UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Computer Science Curriculum

Uku Pattak

Customer Support Gateway Built as Lean Software

Bachelor's Thesis (9 ECTS)

Supervisor **Jordan Valdma, MSc**
Supervisor **Helle Hein, PhD**

Tartu 2015

Customer support gateway built as lean software

Abstract:

Operating fast and iterating with tight feedback loops often differentiates startups from large corporations. Having the ability to adapt and learn rapidly from the market gives an competitive advantage. The final success of a product or a service often depends on user experience. Important aspect of user experience is Customer Support, which in companies often invest heavily.

In this work, we looked into ways to quickly deliver a piece of software with principles of Lean Software Development, Build-Measure-Learn cycle and Minimum Viable Product. We defined the problems with providing help to TransferWise customers and implemented a successful solution for the stated problems after analysing the alternative solutions from other services.

Keywords:

Lean software, Build-Measure-Learn, Customer Support, TransferWise.

Klienditoevärava ehitamine *Lean* tarkvara kujul

Lühikokkuvõte:

Tihtipeale eristab *startup*'e ehk idufirmasid suurfirmadest kiire tegutsemine ning sagedaste tagasisidetsüklite itereerimine. Kiire õppimis- ja kohanemisvõime annavad olulise konkurentsieelise. Toote või teenuse lõpliku edu määrab kasutajakogemus. Seetõttu pööravad firmad suurt tähelepanu oma klienditoele, mis moodustab suure osa kogemusest.

Käesolevas töös tutvustatakse viise, kuidas kiirendada tarkvara tarnimist. Selleks vaadatakse lähemalt *Lean* tarkvaraarenduse põhimõtteid, ehita-mõõda-õpi tsüklit ja minimaalset valmistoodet. Järgnevalt defineeritakse kliendiabi probleemid *TransferWise*'is, analüüsitakse alternatiivseid lahendusi teistelt teenustelt ning implementeeritakse edukas lahendus püstitatud probleemidele.

Võtmesõnad:

Lean tarkvara, ehita-mõõda-õpi, klienditugi, TransferWise

Contents

Introduction

Competition today is ruthless. Having the agility to quickly iterate with tight feedback loops and run faster is where the competitive edge lies. Technology companies today compete on how quickly they can take customer feedback and turn it into released features or improvements. We will look into ways to quickly deliver software by looking at the Lean Software Development principles and associated methods.

The final success of a product or a service often depends on user experience. Therefore companies put a lot of effort into their Customer Support (CS) which is a big part of the experience. In the end, it is all about delivering a brilliant product that users love to use.

The present thesis will look into ways to make user experience of a service called TransferWise better by concentrating on the CS side. The ideas are collected from other services that have excelled in providing help to their customers and then the journey to the solution is described.

Goals

The thesis carries the following goals:

1. Introduce the reader to the terms Lean Software Development (LSD), Minimum Viable Product (MVP), Build-Measure-Learn (BML) cycle and Test Driven Development (TDD).
2. Bring out problems related to providing help to TransferWise customers and investigate alternative solutions.
3. Provide an enhanced solution to the problems by gathering ideas from alternative solutions and building it with LSD principles in mind.

Outline

The work is organised as follows:

1. *Lean Software Development* - Introduces the term Lean Software Development, describing its origins and history, usefulness and giving an overview of its principles and methods.
2. *TransferWise* - Introduces TransferWise, its customer support and development processes.
3. *Problem Statement* - Introduces customer help problems in TransferWise and explains what metrics are affected by that.
4. *Current Alternatives* - Analyses current alternatives to find ideas for solving the problems.
5. *Our Journey to the Solution* - Introduces the solution to the previously defined problems.
6. *Future works* - List of possibilities for future work on our solution.

Chapter 1

Lean Software Development

1.1. History of Lean Thinking

In order to properly understand Lean Software Development an appreciation of its conceptual roots is appropriate. In 1950, Toyota was a small company that had moved from crafting looms to manufacturing cars. Toyota faced stiff competition from the established US manufacturers, and their small home market lowered the prices down. Owner Kiichiro Toyoda saw that there was incredible waste and unnecessary delays in car manufacturing. His vision to compete with the US behemoths would be to re-think car manufacturing from the ground up [1].

Toyota's manager Taiichi Ohno responded to Toyoda's vision by developing what came to be known as the *Toyota Production System* or in other words "a system for the absolute elimination of waste" [2]. Anything that does not create value for customer is waste and needs to be removed.

In 1990 the book '*The Machine That Changed the World*' gave a new name to The *Toyota Production System* [3]. From then on, Toyota's approach to manufacturing would be known as Lean Production. Those lean principles have also been extended to the supply chain, to product development, and to software development.

1.2. Definition

Lean Software Development (LSD) is a translation of lean manufacturing principles and practices to the software development domain. Principles are guiding ideas and insights about a discipline, while practices are what you actually do to carry out principles [1]. Lean software development has seven principles and many practices which have a very similar philosophy with Agile Software Development methods [4].

Agile Software Development is an umbrella term[1] for several software development methods (including Extreme Programming and Scrum) that were developed in 1990s [4]. These methods share a common philosophy which was described as values and principles in the Manifesto for Agile Software Development [5].

1.3. Lean Principles

The term "Lean Software Development" originates from the book by the same name, written by Mary and Tom Poppendieck [1]. The book presents the traditional lean principles in a modified form to be suitable for software development. Let us go through them briefly.

1.3.1. Eliminate Waste

Taiichi Ohno explained how *Toyota Production System* works "All we are doing is looking for the timeline from the moment a customer gives us an order to the point when we collect the cash. And we are reducing that timeline by removing the non value added wastes" [6].

Waste is anything that does not add value. The first step of eliminating waste is to recognize it. The second step is to develop a capability to really see waste. In software development it can be partially done work or extra features. Only about 20% of the features and functions in typical software are used regularly [6]. Other 80% still need unnecessary testing, documentation and support.

[1] A term used to cover a broad category of functions rather than a single specific item.

1.3.2. Amplify Learning

Software development is a knowledge-creating process. The best approach for improving software development environment is to amplify learning [1]. The learning process is sped up by usage of short iteration cycles. During those short sessions the development team learns more about the domain problem and figures out possible solutions for further developments.

1.3.3. Decide as Late as Possible

Irreversible decisions need to be delayed to the latest point. By the time the decision needs to be made there will be more information about which of those options is the best route to take. It also gives us time to potentially explore the different options in more detail and experiment, helping to come to the right conclusion [7].

In areas of complexity or uncertainty, where things are very likely to change, this is especially important. Example of deciding as late as possible in *agile development methods* is iteration planning. In agile, we decide what features to include in each iteration and analyse them just in time for them to be developed. Keeping decisions about features and the development of those features close together helps to ensure that the right product is delivered, because it leaves less room for change [7]. A key strategy for delaying commitments when developing a complex system is to build a capacity for changes into the system.

1.3.4. Deliver as Fast as Possible

It is common for people to think too deeply about future requirements that may or may not ever arise, or over-engineer solutions, both in terms of the software architecture, and also the business requirements [7]. This will slow down the pace of the development iteration.

In development the discovery cycle is critical for learning: design, implement, feedback, improve. Without speed, we cannot delay decisions. Without speed, we do not have reliable feedback. The shorter these cycles are, the more can be learned. Speed assures that customers get what they need now, not what they needed yesterday.

