Aalto University
School of Science
Master's Programme in Information Networks

Kaisa Halmetoja

Designing a tool for scalable customization of marketplace transaction processes

Master's Thesis
Espoo, November 24, 2019

Supervisor: Marko Nieminen, Professor
Thesis advisor: Juho Makkonen, M. Sc. (Tech.)

Aalto University
School of Science
Master's Programme in Information Networks

ABSTRACT OF
MASTER'S THESIS

Author: Kaisa Halmetoja

Title of the thesis:
Designing a tool for scalable customization of marketplace transaction processes

Date: 24. November 2019 **Number of pages:** vi + 91

Major: User-Centered Design

Supervisor: Professor Marko Nieminen

Thesis advisors: Juho Makkonen, M. Sc. (Tech.)

The marketplace transaction process defines in which ways the users can create value at the marketplace. Being able to customize the process efficiently is essential for the success of the marketplace business. To be useful, the tool for customizing the transaction processes must correspond to the needs of the marketplace operators and developers. These needs were researched and categorized by using job stories. Additionally, users' instinctive ways of visualizing their own processes was studied.

Deriving from the user research an incremental design for customizing the transaction processes was created. A listing of marketplace transaction processes and their details, and a graphical presentation of a selected process and its details was created to inform the users of their processes, supported by documentation. A command line tool was designed to enable users to observe and customize their transaction processes. A visual tool was designed to enable the users to choose their first transaction process and its functionalities.

Analytical evaluation of the designs revealed that informing the users about their process details decreased the number of support tickets requesting information about transaction processes by two thirds. Offering the users a way to choose their initial transaction process was able to meet 66% of users' initial requirements from the process. Together with the command line tool, these two tools offer users the possibility to get all their transaction process-related needs met.

Keywords: User-centered design, transaction process, marketplace, job stories, user research

Publishing language: English

Aalto Yliopisto
Perustieteiden korkeakoulu
Informaatioverkostojen maisteriohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä:	Kaisa Halmetoja	
Työn nimi: Markkinapaikan transaktioprosessien muokkaukseen sopivan skaalautuvan työkalun suunnittelu		
Päiväys:	24. marraskuuta 2019	**Sivumäärä:** vi + 91
Pääaine:	Käyttäjäkeskeinen suunnittelu	
Valvoja:	Professori Marko Nieminen	
Ohjaaja:	Diplomi-insinööri Juho Makkonen	

Markkinapaikan transaktioprosessi määrittelee ne tavat, jolla asiakkaat voivat luoda keskenään arvoa markkinapaikalla. Tämän prosessin sujuva muokkaaminen on markkinapaikan valvojalle tämän liiketoiminnan onnistumisen kannalta oleellista. Jotta transaktioprosessin muokkaukseen sopiva työkalu olisi käyttäjilleen hyödyllinen, tulee sen vastata markkinapaikan valvojien ja ohjelmistokehittäjien tarpeita. Näitä tarpeita tutkittiin ja luokiteltiin käyttötarinoiden avulla. Tämän lisäksi tutkittiin, kuinka käyttäjät itse visualisoivat prosessinsa.

Tutkimustuloksia hyväksikäyttäen luotiin inkrementaalinen suunnitelma transaktioprosessien muokkaamiseen. Käyttäjien tiedottamiseen suunniteltiin dokumentaatio, listaus markkinapaikan prosesseista ja niiden tiedoista, sekä kuvaaja ja mahdollisuus saada yksitoiskohtasta tietoa valituista prosesseista. Muokkaukseen suunniteltiin komentorivityökalu, jonka avulla transaktioprosesseja pystyy tarkastelemaan, muokkamaan ja niiden kelvollisuutta arvioimaan. Ensimmäisen transkatioprosessin valitsemiseen suunniteltiin visuaalinen työkalu, jolla käyttäjä voi valita ensimmäisen prosessinsa ja sen ominaisuuksia.

Suunnitelmien analyyttisen arvioinnin mukaan asiakkaiden tiedottaminen heidän prosesseistaan ja niiden yksityiskohdista vähensi transaktiprosesseihin liittyviä kyselyjä kolmannekseen. Tarjoamalla asiakkaille mahdollisuus valita heille sopiva ensimmäinen transaktioprosessi tyydytti 66% tutkitun joukon lähtökohtaisista vaatimuksista. Komentorivityökalu on työkaluna vaativampi, mutta tarjoaa laajat mahdollisuudet

Avainsanat:	Käyttäjäkeskeinen suunnittelu, transaktioprosessi, markkinapaikka, käyttötarinat, käyttäjätutkimus	**Kieli:** Englanti

Acknowledgements

ln a diary I had written on the first grade I had predicted my future career:

Mitä minusta tulee isona? Minusta tulee psykoloki, kirjailija tai diblomi-insinööri.

So here I am now, fulfilling one of the three professionally, one in secrecy, and one in the kitchen.

I'd like to thank my supervisor Marko Nieminen for excellent advice and convincing me not to panic. I'd like to thank my wonderful colleagues at Sharetribe for showing what is it like to work with integrity and passion with the firm belief in our ability to change the future.

I'd like to give my parents the warmest thanks for the inspiration and love for self-growth, books and making things work better. And I'd like to thank Dan, for convincing me that writing a thesis is not a big deal, making it the reality.

Espoo, November 24th, 2019
Kaisa Halmetoja

Contents

1. Introduction

1.1 Background and motivation

Sharing economy is a term used for a new method of exchanging and trading goods and services from peer to peer through online marketplace platforms, bypassing big corporations that traditionally acted as a middleman (Hamari, Sjöklint & Ukkonen, 2016). The emergence of digital economy, particularly online marketplaces, has given space for this new form of economy (Wang & Zhang, 2012). The sharing economy enables consumers to become producers and micro-entrepreneurs with the aid of digital technologies (Basselier, Lagenus & Walravens, 2018). The field has a wide range of names, from collaborative economy to access economy and from peer to peer to business to consumer marketplaces (Hamari, Sjöklint & Ukkonen, 2016).

The field of sharing economy is on the rise. Forbes (Geron, 2013) has estimated that the sharing economy has created revenue worth over €3.5 billion in 2013, with growth over 25%. Simultaneously, investors have invested hundreds of millions to sharing economy start-ups, estimating it to be the new mega-trend (Alsever, 2013). Simultaneously, the rise of the digital age and sharing economy gives the opportunity to new kind of entrepreneurship. The digital world lowers the barriers of launching new products and trying out ideas. (Richter et al., 2017)

For an entrepreneur interested in establishing an online marketplace, there are multiple choices available. One might build it from scratch, but many offer software solutions to build upon. Building a marketplace is a business and software project, needing skills accordingly. A new entrepreneur can choose from all-goods-included packages for their bases or choose a headless product.

Readily available online marketplace software solutions such as Arcadier (https://www.arcadier.com), Marketplacer (https://marketplacer.com) or Kreezalid (https://www.kreezalid.com) include a certain set of tools that the user is limited to. In this kind of solution, the user is confined to the feature set of the service provider. As the marketplace solutions offer tools and features needed by a generalized marketplace, niche ideas usually have a specified set of needs and features the entrepreneur will wish to have.

API-first software solutions seek to fill the gap between fixed feature set and the often costly and time-consuming process of creating a marketplace software from scratch. The solution offers the basic functionalities of a generalized marketplace together with application programming interface (API). This gives the entrepreneur means to develop

the features they wish to extend the standard feature set of the solution, and considerably reduces the time and resources needed to reach the market.

Creating a generalized marketplace platform often means generalized solutions. At the heart of any marketplace is its transaction process. It can be considered as the engine that defines and guides the interactions between the users and their possible outcomes within a marketplace. Being able to customize this process structure to meet their needs is crucial for many entrepreneurs. The goal of this thesis is to find out, how this need could be fulfilled in the context presented in the next section.

