Optimizing a support department through analytic insight and customer interaction

A case study

By Noah Lloyd-Edelman

Introduction

Transifex started as an open-source project on Github in 2007. During that time, requests and bugs were delivered straight to developers, who could accept or reject the changes on the spot. Questions could be asked as pull requests and there was complete visibility into the product's roadmap and its changes.

As the company grew and the customer base expanded, a support team was created to handle the company's growing pains. This team answered questions, resolved issues, served as sales engineers, and provided design and engineering assistance to all parts of the company. Although incredibly helpful, they were not providing analytic insights and actionable feedback to the company. I wanted to change this.

What is Transifex?

Transifex is a software platform designed to help internationalize digital content, that is, to present it in different languages and locales. It does this by creating a way to easily manage, collaborate, and translate digital content for organizations working around the world.

A developer can hook their product into Transifex, collaboratively translate, then export the finished files as quickly as they can translate them. There is no need to hire a translation agency or hunt down translators, third-party translation vendors are built into the platform. By incorporating a slew of translation utilities and supported file formats, most content can be directly imported and exported to the user's platform. Transifex can even collect and localize websites with a single JavaScript snippet.

Understanding the technical debt

Before I could consider helping customers, I needed to learn the system inside and out. This was the first and most time-consuming challenge. The Transifex staff were incredibly helpful and patient, but it was clear from the onset there was a lot of technical debt in the system.

Technical debt symptoms:

- The documentation was outdated
- There were more than five hundred unanswered support tickets
- The customer success team was not consistently providing actionable feedback to management and development
- There was no analytic insight into the needs of our customers or the company's performance in meeting those needs

In other words, up to this point, the company had been flying blind.

Creating a plan of attack

Responding to the mountain of tickets was our first priority. At the same time, we used document sharing and version control software to edit and significantly improve the troubled areas of the documentation. I installed analytics to start building a traffic baseline.

General guidelines:
1. Whittle away at the backlog of tickets
2. Make many small changes rather than a few large ones, the documentation should be *alive*
3. Keep the documentation up-to-date
4. Find, implement, and share any long-term solutions to reduce the ticket volume

Assumptions
- It was more time and cost effective to create a new strategy with our current system
- It was not worth the investment of switching to a new framework or hiring more people
- Tickets were the most time-effective way of interacting with customers

Hypothesis
Through analytic insight and communication with our customers on a daily basis, we could gain enough insight to make impactful changes to the software and company. We could even measure the success of changes by measuring ticket volume, issue types, and various other tracking metrics.

Immediate goals:
1. An empty support inbox by the end of the day
2. The creation and use of an issue priority system, in-company communication network, and issue tracking system
3. A weekly report with important issues that needed to be addressed within the company or software

The Transifex team

- Dmitry and Diego - the CEO and CTO, gave useful suggestions and feedback
- Marilena - worked on the complex tickets and was usually working deep inside the codebase
- Elena - editor and guide writer, helped wrangle the developers
- Me - editor, sales engineer, web developer

Using agile development for workflow optimization

When making digital products and services, it can be tempting to spend too much time trying to formulate a flawless strategy. We didn't have time for that. We needed a simple and actionable strategy which highlighted the specific problems we were going to solve for our users. The agile methodology was fantastic because it allowed us to quickly break problems into small actionable steps. We were able to consistently improve the product, while being flexible enough to handle large unforeseen issues which arose from the various parts of the company. There were a lot of twists and turns that made the flexible and fluid planning process absolutely essential. In fact, the lean strategic process happened in concert with design, allowing us to test our ideas in practice. We quickly figured out what was not working, made the necessary changes, and moved on.

In Athens, planning and development was incredibly easy. We would book a conference room, brainstorm some ideas, go back to work, and (ideally) have the changes implemented by the end of the day. If we had questions or found an alternative, discussing them was as easy as turning around and talking to the appropriate person. This was incredibly efficient and by the

end of the month the documentation had been completely rewritten and we had made significant progress with our ticket handling.