1.3.5. Empower the Team

There has been a traditional belief in most businesses about the decision-making in the organization – the managers tell the workers how to do their own job. In a Lean Software Development, the roles are turned around. The people who actually do the work combine the knowledge with the power of many minds. When equipped with necessary expertise and guided by a leader, they will make better technical and process decisions than anyone can make for them.

Top-notch execution lies in getting the details right, and no one understands the details better than the developers who actually do the work [1]. This means responding to people promptly, listening attentively, hearing their opinions and not dismissing them even when they are different to your own. Another important part of respecting people is giving people the responsibility to make decisions about their work. To achieve this, it is important to build knowledge and develop people who can think for themselves. People who can think for themselves and are experts in their area often need to be empowered to feel respected [7].

1.3.6. Build Quality In

Build integrity in means that the system's central concepts work together as a smooth, cohesive whole [7]. Software needs an additional level of integrity– it must maintain its usefulness over time. Software is usually expected to evolve gracefully as it adapts to the future. Software with integrity has a coherent architecture, scores high on usability and fitness for purpose, and is maintainable, adaptable, and extensible.

Quality is obviously extremely important, or you inevitably create all sorts of waste further down the line. It is important to avoid quality issues materialising as early as possible, and also to build it in throughout the entire development process, not just at the end.

1.3.7. See the whole

The larger the system, the more organizations are involved in its development and the more parts are developed by different teams. Quite often, the common good suffers if people attend first to their own specialized interests. When individuals or organizations are measured on their specialized contribution rather than overall performance, then it can ultimately result in an exponential increase in the time to add new features, notably lower quality product, which affects the end users and ultimately may also affect their efficiency or the competitiveness of the product [6]. A lean organisation seeks to optimise the whole value stream, not just individual functions or teams.

1.4. Minimum Viable Product

A common way to follow the "Eliminate waste" principle is to build a *minimum viable product* (MVP). "A Minimum Viable Product is that version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort." [8].

A MVP has just those core features that allow the product to be deployed, and no more. It is a strategy targeted at avoiding building products that customers do not want, that seeks to maximize the information learned about the customer.

1.5. Build-Measure-Learn

We use *Build-Measure-Learn* (BML) loop process to amplify learning and deliver as fast as possible. BML has three main phases illustrated on Figure 1.1. The first phase is the **Build phase**, in which the goal is to build a minimum viable product as quickly as possible. This is followed by the **Measure phase**, where the goal is to determine whether the real progress is being made; and finally, by the **Learn phase**, where a decision is made whether to persevere (carry on with the same goals) or pivot (change some aspect of the product strategy) [8].

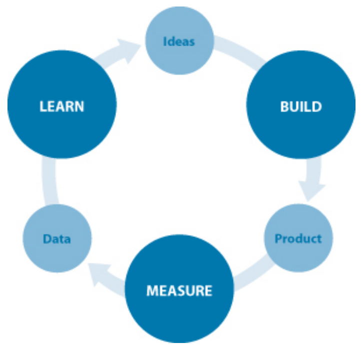


Figure 1.1: Build-Measure-Learn phases with Ideas-Data-Product outcomes [9]

The Build-Measure-Learn loop emphasizes speed as a critical ingredient to the product development. A team or company's effectiveness is determined by its ability to ideate, quickly build a MVP of that idea, measure its effectiveness in the market, and learn from that experiment.

1.6. Test Driven Development

Build integrity in is done with *Test Driven Development* (TDD) that relies on the repetition of a very short development cycle: first the developer writes an (initially failing) automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that test; and finally, refactors the new code to acceptable standards. It encourages simple designs and inspires confidence [10].

Chapter 2

TransferWise

2.1. Overview

TransferWise is a money transfer service with more than 300 currency routes globally. The difference between conventional money transfers and TransferWise lies in how payments are routed. Instead of transferring the sender's money directly to the recipient, TransferWise redirects to the recipient of an equivalent transfer going in the opposite direction. Likewise, the recipient of the transfer receives a payment not from the sender initiating the transfer, but from the sender of the equivalent transfer. This process avoids costly currency conversion and transfers crossing borders by allowing the company to minimise the amount of money exchanged between currencies, and the savings are passed on to customers [11]. This is illustrated on Figure 2.1.

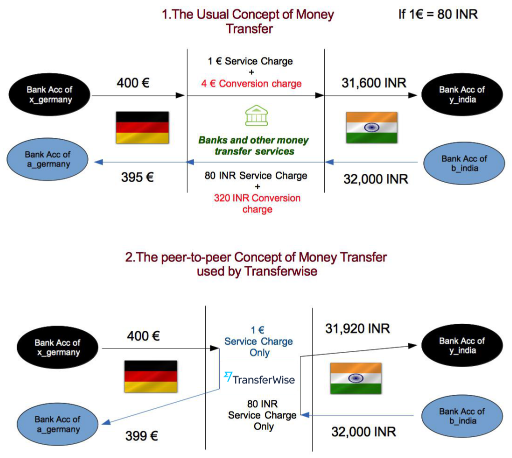


Figure 2.1: Comparison between TransferWise and other money transfer services

2.2. Customer Support

Customer Support (CS) is a division of TransferWise that attempts to help the user resolve and understand specific issues with the product or services provided. Because of the nature of TransferWise's activity in the financial services market, specifically involving users money, the volume of contacts to this department is high. As one could expect, without support a client would be much less inclined to take a risk with a service they do not trust and understand, particularly if it could result in a loss of their savings. TransferWise's emphasis in this area is to make sure no risk taking is needed, the service is clear and understandable, and above all, provides a service that customers can trust.

2.3. Development

TransferWise service development is driven by metrics. A*ctionable metrics* can lead to informed business decisions and subsequent action. These are in contrast to *vanity metrics* - measurements that give "the rosiest picture possible" but do not accurately reflect the key drivers of a business [12]. It is important to understand which metric is the actionable one and which one can only mislead the development.

Teams who develop TransferWise are autonomous and independent. Their work progress is tracked with a metrics called *key performance indicators* (KPI). Thesis author is working in a team called Transfer Creation to whom is most important to convert all the users who have intention to make transfer. This is measured with a conversion metric which is the team's main KPI. Conversion metric is affected by many different factors. One of those is the service quality - defects and bad user experience. We'll discuss this topic in more depth in Chapter 3.

Chapter 3

Problem statement

3.1. Bug Report System

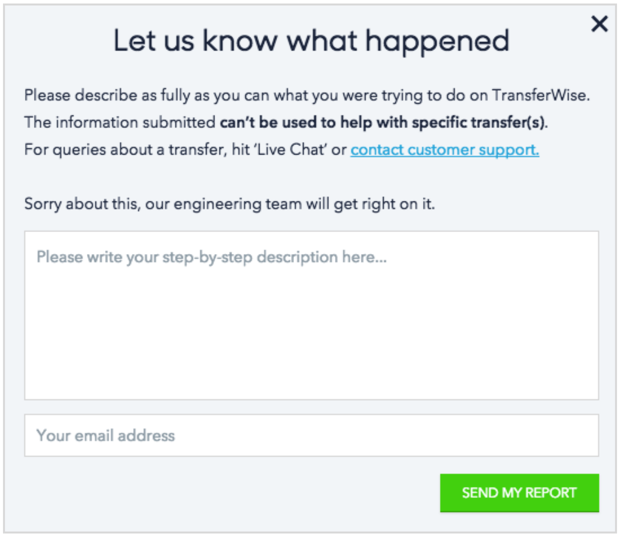
Usually the defect reports are coming from the users. They report the issue to the CS who then talks to the responsible team. The problem here is that there is not much information collected from the customer, because customer do not know what data we want from them and how it should be structured.