1.2 The company and the product

Sharetribe is a Finnish software company founded in 2011. The company develops online marketplace software with the goal of democratizing the sharing economy. Sharetribe, so forth called the company in this thesis, offers two products: Sharteribe Go and Sharetribe Flex.

Sharetribe Go is easy to use and affordable platform for aspiring marketplace owners. The main value proposition is that with Go, a marketplace can be set up in a matter of minutes, and it requires no technical skills to set it up. It offers a basic layout suitable for many types of marketplaces and a selection of customization tools. However, marketplace functionalities are limited to those that Sharetribe offers and the user interface can be customized only to certain limits.

Sharetribe Flex is targeted to entrepreneurs or companies with moderate funds and access to sufficient technical skills. Flex is directed to those entrepreneurs whose needs exceed what Go can offer. For these entrepreneurs, customizing and scaling their business is their main focus. With Flex, the ability to customize at least the front-end is needed for the successful establishment of the marketplace. This requires programming skills that either need to exist within the team or to be bought externally.

Transaction process defines the marketplace transactions, or in other words, how the users can interact with each other to create value at the marketplace. This is the core engine running the marketplace, and being able to customize this process to meet the needs of the marketplace users is of utmost importance to the marketplace operators. As the transaction process for renting out camping equipment and booking professional trainers differs remarkably, being able to customize the process to suit these needs directly affects the marketplace's success. This thesis seeks to how to empower the users to customize their processes in a way that is both effective and profitable for them.

1.3 Research questions and scope of the study

This thesis introduces the process of designing a tool for visualizing and customizing marketplace transaction processes. In the process, a set of designs is created based on a process of collecting and analyzing information collected with various user research methods. The value of the designs is based on the additional value they bring to the customers using the product.

The main research question is:

What kind of tool for customizing transaction process offers additional value to customers?

In order to find the answer to the main research question, it first must be understood what is the additional value it brings to the customers. It is known that the main users are marketplace operators and marketplace developers. Additional value is something that answers to these users' needs and requirements for the tool that will be designed. Therefore, to answer the research question, two secondary research questions are needed:

1. *What kind of needs do marketplace operators have considering customizing their marketplace's transaction process?*
2. *What kind of needs do developers have considering the tool for customizing transaction process?*

Section one of this thesis introduces the background and goals of this research. Section two concentrates on creating an understanding of the processes and concepts that are used in designing the new tooling. The literature review sets the base for further review by creating an image of how related studies have addressed the issues presented in this thesis. After creating an understanding of the overall context, section three concentrates on defining a set methodologies that can be used to understand, categorize and analyze user research amongst users in this particular context. These are needed to create a basis for answering the two secondary research questions.

After the methodologies have been described, in section four these methods are taken into use and the results of user research are presented. These results give us answers to the secondary research questions. In section five, these results are put in to use and designs are created based on them. These designs present an answer to the main research question. To

be confident that the suggested designs answer the users' needs and therefore bring additional value to them, the designs are evaluated in section six.

The scope of this thesis is limited to creating designs that answer to these users' needs. The thesis concentrates on introducing a process of solving a novel business problem and creating design artifacts as solutions. To properly introduce the research process and to properly create designs and conduct the evaluation processes, the scope does not include the actual development of the tool, usability evaluation, or iterating the designings. Therefore, the evaluation consists of evaluating how well the designs are able to answer the needs that arise in the user research phase. Also, evaluating the effects of the designs on the company's performance is not included in the scope.

2. Literature review

To create a better understanding of the context around the research question and this particular application area within software development, the literature review will examine topics in closely related areas. As the subject of this thesis belongs to the area of software design, the literature review will concentrate on creating an understanding of what are the best practices in software design and user research in this area.

The literature review will concentrate on creating an understanding of how to better approach complex design problems (section 2.1) and what others have done while facing similar problems (section 2.6). It will review the best practices and approaches in information visualization (2.2) and process modeling (2.3). To understand the context of use and users, the review will seeks to create an understanding of how to design for developers (2.4) and how is a common developer tool, command line interface, used (2.5).

2.1 Designing complexity

Rittel and Webber (1984) introduce wicked problems. They introduce these as problems that are characterized by unstable requirements, complex interactions among subcomponents and critical dependence upon human cognitive abilities. These are often also characteristics of designing complex tools within information systems.

The transaction process itself is not complex, but editing it brings novel and complex challenges, as the logic behind the simplified graphical model includes several dependencies, some of which cannot be validated computationally. The requirements for the process are set, but they include a vast quantity of various combinations and the process can take several paths. There are complex interactions between the subcomponents. Finding all the dependencies between various components requires human inspection as the intended combinations cannot be validated without it. For example, a certain transition can be accessed through multiple nodes (states) and will be passed parameters according to the path it has been accessed to. It is possible to try to use parameters that are not passed through this particular path or that will present themselves incorrectly if accessed through another path than intended.

Customizing the transaction problem quickly becomes a complex problem, especially if any kind of graphical WIMP approach is considered, as including necessary data,

dependencies and element combinations into a single graphical interface will quickly turn hard to use due to its complexity.

The need for flexibility and maintaining several complex elements is what makes this specific problem complex. Hevner et al (2004) describe how design science research addresses complexity. In design science research, the problem is simplified by explicitly representing only a subset of the relevant means, ends, and laws or by decomposing a problem into simpler sub-problems. Such simplifications and decompositions may not be realistic enough to have a significant impact on practice but may represent a starting point.

Any design problem can be divided into smaller subproblems as any system can be divided into smaller subsets. The needs that users have for inspecting and editing the process can be divided into simpler problems. Job stories, introduced in the Methodology chapter, are a tool used for dividing customer needs into subproblems. Similarly, the needs and cognitive processes, such as learning to use a new technology and understanding graphical descriptions, can be addressed in subsets. The transaction process itself can be divided into multiple subsets. For example, handling aliases and versions, making changes to the process structure, changing parameters and validating the process.

Hevner et al (2004) further suggest that progress can be made iteratively as the scope of the design problem is expanded. As means, ends, and laws are refined and made more realistic the design artifact becomes more relevant and valuable. In this thesis, the users' needs are divided into steps by using the aforementioned Job stories that implement solutions incrementally. This also provides incremental value and makes it possible to expand the scope piece by piece.

Simon (1996) reminds that given the wicked nature of many information system design problems, it may not be possible to determine, let alone explicitly describe the relevant means, ends, or laws. In such situations, the search is for satisfactory solutions, without explicitly specifying all possible solutions. The design task involves the creation, utilization, and assessment of heuristic search strategies. That is, constructing an artifact that works well for the specified class of problems.

Remembering this in design work encourages the designer to looks for solutions out of the box and looking for satisfactory solutions instead of perfect solutions. There is the danger of creating solutions that seemingly solves all users' problems, but simultaneously are difficult to use and include unnecessary steps for most use cases. This kind of solution could be described to be a Swiss army knife - it does everything, but badly. If users' experience and the ease of use is considered a valuable part of the process, it can be argued that if the solution is unnecessarily complex, it is not satisfactory.

It can be seen that complexity reduces the efficiency of the product used and therefore also the user satisfaction. Frokjar, Hertzum, and Hornbak (2000) define efficiency as “the relation between the accuracy and completeness with which users achieve certain goals and resources expended in achieving them”. Efficiency is indicated by task completion time and time used for learning to use the product.

Usability composes of various parts, including ease of use, user satisfaction and efficiency (Nielsen, 1993). However, effectiveness does not guarantee satisfaction nor does ease of use guarantee with effectiveness. This is an important aspect when designing a tool for complex problems. However, the same goes other way around - the ease of use does not guarantee user satisfaction. A solution may be complex but efficient or easy to use and inefficient. The balance between complexity, efficiency, and ease of use must be balanced against the user requirements. In some cases, users are more ready to use more time to learn to use complex tooling than use inefficient tooling.