Things fell apart when I moved back to California. The nine-hour difference between our offices meant I could fit in a one-hour call in the mornings. If I had questions or suggestions afterwards, they had to wait until the next morning. To stay on track, we switched to a “second-person approval system.” This meant we worked independently, while always getting feedback from a second person before our changes went live. This cut down on our published errors, while allowing everyone to work at their own pace.

I maintained the documentation and marketing websites, Elena wrote the product guides, and Marilena handled the Transifex code. We all worked with support tickets. Elena took the general questions, I handled complex issues, and Marilena solved the serious bugs.

Product damage control in Athens

Our initial design process focused on “damage control.” This involved making small tweaks which were necessary to assure the stability of the platform. We then transitioned into an “improvement mode,” which was much larger in scope and focused on long-term improvements. The analytic baseline I created allowed us to measure the impact of our actions while we were testing solutions during the improvement phase.

During my month in Athens, we focused on designing the documentation structure, style guides, and ticket sorting tags. This was the beginning of our information wrangling session and we wanted to make certain what we were designing now would fit into the long term plans of the company. We met multiple times each day and reconfiguring our plans when a better idea came along (using whiteboards, documents, prototypes, and scratch paper).

After our systems were under control, we began working on quality-of-life improvements for the product. Since my team was split between offices, our progress and task tracking were handled online with JIRA and Trello. These platforms allowed us to manage, assign, and discuss issues remotely. Since we couldn’t book a conference room together, we would often design, implement, test an idea, and then share it with the team to see if it should be pursued further.

Creating personas for our users

Our “power users” mostly consisted of software developers, business managers, translators, and localization users. Because Transifex is an internationally-used piece of software, many users had very little technical knowledge and a loose grasp on English. This made it critically important for material to be as simple as possible. We extensively used images, code samples, and other information which could quickly be skimmed. We hoped our users were problem solvers, so we made certain to provide an answer with an explanation of the “hows” and “whys”.

Solved Problems

Reducing ticket response time and adding prioritization
Our first challenge was taking care of backlogged tickets. This consisted of reading each ticket, assigning it the appropriate tracking tags, replying if necessary, then tracking and closing the ticket when the fix had been made. This process took about five months. In the end, my solution

time went from four days to less than twelve hours and my reply times dropped from two hours to four minutes. This was done using reply macros, customer solution suggesters, and other useful tools we implemented

For VIPs, we added an auto-tagging system which prioritized “premium” and “enterprise” accounts, allowing us to quickly find important tickets. We divided the product into categories and assigned them to each support member. This way, we were able to quickly answer questions without having tickets bounce between members or stepping on each others toes. Ultimately, this allowed us more free time to work on other parts of the software and the real-time updates allowed us to meet the actual needs of our customers.

Establishing company communication for product changes
There were several instances in which important platform changes had been made by the Athens office without informing the office in California. Unfortunately, it was only when using the product did we learn about the changes. To remedy this situation, each change to the codebase was documented before being pushed to the live server. These notes were scraped and sent to the company as an email. If someone was gone for a holiday, they could simply read through the email chain.

Optimizing remote work
There’s nothing like working face-to-face, but this was not an option when Transifex had offices both in Greece and the United States. Instead, we switched to a second-person approval system and cloud-based project tracking to make certain we were making progress. In the end, this allowed us to monitor the health of the company and while delivering useful and actionable content.

Leveraging analytic customer insight
The documentation and ticket analytics allowed us to view the website traffic and where the troubled areas lie. After we corrected these areas, we could monitor our support traffic to see if it made a noticeable difference in metrics like ticket volume and type.

Leveraging support for ticket insight
By tracking our reply times, ticket volume by type, and satisfaction metrics, we were able to review our answering process and refine it, providing the best possible solutions in the optimal amount of time. My reply times went from two hours to four minutes, while customer satisfaction increased by more than twenty-five percent. This was achieved using reply macros, adding solution suggestions while submitting a ticket, and giving actionable information in every reply. We also created several channels for expedited communication with the development and management members so challenging problems could be addressed as quickly and effectively as possible.

Delivering actionable insight to the company
After reducing the backlogged mountain of tickets to a more manageable size, we began to prioritize and deliver five issues each week to the developers. They were often engineering tasks which ranged from complex bugs to simple error message fixes.