To get more information from users, CS needs to have a multiple roundtrips where they ask specific questions based on context of the issue. But still sometimes engineers can only use their imagination while fixing reported defects.

To solve this issue we came up with a system called *Bug Report*, which allows users to report defects with their own description that is merged with a background info:

- screenshot of the problem
- JavaScript browser log
- client technical info - browser name and version, Operating System name and version, screen size.

At first we wanted to validate the assumption that users will report bugs voluntarily. For that we created a MVP with a basic form (as seen on Figure 3.1) without any extra data from the background.


Let us know what happened

Please describe as fully as you can what you were trying to do on TransferWise.
The information submitted **can't be used to help with specific transfer(s)**.
For queries about a transfer, hit 'Live Chat' or contact customer support.

Sorry about this, our engineering team will get right on it.

Please write your step-by-step description here...

Your email address

Figure 3.1: Bug report system screenshot

The experiment was planned for 2 weeks and for 10% of users. Link to the form was placed in three of the most popular pages, in their header, footer and Live chat window. For technical reasons it only ran for 5 days. There were 12 776 participants, 38 clicks on links (that opened the form) and 2 submits. From those two submits there were 0 bug reports which could only mean that the experiment failed. By analysing the reason why the experiment failed we concluded that the links to the system were placed on the page where users were looking for help rather to report a bug.

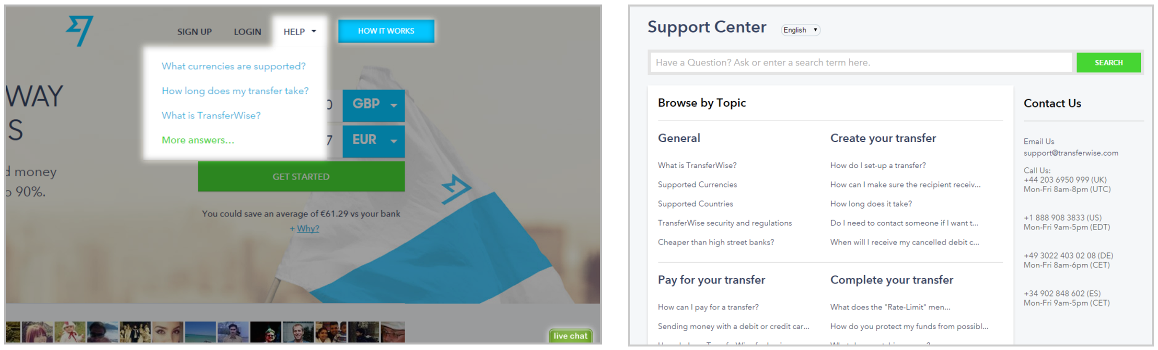
Learnings from the Bug Report system were promising. We started thinking about better places for the links and then we found a new problem: TransferWise misses a clear gateway to help, all the information about getting help is scattered around the service and users have to figure out their own way to the Help page.

3.2. Help Out of Sight

In TransferWise there are multiple ways to get help:

- Contact CS via:
  - Live chat, which is opened at 11 AM - 11 PM,
  - call via phone which has multiple different country lines opened on different times,
  - send an email.
- Frequently Asked Questions (FAQ) list.

Live chat option can be used on every page by clicking on the "Live chat" button on the bottom right corner. This makes it the most prominent way to contact CS. Email and call information can be found partly in the header, footer and on the Help page. FAQ has its own section in there also with a few hard coded questions in the header. All of these are focused on Figure 3.2.



Support Center English

Have a Question? Ask or enter a search term here. SEARCH

Browse by Topic

General
What is TransferWise?
Supported Currencies
Supported Countries
TransferWise security and regulations
Cheaper than high street banks?

Create your transfer
How do I set-up a transfer?
How can I make sure the recipient receiv...
How long does it take?
Do I need to contact someone if I want t...
When will I receive my cancelled debit c...

Pay for your transfer
How can I pay for a transfer?
Sending money with a debit or credit car...

Complete your transfer
What does the "Rate-Limit" men...
How do you protect my funds from possibl...

Contact Us

Email Us
support@transferwise.com

Call Us:
+44 203 6950 999 (UK)
Mon-Fri 8am-8pm (UTC)

+1 888 908 3833 (US)
Mon-Fri 9am-5pm (EDT)

+49 3022 403 02 08 (DE)
Mon-Fri 8am-6pm (CET)

+34 902 848 602 (ES)
Mon-Fri 9am-5pm (CET)

Figure 3.2: Ways to get Help in TransferWise

As we can see the information is not consistently placed and the service favors contacting CS rather than directing users to read FAQ before. This will lead to two problems.

Firstly, users who are not Sherlock Holmes, may not find a help, because all the links to it can be hidden or hard to find. Secondly, if they can find a way to contact CS then the response time could be over 8 hours which means they may discontinue using the service. The long response time is partly caused by a big amount of users who ask trivial questions what are answered on the FAQ page. These problems can affect higher level metrics like conversion.

3.3. General Problem

Conversion metric is the most widespread metric in TransferWise that indicates success. It shows how many users, who had an intention to make a transfer, were able to make a successful transfer with TransferWise. The success in TransferWise is not 100% and this is affected by many factors. Our hypothesis is that **part of the unsuccessful transfers are caused by users who have hard time to get help.** To prove this assumption we can only iterate with different solutions and see if we can increase the conversion.

In the following chapter we will look into solutions to our problems from other services. From there we try to come up with our own solution in Chapter 5.

Chapter 4

Current alternatives

In this chapter we investigate two service providers who have solutions for our problems stated in Chapter 3. We analyse positive and negative sides of those implementations to gather ideas for our own solution.

4.1. Zapier

4.1.1. Overview

Zapier is a web service that lets you easily connect web applications, making it fast to automate tedious tasks [13]. For example it is possible to send out an Email with Mandrill application when Firebase database has a new entry. Zapier depends on multiple application programming interfaces to connect different applications and this can make the servic fragile.

4.1.2. Reporting an Issue in Zapier

Zapier can detect problems with different apps faster with their easy to access issue reporting. On every page there is the red button "Get Help" which allows Zapier clients to find help with ease (as seen on Figure 4.1).

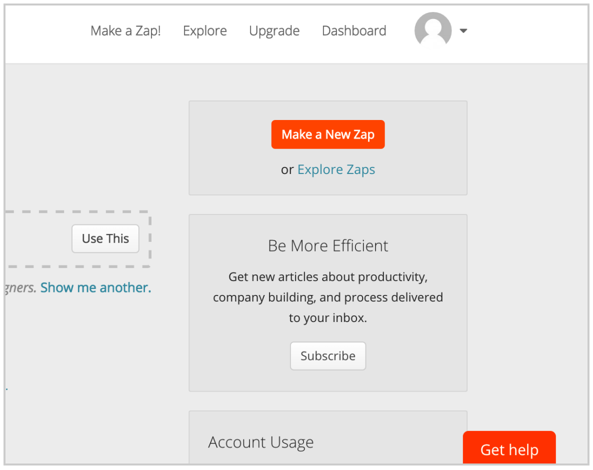
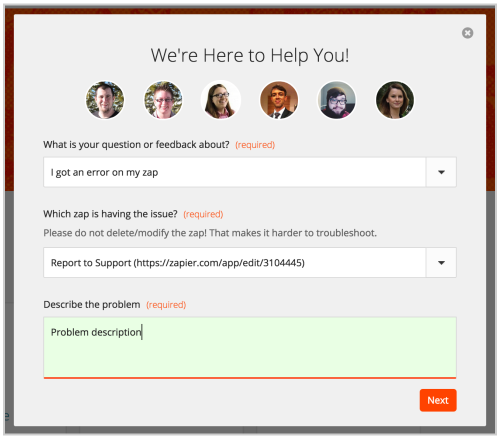


Figure 4.1: Zapier way to give help

The button takes a small amount of the screen and attention, trying to be found only when user will need help. Color of it is a bad choice for TransferWise: red means financial trouble, or it could mean danger [14].