Creating satisfactory, easy to use solutions for complex problems can also be achieved by removing some of the complexity - according to Hevner et al’s (2004) theory, this means removing some of the requirements, interactions, dependencies or human dependencies. Hevner et al. suggest removing some of the user requirements, which can be done through prioritization. Reducing interactions or dependencies may lead to decreasing the quality of the solution, but it is also possible to find subcomponents of the process that do not have dependencies. It is also possible, that the solution realm offers a way to address the complexity in a way that does not require simplification.

More user-centered than process centered ways to address complexity in design are making use of the existing models that the users have and chunking. When users’ mental models match how the actual system model works, people find the user interface intuitive (Fitzpatrick, 2016).

On way to address complexity in design is chunking the information that is presented. Chunking is a term used in cognitive psychology to refer to a process of binding individual pieces of information together to create meaningful entities (Neath & Suprenant, 2003). These pieces of information can be, for example, a group of numbers or syllables.

The term was first introduced by George Miller in 1956 in his article *The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information*. Miller found out that our immediate memory is limited by the number of items, which is seven chunks of information, plus or minus two.

Chunking reduces the cognitive strain and therefore leads to a high amount of collect recalls. In other words, chunking is a form of strategic encoding that involves transforming the set of data into a compressed form and therefore it can extend the capacity of working memory. It can be used as a strategy for learning and recalling information. Decreasing the load for working memory is essential for the successful performance of many cognitive processes. (Bor et al., 2003)

2.2 Information visualization

The book *Information Visualization: Human-Centered Issues and Perspectives* by Kerren, Stasko, Fekete, and North (2008) defines information visualization (shortly infovis) as a research area that is focused on aiding people on data analyzation by the use of tools and techniques of information visualization. The subject of this thesis is not directly related to helping people extract information from large datasets, which is what most infovis principles concentrate on. Rather the focus of this thesis is visualizing process descriptions, which can be considered as a subset in infovis. Infovis techniques, such as guidelines on pattern recognition, help in this subject.

Visualization can reduce time required for searching relevant information and decreases the amount of needed cognitive resources and memory capacity (Card, Mackinlay & Shneiderman, 1999). With transaction processes, visualizing the process would help the users to understand the connections and marketplace use flow. It can be said that visualization would with no doubt decrease the cognitive resources needed to get an understanding of the transaction process as a whole. It can also help the users to search and locate relevant information locating in a certain part of the process, and to memorize it.

Information visualization is not, in itself, a scientific tool, but a tool for helping users make insights on the collected data (Kerren et al., 2008). There is no specific way the information visualization tools should be used or another goal for their use than the subjective experience of the user. As such, the only way to validate developed visualization tools is to create a working implementation where the tool is used (Purchase et al. 2008).

It is arguable where information visualization ends and design work begins. In infovis, the goal is to help the user to make insights from the data. The infovis tools or principles do not consider user experience or usability in interaction. They do not include guidelines considering beauty, excitement or pleasure of use, which can be considered as goals of

design (Hevner et al, 2004). They do not either consider ease of use or efficiency, as the tools consider graphical output without interaction.

Interacting with the data presentation has been recognized to help the users to engage and make them more prone to understand the intended messages (Heer et al., 2008). This is a notion that is especially valuable when the dataset consists of complex information or if the user has to modify the dataset. As the goal of this thesis is to create a tool for customizing the transaction process, it is essential that the user is able to understand all the relations in the process. Being able to interact with the data and discover relations and connections deepens the user's understanding of the process and enables them to do more informed decisions when it comes to customization.

Keim et al. (2008) recognize the problem of information overload. The term is used when there is a danger of the user getting lost in the data and losing the connection to the task at and. This may be due to the irrelevance of the data presented, the data being presented in an inappropriate way or processed in an inappropriate way. This can be used as a guideline when deciding what information should be shown and when. Keim et al. suggest that only relevant data should be represented and it should be grouped so that it creates meaningful entities.

Keim et al. (2008) guide to concentrate on the relevance of the information for the task at hand. They also suggest concentrating on how interaction could facilitate problem solving and decision making. To support this, the task at hand should be clearly defined so it can be recognized what information is needed. This can be addressed by creating clear sub-problems to be solved and offer information related to those problems, with the right timing. Chunking the information so that it is visible only when it is searched for decreases the amount of clutter and mental load.

Keim et al. (2008) also note the problem that fully automated data processing methods represent the results in a way that is ineffective in communicating the knowledge it contains. This might be the case with automatically made visualizations of the transaction process, too. For example, graphs that are created by automatic functions are not designed to be easy to read or to emphasize important information over less important information. Even though a graph would include all the information that a graph formed by a human would, the information might be presented in a form that does not convey the information in an effortless way. This can be easily addressed by bringing a visual design component into the process, instead of trusting automated processes.

2.3 Process modeling

Aguilar-Saven (2002) describes the term business process as follows: “A business process is the combination of a set of activities within an enterprise with a structure describing their logical order and dependence whose objective is to produce a desired result.” According to her, business process modeling enables a common understanding and analysis of a business process and can provide a comprehensive understanding of it.

This description of the benefits of process modeling matches the goals of this thesis. The transaction process is, at the core of it, a technical description of a marketplace business process. For the operators and developers to be able to communicate effectively and plan their work and operations, they need to have a comprehensive understanding of the process.

Phalp et al. (1998) suggest that different approaches for modeling business processes attempt to satisfy different goals. Phalp et al. distinguish two approaches to business process modeling: pragmatic approach and rigorous paradigms. Pragmatic approaches seek to capture the process and present it in an understandable way. This kind of presentation has to be easy to understand and fast to learn to be efficient in communication. As models can be complex, it is common that they are found to be difficult to understand, so it must be considered what kind of audience will be using the model. The other approach, rigorous paradigms, is used for process analysis. In this approach, the model should be able to show both dynamic and functional aspects of the process. The transaction process description doesn’t fit directly in either of the categories, as the goal is to inform in-depth and offer some tools for the analysis. However, the exact goal is not to use the model for analyzing the process, but rather to understand it and the technical aspects of it.

As the transaction process is a technical description, it has some characteristics that differ from traditional business models, such as transition parameters. This might cause it to be too different from any traditional, non-technical process modeling techniques that they can fully be applied to use. However, understanding the basics of process modeling techniques and the various ways to model processes offers a background for creating a model suitable for this project. Aguilar-Saven (2002) introduces, amongst others, flowcharts, role action diagrams and unified modeling language.

