental shift in team dynamics and responsibilities was initiated. This change aimed to empower team members to work collaboratively across functions and take ownership of their tasks and goals. The boundaries between domains or backend and frontend have collapsed. There was a mind-shift in the team and the minimum work required to fulfill the user story

was redefined to include covering all the necessary tasks through the codebase, from backend to frontend.

How did we handle the challenges related to the number of developers on the team?

- We faced a challenge associated with the increasing number of developers in a single team. Imagine that the team had 7 members, and each member generated 1 to 3 Pull-Requests daily, every member of this large team received a notification to start the feedback loop on the respective PR - with this number of people, it started to become counterproductive by the oversaturation of responses and feedbacks. Especially, when the final approval of the Pull-Requests was the responsibility of the technical lead who needed to know what was entering the codebase.
- A strategic decision was made to divide the team into development squads. Each squad, led by a technical lead, consisted of 2 to 3 software engineers, while we always tried to have one senior-ish member in each squad. This restructuring was aimed at increasing efficiency and preventing developers from being overloaded with communication and the number of assignments or administrative tasks, especially in the case of tech-lead roles.

How did we ensure a collaborative approach to problem-solving within the team?

- The team adopted a collaborative problem-solving approach where no individual focused exclusively on a specific domain or platform. Instead, everyone took on tasks from the backlog, with specialists in certain domains serving as advisors and collaborators. This approach fostered a sense of teamwork and collective responsibility. Also, within the processes, there was space and opportunities to foster specialists in certain new features and areas by building a culture of feature owners. They later helped the tech leads to specify and technically design the user stories.

How did the changes provide more time for tech leads to focus on technical specifications?

- Following the restructuring, tech leads were afforded significantly more time and space to dedicate themselves to improving the technical specifications of ideas. This allowed for a more thorough and detailed exploration of tasks and their technical requirements. User stories started to be technically assessed in their early stages when they were still an idea, and a technical exploration initiative was introduced using spikes (short and effective research). This prevented failures in the implementation process that had previously occurred.

Innovations we introduced to make large team meetings more efficient

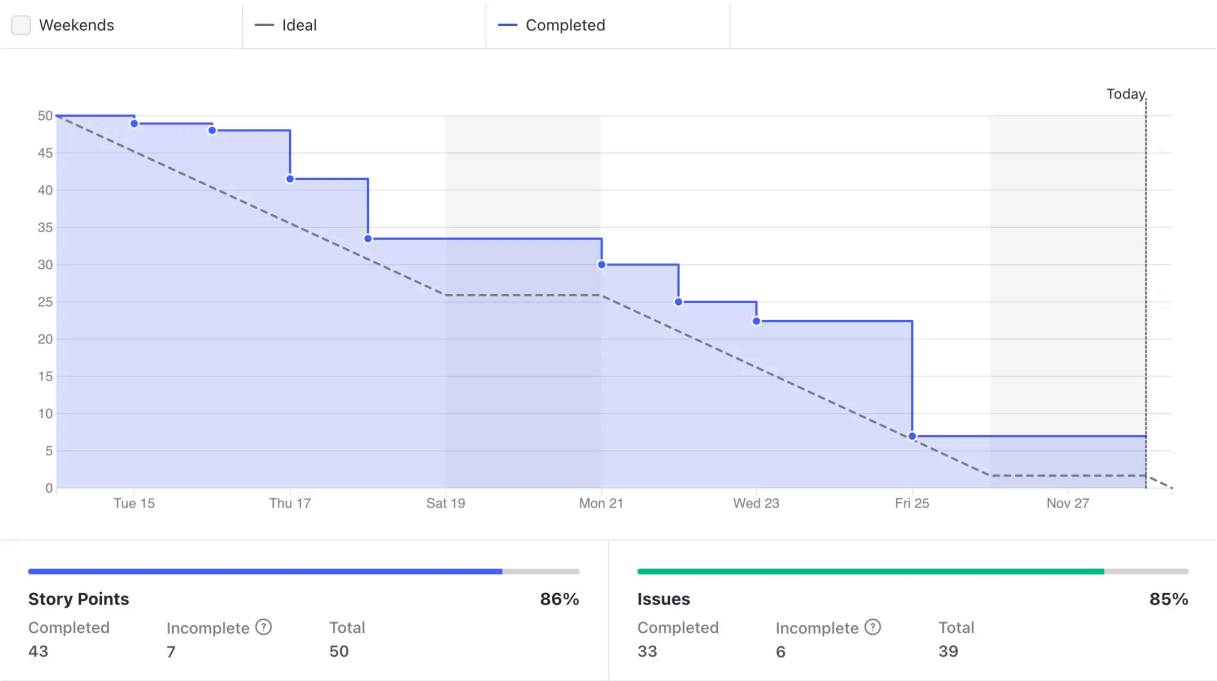
To change the way the team worked we established routine and iterative team functioning. An agile and iterative way of working in a bi-weekly sprint structure was introduced.