Clicking on the button directs users to the Help page where they can choose a specific topic or an application with they have a problem. There will be a preciser FAQ list when choosing one. With filtering they reduce a thinking needed from the users and increase the user experience [15]. But with directing clients away from their current context to the Help page may confuse them and decrease the user experience.

On every Help page the “Get help” changes to “Contact us” button and clicking on it opens up a modal window (as seen on Figure 4.2). It has a contact form which allows users to report issues from the service rather than using an email. This is more convenient for clients and it gives extra opportunities for owners to get more structured data out of the reports via different inputs.


We're Here to Help You!

What is your question or feedback about? (required)

I got an error on my zap

Which zap is having the issue? (required)

Please do not delete/modify the zap! That makes it harder to troubleshoot.

Report to Support (https://zapier.com/app/edit/3104445)

Describe the problem (required)

Problem description

Figure 4.2: Support form with different flows

The form window has multiple trust elements to give people feeling that their reports are read and taken care of:

- customer support profile pictures
- "We're Here to Help You!" title.

This way they can build trust and increase reporting conversion [16].

4.2. UserVoice

4.2.1. Overview

UserVoice is a software-as-a-service provider of customer support tools [17]. The tools list is long and it is made to fully meet CS needs:

- widget for chat and feedback
- support ticket system
- forums
- etc.

Because TransferWise already has a tool for their CS then there is no point of using UserVoice. We can only analyse what the widget offers.

4.2.2. Using the UserVoice widget

UserVoice widget is used by many service providers including Memrise and Bitly. It is highly customizable, but we will be focusing on the default variant: a blue button with a Question sign on the bottom right corner (as seen on Figure 4.3). It has a same positioning method that Zapier has, but with a more calming color. Using only icon can seem like a cleaner design, but it can confuse the users [18].

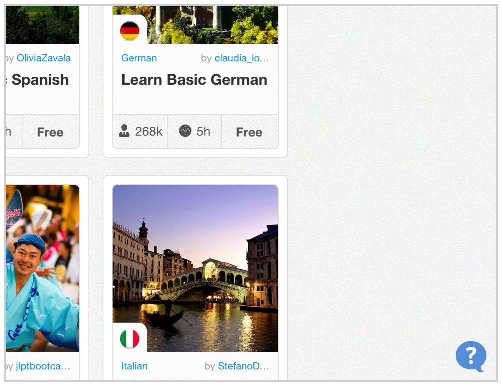


Figure 4.3: UserVoice widgets button

Clicking on the button pops out a much smaller modal window with no overlay compare to Zapier (as seen on Figure 4.4). Window takes small amount of a page room and it allows users to see the page they are. This makes writing a comment or a question about the page more convenient and no context changing is needed. The problem is information quantity that could be shown in this small window.

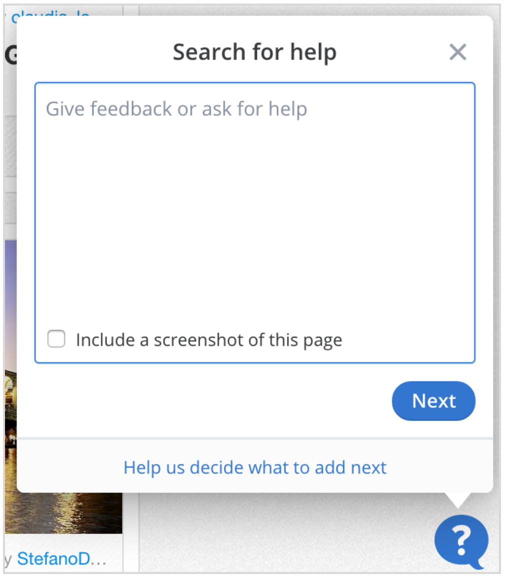

Search for help
Give feedback or ask for help
Include a screenshot of this page
Next
Help us decide what to add next
StefanoD...

Figure 4.4: Widget first view - a form

Customers can read helpful articles on the second step before submitting the report. Such flow has two sides:

- positive side - user will construct their question before and probably will be smarter to look answers from FAQ afterwards;
- negative side - user, who has put a lot of effort and time constructing the question, can be steamed-up after seeing a answer right away on the next step.

4.3. Conclusions

Both services have similar concepts in button placement, modal window and the flow. Link to a help is always visible to the users using fixed bottom right position. Although, Zapier has a clearer way to represent it with text rather than using icon, but with a bad color for a financial service.

UserVoice provides help information with modal window to keep users in their context and shows FAQ on an intermediate step before letting users to submit report. Zapier puts emphasis on FAQ and for that they have a separate page before user can contact support.

Each flow has their good and bad sides. In the following chapter we will pick some of the ideas from those services and use them to solve issues stated in Chapter 3.

Chapter 5

Our Journey to the Solution

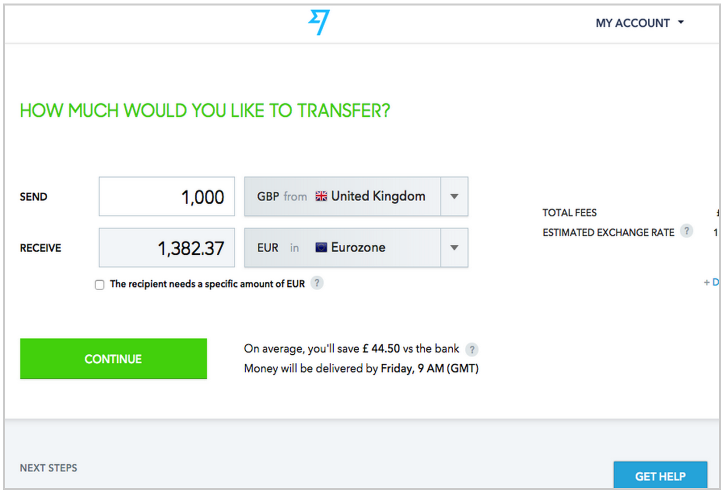
Our solution is built as a *MVP* with *BML* cycle in a form of a new module that integrates with the TransferWise service. In combination with some of the existing solutions we now have ideas how to solve the problems defined in Chapter 3.

5.1. Gateway to Help 1.0

5.1.1. Building

We started writing down ideas from existing solutions and mixed them with our own thoughts. After having a long list of things generated, it was time to eliminate the waste and do some sketches with minimal amount of features which will do the work.

First problem to solve was how get help from anywhere. For that we followed existing solutions by creating a fixed button "Get Help" as can be seen also on Figure 5.1. This would be positioned on every page and will open the Gateway.



HOW MUCH WOULD YOU LIKE TO TRANSFER?