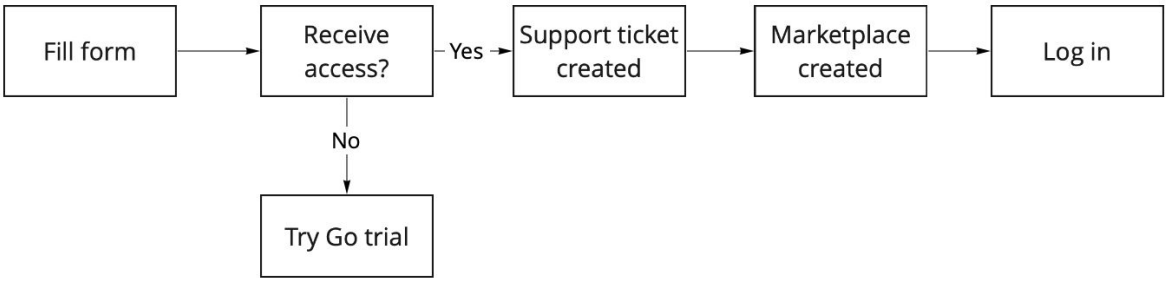

Figure 1. Example of flowchart

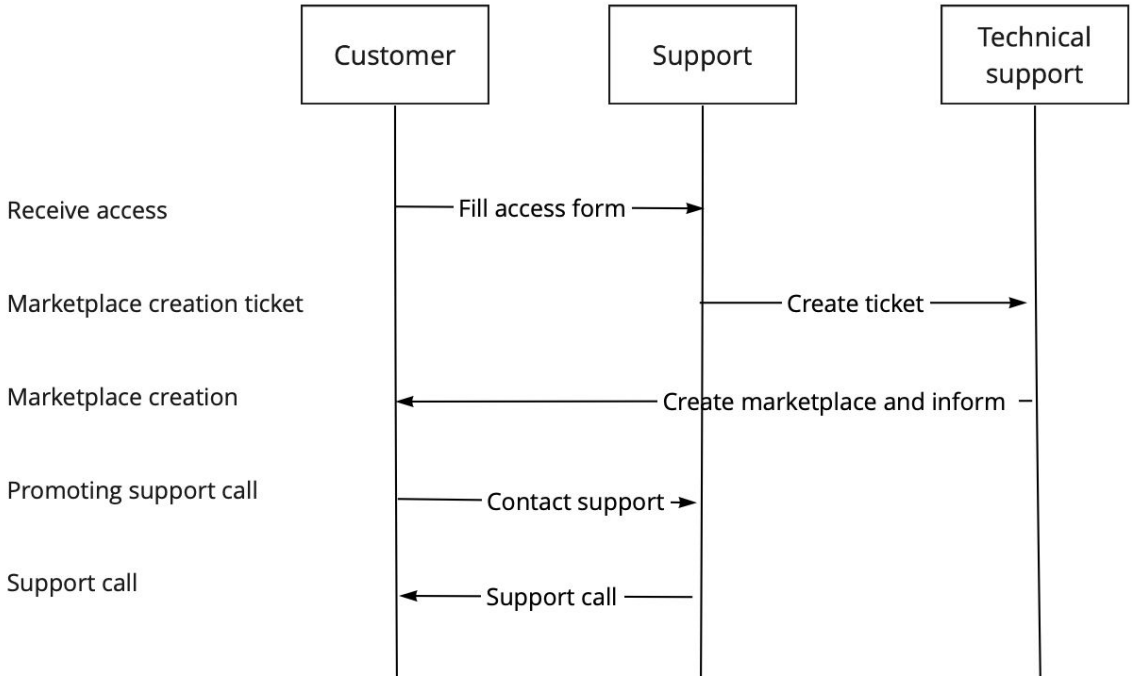

Figure 2. Example of RID

Flowchart can be used to describe almost any process, and it is used to graphically represent program logic sequences, work processes, organizations and other formalized structures (Lakin et al., 1996). The main advantage of flowchart is its flexibility and communication ability (Kraljic et al., 2008). The form offers a flexible way to present processes while being easy to understand. Flowchart offers a way to make processes easily recognizable and brings out the process flow direction and any inconsistencies or dead ends.

Role interaction diagrams concentrate on the actions by different actors, roles. The activities are shown on left vertically, while the roles are shown horizontally on top. The actions are represented as arrows pointed from a role to another and further explained by text. Role interaction diagrams are intuitive and easy to read, but more rigid than flow diagrams. The diagram is best used only with only few actors and activities and gets hard to read with a greater amount of information. The diagram is mainly used for workflow design and coordinating activities between different actors. (Kraljic et al., 2008.)

Unified modeling language (UML) is a standard object-oriented modeling language. It is specified into visualizing, constructing and documenting software systems and artifacts, but can also be used for business modeling. UML consists of nine different diagrams out of which Activity diagrams are the closest to the research subject of this thesis. (UML, 2002.)

Activity diagrams reveal the work involved in changing the object states (UML, 2002). It consists of object states and activities. According to UML, activities are drawn as boxes, containing the name of the operation. The arrows indicate transitions that are activated by the completion of an activity and the direction of the activities. Guard conditions can be written next to the arrows.

Table 1: Comparison of process modeling techniques

	Description	**Strengths**	**Weaknesses**
Flow Chart	Graphical representation of the flow of actions	Simple and easy to understand The base for many different models	Doesn't support sub-activities
Role interaction Diagram	Activities connected to roles in a matrix	Clear distinction between roles	Tend to be messy with many roles and/or activities
Activity Diagram	Graphical representation of a workflow; steps of activities	Includes different symbols to mark various actions	Remembering various notations might prove difficult

These process modeling techniques give us an understanding of how it could be possible to depict the transaction process or what kind of elements could be taken and combined from various techniques.

2.4 Developer experience

In their article Developer Experience: Concept and Definition (2012), Fagerholm and Münch seek to define the term developer experience and explain the need for such term. They define developer experience as "a concept that captures how developers think and feel about their activities within their working environments, with the assumption that an improvement of the developer experience has a positive impact on software project outcomes." Developer experience is seen to derive from experiences relating to development infrastructure, feelings about their own work and the value of one's own contribution.

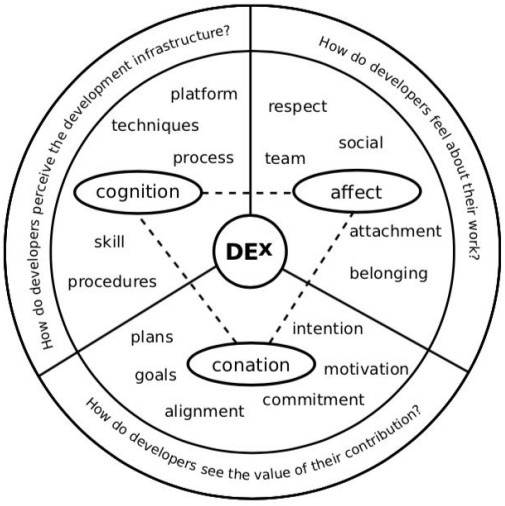

Figure 3. Conceptual framework of developer experience (Fagerholm & Münch, 2012)

Fagerholm and Münch (2012) note that a positive human experience is a strong indicator of a successful development process, while various tools and methods can increase the productivity of already highly skilled and motivated teams. How developers experience their working environment, their tooling, and the work itself is the factor to successfulness.

The subject of this thesis is to design a tool for developers and operators alike. Having efficient, usable development infrastructure is one of the key parts of developer experience. The tool for process customization is meant to bring more autonomy and efficiency to developers' work with Flex. Another important aspect of concentrating on developer experience is that developers are, in increasing amount, the key influencers in purchasing decisions (Bhowmick, 2018).

Designing a tool for developers is designing a tool for a specialized profession. These tools are often built to enable developers to do something they have not been able to do before. The tools are made to enhance functional power, the developers' capability of getting things done, and ease of use is not the first criterion when designing these tools.

In his article Designing for developers (2018), Arin Bhowmick introduces how developers' needs differ from everyday application users. Bhowmick advises to use tools that are already familiar to developers and warns from assuming that a graphical interface is necessary. He reminds us that many developers are using command line tools daily and they are used to specific tools. Especially for repetitive tasks, many developers prefer the fastest way

Bhowmick (2018) mentions that "it's worth remembering that many developers can handle more complexity in their products than other users we might be used to designing for." For many, efficiency is more important than simplicity. Bhowmick recommends making the tool as simple as possible for the new users but also recommends providing the more experienced users with as much power and control as possible for them to be able to work fluently. Additionally, it is good to remember that even the new users are already professional on the field and can handle complexity.

2.5 Command line interface

Command line interface (CLI) is a way of interacting with software and operating systems by giving commands as textual inputs. In the early era of computers, CLI was the primary way of interacting with computers. (The Linux information project, 2004) Even as graphical interfaces have nowadays replaced command line interfaces, many in technical professions such as developers or system operators use command line interface and tools in their daily work. As command line interfaces are already a part of developers' toolset, they might offer an effective way to study and edit the transaction process.