This included meetings such as Sprint Planning at the start of each sprint, where the team reflected on the previous sprint's successes and challenges, determined velocity, and planned tasks according to Sprint Goals. This was, of course, complemented by a retrospective to assess the team's performance and improve processes together.

Additionally, stand-up meetings for the large team were streamlined by incorporating an asynchronous text version. A Slack bot was introduced to ask team members about the status of their tasks, what they intended to continue working on, and any impediments they encountered. These async standups shortened the long morning debates about who worked on what the previous day and what they intend to work on today, by all members of all squads. Additionally, it gave tech leads, managers, and stakeholders a structured overview of how the work was progressing, as the async standup reports contained specific links to the tasks and PRs. At the same time, this innovation

allowed the Scrum-Master to quickly and efficiently point out obstacles and priorities in morning meetings that were already face-to-face. These used to last only a few (very efficient and addressable) minutes.

The Refinement session, occurring once a week, served as a crucial element in the team's workflow. During this session, tasks that had passed the technical specification were evaluated by story points, reaching a consensus within the team. This provided a structured and collaborative approach to refining and enhancing the quality of tasks in progress.



Sprint 4: At the beginning of October, we introduced a new division of the team into Squads and started to work agile in bi-weekly Sprints. Positive results didn't wait too long

How and on what evidence have we been able to further improve the team's performance

Agile coaching by the Scrum Master helped team members understand the significance of the fundamentals of agile methods and how crucial they are. This includes interpreting burndown charts and other graphs and analytical

tools provided by project tracking software. It allowed us, as a team, to appreciate the importance of continuously delivering tasks we committed to.

After the initial sprints, we began to see the results of what the team could deliver. This significantly aided in effectively planning each subsequent sprint and setting realistic expectations for both ourselves and stakeholders, which were previously ambiguous due to a lack of data. This also contributed to overall developer satisfaction, as they no longer felt overwhelmed by a waterfall of tasks and always had a realistic view of when work in the current sprint could be completed.

We introduced a calm-down day, every Friday on the last day of the sprint, where team members with completed tasks could focus on self-development, learning, and contributing ideas to improve the product. This helped them clear their heads and get some enthusiasm for the next sprint.

The team's velocity gradually improved as we started tracking time spent on individual development-related activities in more detail. We could identify how much time team members spent on programming, code reviews, testing, incident analysis, bug fixing, idea explorations, and validations, writing specifications, as well as the time consumed by meetings, collaboration, and cooperation (peer review, pair programming, etc.). Based on this data, we could precisely identify process improvements:

- For example, we reduced the time spent on code reviews by setting limits on the number of participants in code review loops. Tech leads also delegated some of their approval responsibilities of less complex tasks to senior members of squads, which expedited the delivery of tasks to production.
- We improved the software development life cycle by implementing the V-model. Every idea now goes from concept through the functional and technical design phases, enhancing our technical specifications, and making tasks more straightforward and granular for the solvers. While this slightly increased meeting lengths, it ultimately increased the speed and reliability of

delivered solutions, reducing ad-hoc meetings and additional communication between solvers.

- Additional testing of solutions was minimized by enhancing and implementing E2E tests, but primarily thanks to Trunk-Based development. This allowed a solver to see the finished result on a deployed environment similar to the staging environment from the git branch of each task.

We also utilized other agile methods to identify areas for improvement. During incidents and production problems, we conducted 5-Whys sessions as root-cause analysis, using the method of repetitive & iterative questions to get to the core of the problem. SDLC retrospectives also proved beneficial. These sessions involved evaluating the technical specification of an EPIC after it was closed, helping identify opportunities to improve technical design and better granularize partial tasks.

We identified tasks requiring a lot of collaboration and ad-hoc meetings and replaced them with initial Mob sessions. During these sessions, multiple implementers of an EPIC would come together for an intensive session of extreme programming, addressing all unknowns and issues simultaneously, which would otherwise need individual attention and synchronization.