SEND 1,000 GBP from United Kingdom

RECEIVE 1,382.37 EUR in Eurozone

The recipient needs a specific amount of EUR

TOTAL FEES

ESTIMATED EXCHANGE RATE

On average, you'll save £ 44.50 vs the bank

Money will be delivered by Friday, 9 AM (GMT)

Figure 5.1: Button which rules them all

Unlike from the UserVoice there is a modal window with overlay to have more room for extra information, but in the same time not directing users from their current locations like Zapier does. Window has multiple views that are isolated steps:

1. FAQ
2. Contact options list
3. Call
4. Email

With separate steps it is possible to understand on which step users choose to get help.

The FAQ view pops out first as seen on Figure 5.2. This way we can provoke users to think by themselves at first. If they can not find any answers from there then we shall give them opportunity to switch to the next view with a click on the “Contact Support” button.

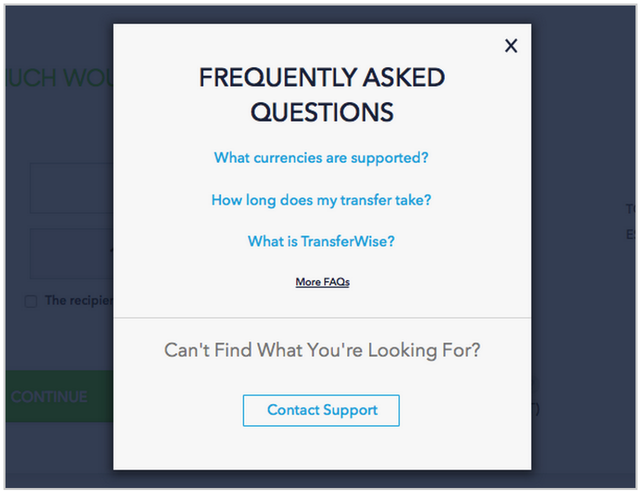

FREQUENTLY ASKED QUESTIONS
What currencies are supported?
How long does my transfer take?
What is TransferWise?
More FAQs
Can't Find What You're Looking For?
Contact Support

Figure 5.2: First step of Gateway

What made our solution different from others is the contact options list. There are multiple choices where customers can choose:

- Chat
- Call
- Email

We do not want to force users to choose one option over the other. Thats why we have options equally in a row with no hints that one is faster than the other etc. (as seen on Figure 5.3).

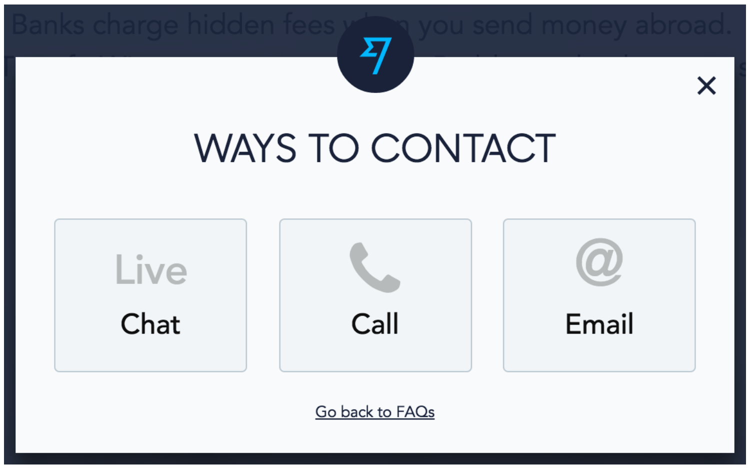
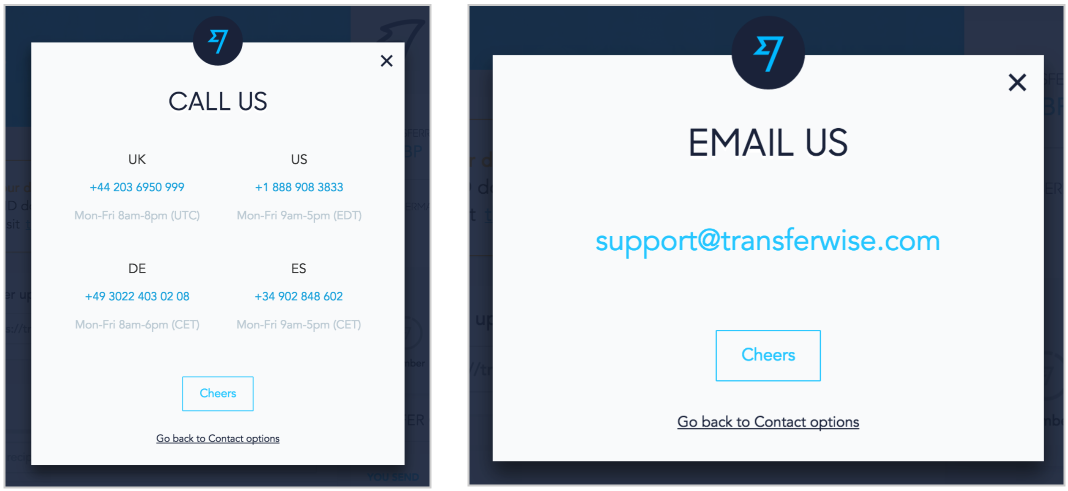


Figure 5.3: Contact options list

Live chat option can be only seen when the 3rd Party service *SnapABug* is turned on. Clicking on the Live chat option will close the Gateway and open the *SnapABug* service. This means the client has reached to the end of the flow.

Call and email options have their own views with static contact information shown on Figure 5.4. Both of them have a unique button with text "Cheers" which indicates successful finish for us. There is always the link to the previous step and the window will close when clicked either on the X or on the overlay.



CALL US

UK	US
+44 203 6950 999	+1 888 908 3833
Mon-Fri 8am-8pm (UTC)	Mon-Fri 9am-5pm (EDT)
DE	ES
+49 3022 403 02 08	+34 902 848 602
Mon-Fri 8am-6pm (CET)	Mon-Fri 9am-5pm (CET)

Cheers

Go back to Contact options

EMAIL US

support@transferwise.com

Cheers

Go back to Contact options

Figure 5.4: Call and email view sketches with real information

We created a AngularJS plugin, isolated module written in JavaScript on AngularJS platform [19] to implement our sketches as fast as possible. AngularJS is widely used in TransferWise. Building it as a component made it very easy to integrate it into different pages. Module was written using *TDD* process to follow "Built integrity in" principle.

Module has been integrated with analytics platform Mixpanel [20]: every click on the links and buttons, every step change and window visibility is listened and sent to the Mixpanel. From there we collect data from different graphs (Figure 5.5) for our metrics.

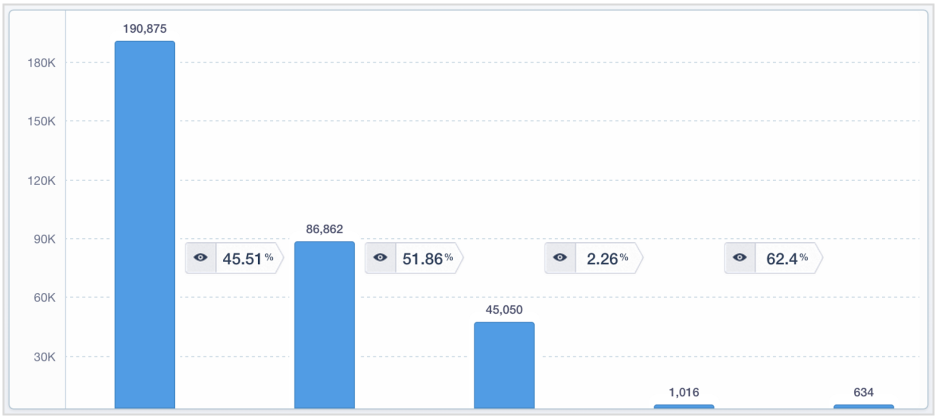


Figure 5.5: Mixpanel funnel graph

Server-side Groovy code was only needed to initialise the experiment. The feature was experimented on a 50% of TransferWise users and could be turned off immediately if anything should have gone wrong. To deliver as fast as possible we postponed internalisation by having everything in English on the user interface.