In CLI, the commands for the system are given usually as short textual inputs consisting of few letters to few words. CLI is an efficient tool for experienced users and also enables users to automate tasks, but for inexperienced users, learning the commands may be difficult, as well as learning how to use the tool without mistakes (Westerman, 1997). In general, CLI is expected to offer minimal feedback, which may slow down learning and make the work more prone to errors. As also people who are not familiar with using CLI must understand and be able to communicate about changes to the transaction process, command line tool might not answer to the operators' needs.

There are several studies suggesting that users perceive command line interfaces as harder to learn and use than graphical interfaces. Shneiderman (1987) found that using direct manipulation interfaces, such as WIMP interfaces, enhanced the accuracy, facilitated learning and diminished errors when compared to command line interfaces. Hasan and Ahmed (2007) studied the influence of interface style to perceived ease of use and usefulness, finding that the participants favoured menu-based interface over command-based interface. However, they noted that the participants in the study had little experience on using similar systems, which might have an effect on the results.

Feizi and Wong (2012) studied how interface designers and software developers perceived the learnability and ease of use of a graphical software application that uses both graphical interface and command line interface. The tasks included such as setting a background colour for canvas (GUI), changing image transparency (CLI) and creating a mouse over event for an image (GUI or CLI). They concluded that CLI was found more difficult to learn and use than the GUI. However, it can be argued that as the system was using both of the interface styles, how to results apply to systems using only one of the styles. Also, as the application has not been designed for one approach, it might have lowered the overall usability of the product as the product lacks consistency.

However, all the studies do not agree and there are mixed results. Davis and Bostrom (1992) studied the effect of interface style on the learnability of computer system and found that there was no significant effect on ease of use. In their study, Wiedenbeck and Davis (1997) compared novice users' perceived usefulness and ease of use of a software application in cases where they used either direct manipulation interfaces or command driven interfaces. They found no significant effect on perceived usefulness and even though they found an significant effect on perceived ease of use, they noted that the effect was small.

A possible explanation for these differences of results in these studies is that the results may relate to the nature and complexity of the task at hand rather than to the user interface style**.** Mathieson and Keil (1998) found that the perceived ease of use was dependable on the fit between the system and task at hand, rather than the system itself.

It seems that the perceived ease of use and usefulness is also related to the users' previous experience with CLI. Wiedenbeck and Davis (1997) found that the users that had previous experience on a different user interface style had very negative attitudes towards any other style. The study suggests that users' attitudes towards software were significantly influenced by their prior experience. Wiedenbeck and Davis summarise that "these results suggest that users' attitudes toward software are strongly influenced by their past history of

usage, including what interaction styles the user has encountered, and this should be considered in the design of software and training programs." As the subjects of the studies mentioned above had previous experience on using computer, and most computer users are used to graphical interfaces, the negative attitude towards new interface styles might have biased the results of the aforementioned studies.

These studies suggest that the significance of perceived usefulness may even be much higher when the user are not novice with the tooling and area of work. With complex tasks, CLI might prove even more useful, than GUI. Davis (1989) explains, systems that offer more functionalities are perceived as more useful than systems that offer less functionality. When tasks become more complex, the CLI may better receive the correct task-system fit.

Feizi and Wong (2012) argument that "CLIs afford more options than their equivalent GUIs, leading to greater flexibility available for users or one can perform a task by using command that its function is not supported by its GUI counterpart." In general, a user is able to achieve more and in a shorter time than using GUI. In CLI, commands are usually short and executing them happens using only the keyboard. For example, deleting a file will require several clicks and confirmations in using GUI, but in CLI this can be done with a single command. For users who value efficiency, this brings along satisfaction and feelings of usefulness. In some cases, especially when considering operating systems, some functions can only be run through CLI.

One possible argument against the usability of CLI is the lack of feedback and lack of visual cues. Many command line tools offer minimal feedback, showing it only when asked for or in case of errors. However, what an inexperienced user might describe as a lack of feedback, might experienced user find as lack of unnecessary distractions. Lack of feedback and confirmation can lead to errors more easily. Therefore proper help and just the right amount of feedback is necessary.

The lack of visual cues and using only typed commands also causes memory load, as the user has to remember all the commands. However, having a high memory load and learning phase doesn't mean bad usability. In professional tools used for carrying out complex tasks, the expected learning time can be longer. Having help available makes the learning process easier and using autofill, commonly used syntax, and full sentence commands without abbreviations make the learning process easier.

2.6 Case study: Intercom Chat Bot

Intercom is a company offering a messaging platform for businesses to support and connect to their customers through various platforms. In August 2018 they released Custom bots - a feature that offers their customers the ability to customize the chat bot to meet their business's needs (Donhue & Shepard, 2018).

Intercom's process of designing and creating shares many facets with the subject of this thesis. They have a highly complex tool - a chat bot - and customers who needed customizability. Their work is rather novel and it includes making a complex, technical tool to be customizable by those who do not have technical or programming background.

The background of Intercom's chatbot resembles the background of the subject of this thesis. Donhue and Shepard (2018) open up the background of the project in their article, explaining that even though their chatbot offered an excellent user experience, the clients wanted to be able to create their own conversation flows in their own order. Sharetribe's default transaction process is functional and meets very well the basic needs of a marketplace, but as customers want to be fully in control of their marketplace user experience, they wish to have the power to customize it to their needs.

Julien Zmiro, product designer at Intercom, described Intercom's approach in his article *The hidden cost of design complexity* (2017): "If we go back to our initial definition of complexity ("many" parts "entwined" together), it seems that there are two very high level ways to prevent it: one is to fight the "many" by reducing the amount of parts to the core essential, the other is to fight the "entwined" by untangling those remaining essential parts – in other words, to make the parts as few and as independent as possible." Zmiro does not claim all complexity to be unnecessary. However, he does suggest that while creating complex tools, unnecessary complexity should be removed. The unnecessariness of a feature or component is defined by the value it brings, and Zmiro suggests that everything that is not absolutely required, should be removed to avoid unnecessary complexity.

The other way to reduce complexity that Zmiro (2017) suggests, is to untangle the elements that are included in the process, tool or software. This means making screens modular or chopping up the process to multiple steps. The transaction process includes multiple steps, such as choosing what actions the user can take and what kind of notification can be sent. Even though these are tied in the technical description of the transaction process, there might be a way to untangle these steps from each other. This creates a clearer, more modular path for the user.

According to Donhue and Shepard (2018) when they were designing the chatbot, they decided not to go with a visual one-to-one mapping tool as a bot builder as it would quickly become overwhelming. Rather, they decided to modulate the tool in line with Zmiro's (2017) suggestion. Their decision to avert from graphical builder because of complexity is a noteworthy warning when considering how to implement transaction process editor. It's possible that in this case the design complexity and the complexity that the user would have faced using the product would have decreased the usability of the product significantly.

Donhue and Shepard (2018) decided to use a very familiar messaging view, similar to most of direct messaging applications, to help the user to benefit from their already existing mental models. An important feature in their application was available templates that the users could engage with and start forming their own ideas on how they would modify their application. Their hypothesis that templates would encourage people to make their own bots proved to be right: twice as many bots were made from templates than those that were made from scratch. They note that "it's essential to provide some stable, solid, safe first steps for your customers to take. Otherwise many will just back away and close the door you convinced them to open." This is another important lesson to take from Intercom - the customers need proper guidance and starting points when taking a new tool into use.

2.7 Synthesis

In the literature review, the problem context around the research question has been examined from supporting aspects.

Methods of process and information visualization have been examined to create an understanding of how business processes and data groups are visualized and to find best practices in doing so. The gathered understanding can be used both while conducting the user research and while designing the tools, to recognize commonly used components. The review has confirmed visualization as a useful way to promote common and comprehensive understanding of processes (Aguilar-Saven, 2002), supporting the motivation to study how people would prefer to view their processes. The understanding of different process models that are commonly used may help to recognize the components used in the user sketches. For the design phase, the literature view has supported the idea of not taking one-to-one mapping approach (Donhue and Shepard, 2018), but rather taking

an approach that supports modulating the design solution to simpler entities (Hevner et al., 2004, Zmiro, 2017).

To create a better understanding of developer experience and tools commonly used by developers, the definition of developer experience and aspect influencing it were examined. Additionally, studies examining the usability of command line interface was studied to better understand the practices of using command line interface as a tool. The review gives a firm basis of knowledge for how to design for developers. Bhowmick's (2018) studies suggest using tools that are already familiar to developers, and not assuming that a graphical interface is necessary. Similarly, he encourages to trust in developers ability to use complex tools in their work. Davis (1989) suggests to that systems that offer more functionalities are perceived as more useful than systems that offer less functionality.

3. Methodology and research context

3.1 Design science research

According to Vaishnavi, Kuechler, and Petter (2004), design science research is a set of techniques for performing research in the field of information systems. The aim of design science research is to improve information systems by creating new knowledge or innovative artifacts and analyzing the use and performance of these artifacts. Fuller (1992) puts the goal of design science research more simply - the goal is "to solve problems by introducing into the environment new artifacts". Artifacts are the products of the design science research process - things or processes such as user interfaces or system design guidelines.

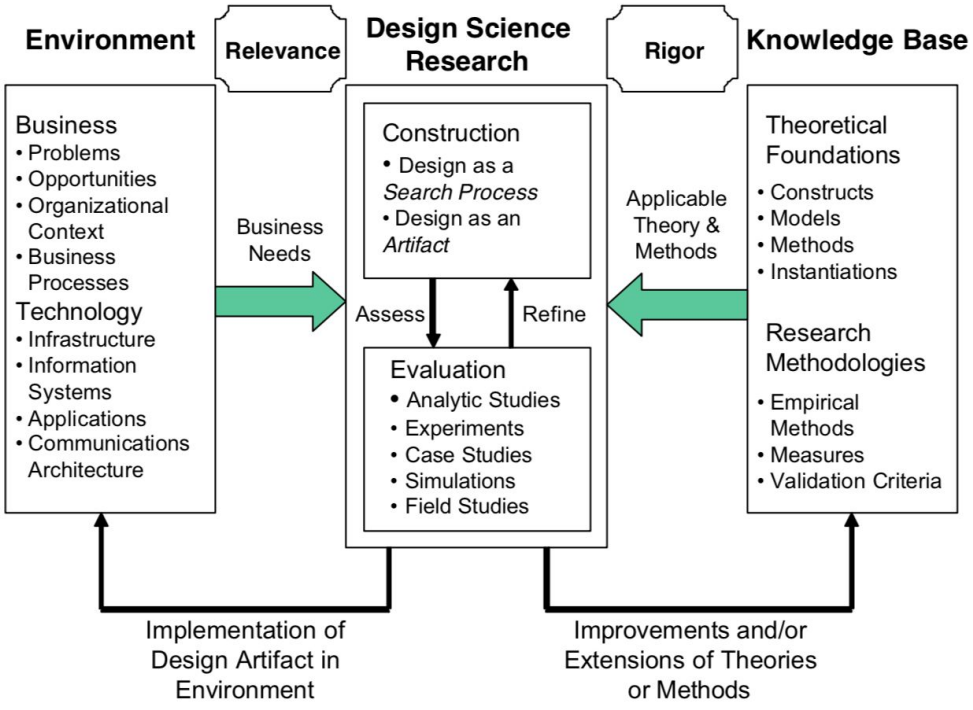

Figure 4. Design science research framework (Hevner et al., 2004)

Hevner et al. (2004) present a framework for understanding, executing, and evaluating design science research in IT. The main components are the business needs that rise the need for design science research and creation of new artifacts. The research is based on existing knowledge base that provides material for creation and evaluation of the artifacts. Hevner et al. offer guidelines to design science research. These are:

1. **Problem relevance**. The addressed problem must be relevant to solve. Often in business organizations this is related to maximizing profits. The research must present an artifact that is implementable and demonstrates a clear contribution to the business environment solving a novel problem.
2. **Research rigor.** The research must be conducted with rigor, meaning that the proper theoretical foundations and research methodologies must be used. The produced artifact must be evaluated in its applicability and generalizability, determining how well the artifact works.
3. **Design as search process**. Creating the artifact is a process of discovering an effective solution to the problem existing in the environment.
4. **Design as an artifac**t. The result of the process, artifact, must be "effectively represented, enabling implementation and application in an appropriate environment".
5. **Design evaluation**. The efficiency and quality of the artifact must be proven by validated evaluation methods. The artifact must solve the problem it was created to solve.
6. **Research contributions**. The process creates contributions by solving the problem vi the artifact as well as contributing to the knowledge base.

In this thesis, design science research framework is used for defining a process of user research, artifact design and analytical design evaluation for designing a tool for customizing transaction processes. The aim is to create an effective process for designing tools in this context and to find viable solutions whose performance has been proved. This will contribute to the research community by proving how the process and artifact are able to solve this particular business problem in this environment.

3.2 Product and process context

3.2.1 Sharetribe customers

Sharetribe focuses on providing marketplace software for entrepreneurs building their marketplace business. Marketplace founders are often either solo entrepreneurs or small startup teams, consisting of two to five people. In this thesis, this person, group or

company is referred to as customer. Usually, the customer or a defined person from the team of the customer works as the marketplace operator - this means as the person leading the marketplace development and operative decisions. To functionally establish a Sharetribe Flex marketplace, the customer needs to have development resources. This means that the customer has to have a developer as a member of the team or hired for the project.

The technical skills of the marketplace operators vary significantly from novice to advanced. Operators with novice understanding are able to grasp the technical details when explained, but not able to produce technical information or content. The users with moderate technical skills are able to understand and produce technical descriptions, but not able to create advanced technical modifications. This is important to keep in mind when the focus is on how to design the tools that are used to customize the marketplace.

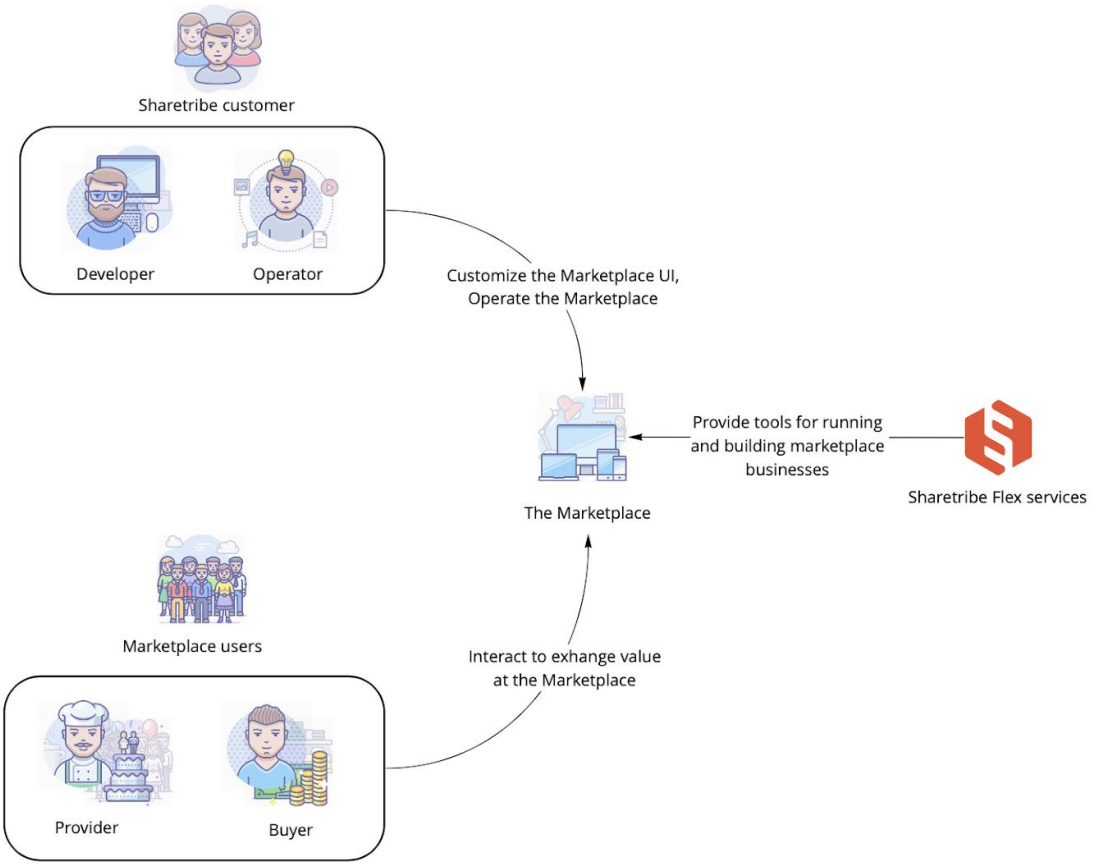

Figure 5. Sharetribe Flex users

The company: Sharetribe

The product: Sharetribe Flex

Customer: Sharetribe customer who has access to Flex. This can be one person, team, a startup or a company.