5.1.2. Metrics

We ran the experiment for 7 days by showing the "Get Help" link to a 50% of users. Collected statistics can be seen in Tables 5.1, 5.2 and 5.3.

Table 5.1: General statistics

Participants (who saw the button)	33 289
Clicked on the "Get Help" button	1291 (4% of participants)
Users (clicked on FAQ link or viewed contact info)	821 (64% of clicks)
Conversion	59.3%

Table 5.2: FAQ statistics

Clicked on FAQ link	307 (37% of users)
Read only FAQ	263 (32% of users)

Table 5.3: Contact statistics

Potential CS contacts	452 (55% of users)
Chose call	103 (21% of contacts)
Chose email	96 (23% of contacts)
Chose chat	253 (56% of contacts)
CS contacts	13 171 (4% increase from previous 14 days)

All of these metrics can be useful from different viewpoints. For example users preferences in contact options (Figure 5.6) will tell CS on which option they should be focusing more, but it can be irrelevant for the FAQ article authors.

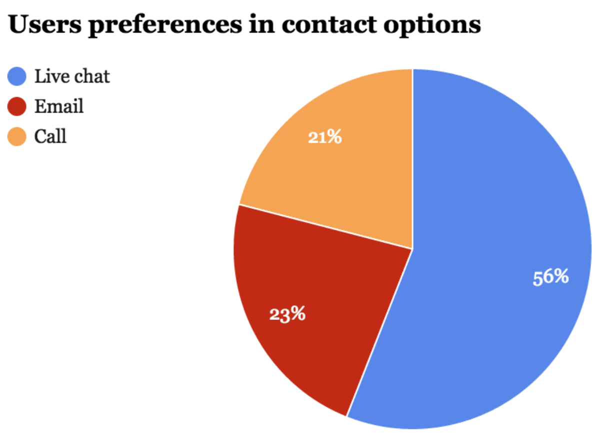


Figure 5.6: Users preferences in contact options

We had to choose *actionable metrics* in our context. The right ones are those that give us insights in our progress to solve the stated problems:

- Where to get help?

- How to provide help faster?
- How to increase conversion among users who need help?

A good way to see if the Gateway gives help, is to look at the percentage of users. Gateway users are people who tried to find help with clicking FAQ links or looked at the contact information. By analysing the percentage of those, we can see how many people did or did not understand the tool.

To make Help providing faster, one can decrease the amount of contacts CS needs to assist. Progress of that can be analysed by looking at CS cases amount. However this metric is influenced by many factors and teams, so it would not give us a clear feedback of our work. To see the real effect, we need to look at how many contacts we saved (people who got help without contacting CS) with the Help Gateway.

For our general problem, i.e. conversion, we are going to track users who needed help (clicked on the "Get Help" button) and made a successful transfer. To clear the noise made by other teams, we chose to look at only new users from the U.K which has the highest traffic, but the smallest amount of teams affecting the channel.

These metrics were selected to be our KPI and we were looking at them in a every iteration to measure our success when trying to solve problems stated in Chapter 3. Unfortunately, we also discovered a problem while collecting metrics. We had no statistics about current solutions (E.g. Help link in the header and the Live chat button) and without those we could not compare the new tool with the current TransferWise solutions. We fixed that problem in the next iteration.

5.1.3. Learnings

Firstly, as mentioned above we had no statistics about the current solutions which were needed for comparison. This was our first learning: think about the events which are needed to be tracked for statistics while building the tool. Also collect some pre-metrics in the first days of the experiment for confirmation.

Secondly, we gave help to 32% of Gateway users without them contacting CS by showing FAQ view first. Because the questions in the view were always the same for the new and the existing users then we expected this number to be around 15%. By having a higher outcome of 17%, we can say that putting the FAQ view before the contact options was a success.

Thirdly, only 64% of the people who clicked on the "Get Help" button really used the tool. Expected results were 80-85%, where most of the non-users would be the “testers” who had no intention to get help. Because our tool is only translated into English and TransferWise customers are from all over the world then it could have affected the metric. It is easy to solve this problem: internationalize the tool to all the supported languages.

5.2. Internationalised Gateway

5.2.1. Building

We started off by adding missing click trackings to the Live Chat button and to the Help link in the header. The next step was to translate the user interface into Italian, German, Spanish and French. For implementation we used the *angular-translate* [21] module.

Some of the translated FAQ links direct users to untranslated articles, which means a link and an article can be in different languages. In this case we followed the "Decide as late as possible” principle: we will choose to translate an FAQ article only then when we see users' interest.

With these changes we expected to increase the number of Gateway users by 5-10%. Also this (experiment) can develop into another side project where TransferWise teams can get insights about their new features, fixes etc. We are going to talk about it in Chapter 6.

5.2.2. Metrics

All the statistics in Mixpanel were checked immediately after the release to make sure everything we need for the experiment analytics is available. After another 7 days we started comparing this iteration (v2) with the previous one (v1). We collected statistics only for our KPI:

- **Gateway users** - users, who clicked on FAQ links or continued to Contact Support step.
- **Read only FAQ** - users, who clicked on the FAQ link and did not continue to Contact Support step.
- **Conversion** - users, who clicked on the "Get help" button and finished the transfer successfully.

Gateway users metric increased by 2% (as seen on Figure 5.7), from 64% (821 users) to 66% (903 users). At the same time "Read only FAQ" metric stayed on the same level - 32% (as seen on Figure 5.8).

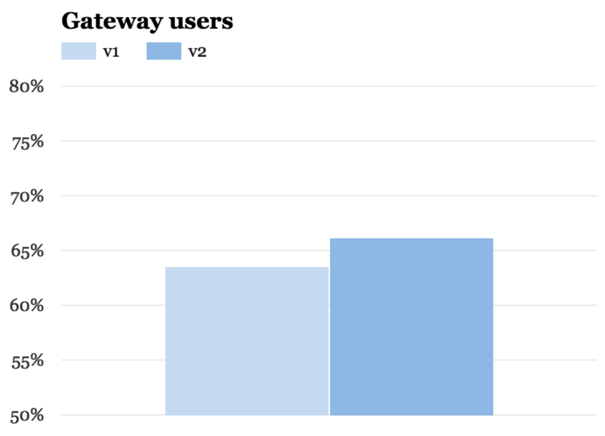


Figure 5.7

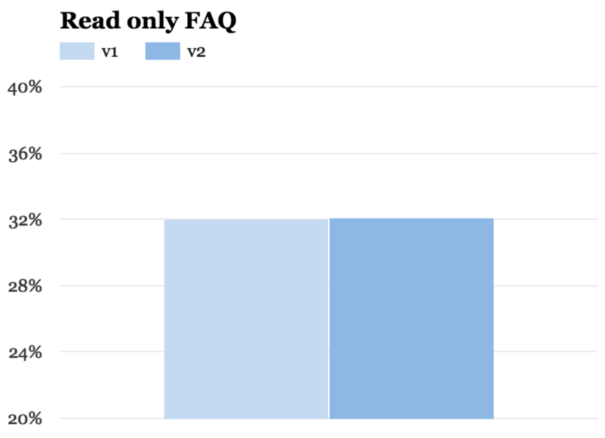


Figure 5.8

Conversion between first and second version escalated by 2%, from 59% to 61% (as seen on Figure 5.9) and the biggest difference can be seen between our solution and current solutions - Live chat button (44%) and Help link in the header (38%). These metrics are visualised in Figure 5.10.

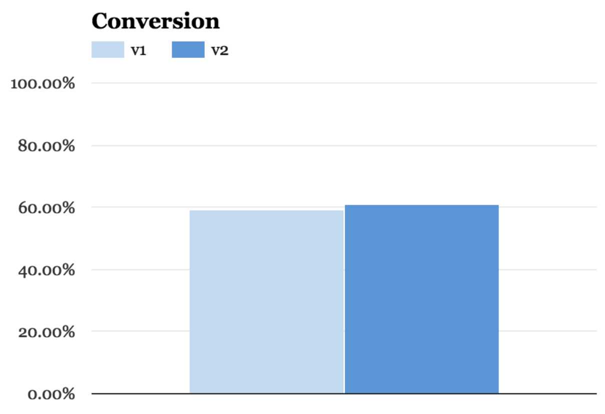


Figure 5.9

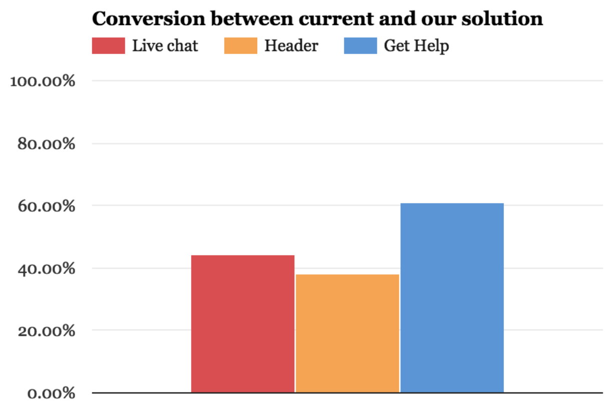


Figure 5.10

5.2.3. Learnings

Firstly, we were expecting to see a surge in the Gateway user's percentage which surprisingly did not happen. By having only 100 more users (2% more of the users who clicked) we can say that translated tool was not getting more attention from the users than the English only version.

Secondly, we were able to increase the conversion, but not with a help of a FAQ (because it stayed on the same level). One can assume that the translations made the flow more understandable - making it more clear what contact options we have - or the CS made a better job.

Finally, we have the statistics which we can use to prove our hypothesis. By having a 20% higher conversion rate from other solutions we can say that there were users who had hard time to get help before our solution. But we are not sure if this conversion rate is the solutions peak. If we could make the FAQ view more useful to the users then we might decrease the CS contacts amount and from that increase the conversion.

5.3. Location based FAQ

5.3.1. Building

In TransferWise FAQ articles are written by a small group of CS people, who are familiar with customers' questions. For example, they know that most of the time new users ask about TransferWise security and regulations on the ‘Landing’ page and "How long does my transfer take?" on the ‘Account’ page. To follow the "Empower the team" principle we teamed up with them to make the FAQ links more helpful.

We came up with a location based FAQ system which means that we are going to show different questions on different locations of the website. Some examples:

- When the user is on the ‘Landing’ page then we will show questions about changing personal settings.
- When the user is on the ‘Make transfer’ step then we will show questions about payment options.

FAQ list was constructed together with the FAQ team where we found 2-4 articles for every page in TransferWise. We used *angular-route* module [22] to detect location changes in AngularJS applications and trigger updates on the questions list in the Gateway. To deliver as fast as possible we hardcoded those questions into AngularJS service and covered the whole logic with automated tests. One last thing was to add Mixpanel event tracking to every question click and we were ready to release the 3rd iteration of the Help Gateway.

We were not expecting to see any dramatic changes in our KPI. From the first iteration we already have shown “More FAQs” link to the users which directs them to the FAQ page. With this iteration we had only made the precise FAQ easier to find and by that more convenient to use.

5.3.2. Metrics

Third iteration experiment was tested in the end of the month, which means most of the users are transferring received salaries back to their homeland. That also means higher traffic in TransferWise. Given fact was reflected from the statistics where we saw 10% increase in the participants. Other from that we also had a 2% more Gateway users (Figure 5.11) and a 4% more Only FAQ users (Figure 5.12). Conversion has gained 3% from the last iteration (62% to 65%) as can be seen on Figure 5.13.

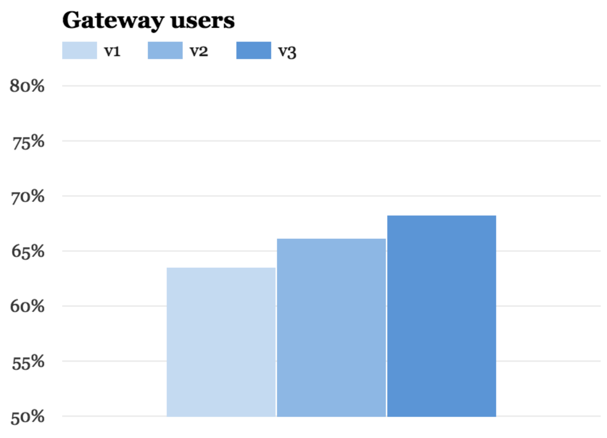


Figure 5.11

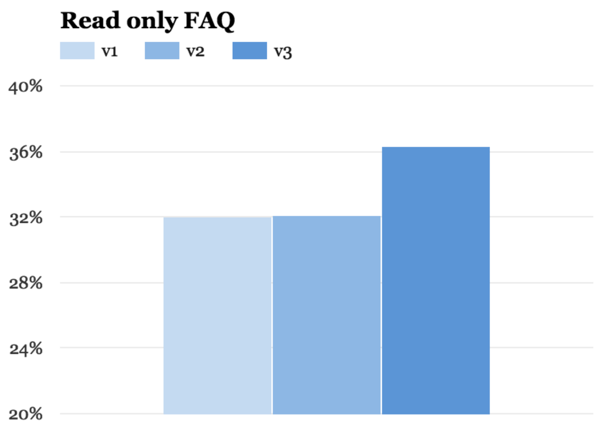


Figure 5.12

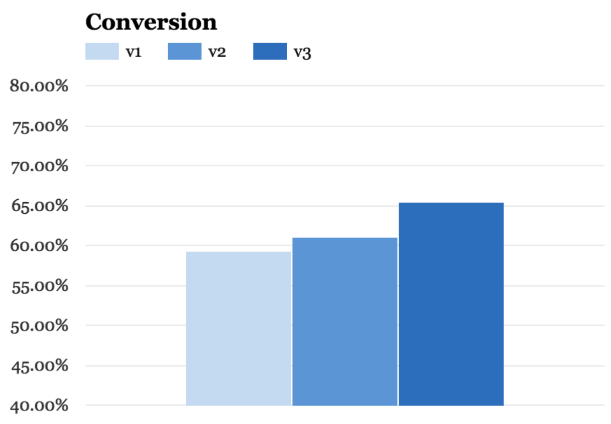


Figure 5.13

5.3.3. Learnings

We were able to move the conversion even further with a help of a FAQ view. In our tracked events we saw an enormous rise in users who interacted with the FAQ links in comparison to the previous iterations. Even though "Read Only FAQ" metric do not reflect the jump, we still got good insights about users problems on different locations.