Operator: Sharetribe customer who makes operative decisions about running the marketplace. In companies without development resources, the agent who hires the development resources.

Developer: A person developing custom software on top of Sharetribe Flex. A developer might be working for operator as freelancer or employer. It is also possible that the marketplace operator is also the developer.

User: A person who interacts with the product or feature. A general term used in human-centered design as described in ISO 9241-210 (2010). In the context of this thesis the term is used to refer to operators and developers.

Marketplace user: Any registered user of a marketplace
Buyer: Marketplace user with the intention of buying goods or services.

Provider: Provider is a user that is allowed to post a listing and/or has posted at least one listing to the marketplace.

Sharetribe Flex consists of four parts: Flex backend, the Console, the APIs, and Flex Template for Web (FTW).

The console is a user interface for marketplace operators, where operators can build, run and track their marketplaces. In other words, it is the management tool for the marketplace. It includes listings of all the users, listings, transactions and reviews and moderator functionalities for managing these. It also includes a "Build" section with tools for managing marketplace functionalities. These tools are aimed for developers.

The Flex Template for Web (FTW) is a web template application that is offered by the company to form the basis of the marketplace user interface. It is backed by the API that supports authoring and discovering content, managing user accounts, and the purchasing flow. The API's include the Marketplace API and Admin API. The marketplace user interface is hosted by the customer and can be customized as they will.

The backend is hosted and maintained by the company. The company offers all the backend functionalities, including databases, payment handling, authentication and the application programming interface for these.

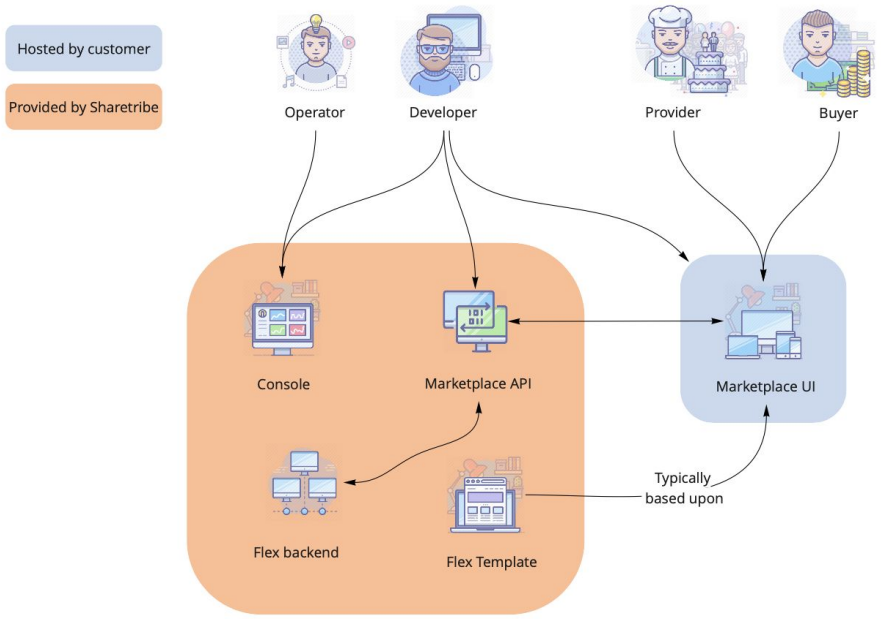

Figure 6. Flex components

3.2.2 The transaction process

The transaction process is in the core of the marketplace, handling the interactions between the end-users - the buyer and the provider. The transaction process starts from an interaction between a buyer and a provider (usually initiated by the buyer). It includes a description of the different paths the process can take and actions that have to be executed along the process. Additionally, notifications such as emails that are tied to the transaction process are included.

As each marketplace operator have their own preferences on how they make business, the need for the transaction process to be customizable is crucial. Different kind of marketplaces have needs specific to their field. For example, for a business that rents cameras it might be highly important to have a functioning insurance and inspection process for the gear. If a marketplace is used for renting scooters for tourists, it might be important to have a possibility to extend the renting period or to store the driving license information.

It's not possible to create a single transaction process that would fit the varying needs of multiple different marketplaces. It is unclear exactly how variating the needs of different fields are. As there are plenty of variating marketplace ideas it is understandable that the need for pet host marketplaces and camera rental marketplaces differentiate, but it similarly unclear how much they actually do variate.

Businesswise having an incompatible transaction process reduces the quality of the user experience. The marketplace customers have needs specific for a certain field. Processes that do not answer these are either cluttered with distracting features and steps or missing important phases.

At the time of writing this thesis, customizations to transaction process were done by contacting the company's customer support and requesting a change. The change is done manually by the developer on customer support sift and pushed to the customer. The task of updating a customer's transaction process includes communication with the customer, making the changes to the process in question, validating the process, deploying the changes to production, updating the process alias and communicating the new process version to the customer. Handling a single request can take up to a day of work.

It is evident that this way of working does not support growth as the company wishes nor the customers' wishes of fluent and flexible development work. Therefore, an alternative way of customizing the process has to be developed. As being able to scale the product and offering an excellent developer experience are the company goals of 2019, finding an answer to the arising problem is the subject of this thesis.

The transaction process is defined as a data format called extensible data notation (EDN), a subset of Clojure, which is used to represent programs and as a data transfer format (Hickey, 2018). Below is introduced a very short process both in graphical and EDN form. In this process, a request is preauthorized and accepted automatically, and the process is marked as complete after the booking period has ended.