Unfortunately we also detected some problems. While trying to deliver a new feature as fast as possible we forgot about the issues that hardcoded FAQ links can cause. There were a couple of situations where some of the shown questions were updated and their URLs were changed. To fix these broken links we had to modify the URLs in our code and do a new release. If we had made the system more dynamical (ex. get FAQ links from the database) then the fix would not need such a big effort from us. We could even change the questions on an ongoing basis.

5.4. Implementation details

The implemented feature was solely built by the thesis author with different client-side technologies. The technology stack includes:

- JavaScript (JS) programming language,
- AngularJS framework,
- different AngularJS modules,
- Karma test runner [23],
- Jasmine test framework [24].

TransferWise client-side applications are mostly built in AngularJS framework. Karma is used to run AngularJS tests and Jasmine is a well known and respected Behavior-Driven test framework, which allows to write tests that reflect the behavior desired. Gateway module structure is shown on Figure 5.14.

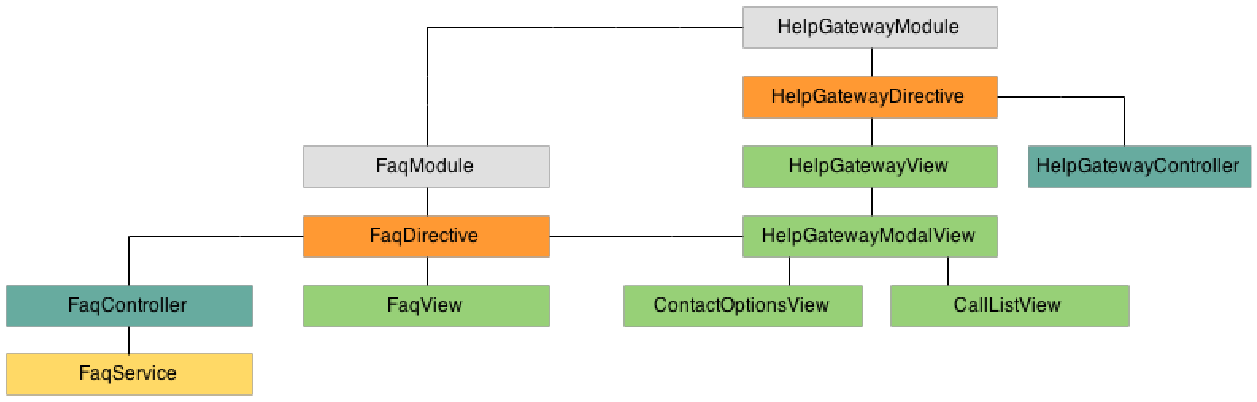


Figure 5.14: Gateway structure

Chapter 6

Future works

6.1. Analysing Service via FAQ

In TransferWise we highly appreciate our customers feedback, but it is hard to get it without annoying them too much. They need to see their benefit in it. This is where the FAQ come into to the play.

When the team looks at the metrics and sees possibilities for improvement in the product then they can make hypotheses of the shortcomings and validate them by showing related FAQ links. When these links get the expected amount of attention from users then they can be sure that the problems exist. At the same time the users get help while not knowing that they are giving a feedback.

6.2. Structuring Customers Questions and Feedbacks

In TransferWise the email contact cases take the longest time to get resolved compare to the Phone and Live chat cases. To lower the resolve time we can build our own structured form, something that we tried with the Bug Report, or use a 3rd party services that focuses on a in-app messaging. New system would replace the Email option in the Gateway.

Conclusions

In this work, we introduced the principles of Lean Software Development, gave a short introduction to Build-Measure-Learn cycle, minimum viable product and test-driven development practices. We defined the problem about the inconsistency in TransferWise help links and the problem with encouraging the users to contact CS before reading the FAQ. We implemented a solution for the stated problems after analysing the alternative solutions from other services.

Our solution to follow above mentioned principles and practices is presented in the form of the new module called "Help Gateway". It is fully covered with automated tests, written in JavaScript on AngularJS platform and is easily integratable to TransferWise. This results in lacking dependencies on server-side and guarantees short delivery time for new features. The Gateway is accessible from every page with a click on the "Get Help" button.

The proposed solution proves that the current link between TransferWise customers and help was insufficient, but the solution has the potential to be even more useful. We have produced some ideas for future development, which we want to try out and experiment in the near future.

Abbreviations

1. LSD - Lean Software Development
2. BML - Build-Measure-Learn cycle
3. MVP - Minimum Viable Product
4. TDD - Test Driven Development
5. CS - Customer Support
6. FAQ - Frequently Asked Questions
7. KPI - Key Performance Indicator

Bibliography

[1] M. Poppendieck, T.Poppendieck, *Lean Software Development: An Agile Toolkit*. Addison-Wesley, 2003.

[2] T.Ohno, N.Bodek, *Toyota Production System: Beyond Large-Scale Production*, Productivity, 1988.

[3] J.P.Womack, D.T.Jones, D.Roos, *The Machine That Changed the World*, Free Press, 2007.

[4] M. Fowler, “Agile Versus Lean”, http://martinfowler.com/bliki/AgileVersusLean.html, 2008 [Accessed April 2015].

[5] “Manifesto for Agile Software Development”, http://agilemanifesto.org [Accessed April 2015].

[6] M. Poppendieck, T.Poppendieck, *Implementing Lean Software Development: From Concept to Cash*, Addison-Wesley, 2007.

[7] All About Agile blog, “Lean Principles”, http://www.allaboutagile.com/lean-principles-4-defer-commitment, November 2010 [Accessed April 2015].

[8] E. Ries, *The Lean Startup*, Crown Publishing, 2011.

[9] Build-Measure-Learn figure, http://www.400minutes.com/build-measure-learn-spiral [Accessed May 2015].

[10] K. Beck, *Test Driven Development: By Example*, Addison-Wesley, 2002.

[11] “What is TransferWise?”, https://transferwise.com/support/customer/en/portal/articles/1567514 [Accessed April 2015].

[12] A. Croll, B. Yoskovitz, *Lean Analytics: Use Data to Build a Better Startup Faster*, O'Reilly Media, 2013.

[13] “What is Zapier?”, https://zapier.com/how-it-works [Accessed April 2015].

[14] S. Weinschenk, “100 things you should know about people: You React To Colors Based On Your Culture”, http://www.blog.theteamw.com/2011/01/12/100-things-you-should-know-about-people-51-you-react-to-colors-based-on-your-culture, January 2011 [Accessed April 2015].

[15] S. Krug, *Don't Make Me Think: A Common Sense Approach to Web Usability*, 2005.

[16] Shaun Cronin, “Cultivating Trust Through Web Design”, http://webdesign.tutsplus.com/articles/cultivating-trust-through-web-design--webdesign-9727 [Accessed May 2015].

[17] “About UserVoice”, https://www.uservoice.com/about [Accessed April 2015].

[18] T. Byttebier, “The Best Icon is a Text Label”, http://thomasbyttebier.be/blog/the-best-icon-is-a-text-label, March 2015 [Accessed April 2015].

[19] AngularJS platform, https://angularjs.org [Accessed April 2015].

[20] “About Mixpanel analytics tool”, https://mixpanel.com/about [Accessed April 2015].

[21] *Angular-translate* module, https://angular-translate.github.io [Accessed April 2015].

[22] *Angular-route* module, https://docs.angularjs.org/api/ngRoute, [Accessed April 2015].

[23] Karma, http://karma-runner.github.io [Accessed May 2015].

[24] Jasmine, http://jasmine.github.io [Accessed May 2015].