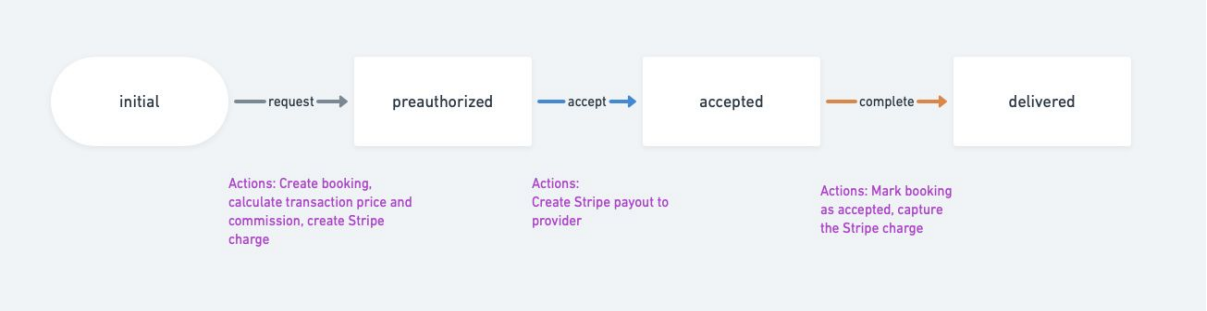

Figure 7. Short transaction process example in graphical form

```
{:version 2
 :states {:state/initial {}

          :state/preauthorized {}

          :state/accepted
          {:delayed-transitions
           [{:transition [:transition/complete {}]
             :at #tegel/timepoint :tegel.time/booking-end}]}

          :state/delivered {}

 :initializer :tegel.action.initializer/init-listing-tx

 :transitions
              :transition/request
              {:name :transition/request
               :actor #{:actor.role/customer}
               :actions [[:tegel.action/create-booking {:prevent-overlapping? true}]
                         :tegel.action/calculate-tx-nightly-total-price
                         [:tegel.action/calculate-tx-provider-commission {:commission 0.1M}]
                         :tegel.action/stripe-create-charge]}

              :transition/accept
              {:name :transition/accept
               :actor #{:actor.role/provider}
               :actions [:tegel.action/accept-booking
                         :tegel.action/stripe-capture-charge]}

              :transition/complete
              {:name :transition/complete
               :actor #{:actor.role/system}
               :actions [:tegel.action/stripe-create-payout]}


 :dag  [[:state/initial :transition/request :state/preauthorized]
        [:state/preauthorized :transition/accept :state/accepted]
        [:state/accepted :transition/complete :state/delivered]]

 :notifications {:notification/new-booking-request
                 {:after-transitions #{:transition/request :transition/request-after-enquiry}
                  :to #{:actor.role/provider}
                  :template :mail.template/new-booking-request}

                 :notification/booking-request-accepted
                 {:after-transitions #{:transition/accept}
                  :to #{:actor.role/customer}
                  :template :mail.template/booking-request-accepted}

                 :notification/money-paid
                 {:after-transitions #{:transition/complete}
                  :to #{:actor.role/provider}
                  :template :mail.template/money-paid}

 }
```

Figure 8. Short transaction process example in EDN form

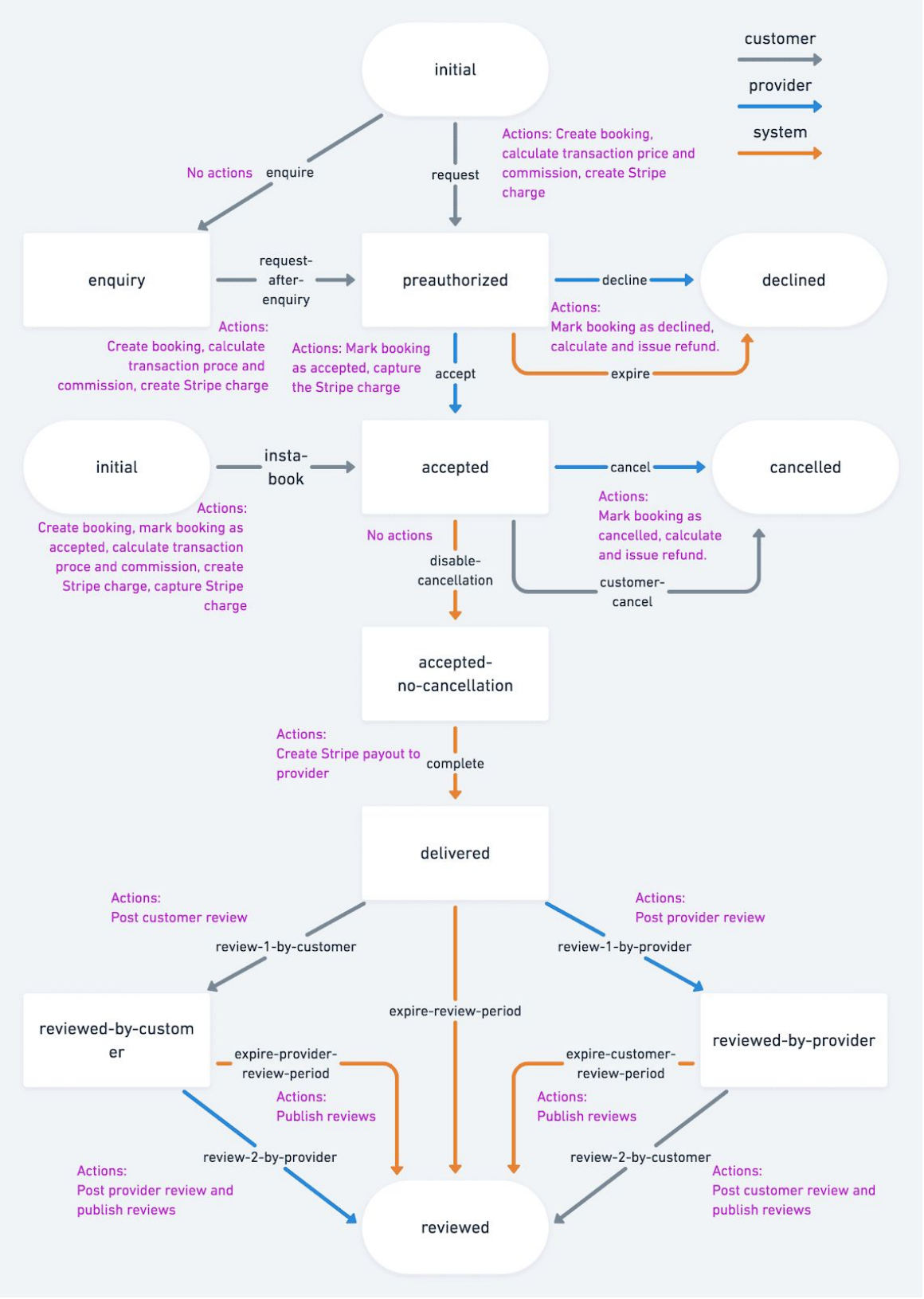

Figure 9. Sharetribe Flex default transaction process. (Sharetribe, 2019)

The default transaction process for Flex marketplaces is visualized in the graph above. In this flow, the user can request a booking or send an inquiry to the provider, after which they can request a booking to start the transaction. The booking request can be either accepted or declined by the provider, or it can expire automatically after seven days. If the request is accepted, the customer's credit card is charged and the payment is held by the marketplace. When the request has been accepted, it is still possible for the provider to cancel the booking, in which case the payment is refunded to the customer. The booking will automatically move to state"delivered" after the booking period has ended, and the payment is released to the provider. Finally, both the customer and provider can review each other.

3.2.3 The transaction process terminology and components

Transaction

Transaction is an interaction process between a buyer and a provider where value is exchanged between them. The purpose of any marketplace is to facilitate transactions between its users.

Transaction process

A transaction process is a description of the possible events and actions in the marketplace, and the outcomes the transaction can have. It defines how the different parties at the marketplace interact to create value. A transaction process consists of a set of transitions between a set of states and a list of notifications. It is possible to have more than one transaction processes on a marketplace. By default, Sharetribe offers two transaction processes to choose from: nightly booking and hourly booking.

Directed acyclic graph

The transaction process is defined as a directed graph. This means that the transaction can take certain paths and it can only move into one direction appointed by the transitions.

Process version

A new version of a process is made every time changes are made. As the transaction process is critical to the functionality of the marketplace, it is crucial that there is a functioning versioning method. This also enables the marketplace to try out multiple process models as well as covers the marketplace end-users from sudden changes in their transactions.

Process alias

Aliases can be understood as tags pointing to a version to be used in a certain context. Alias can be appointed to a version and one version can have one, zero or multiple aliases pointed to it. For example, the version used in production can be tagged with alias “production” and simultaneously another version of the same process can be tagged with alias “test” and used for testing purposes.

An example of using alias: a marketplace operator decides to change the transaction process so that buyers can cancel requests if there is more than 48 hours to the beginning of the booking, but the change only affects daily bookings. The programmer selects the daily transaction process and commits the changes. They wish to test the change in production. The version is tagged with “test” and is used only with the transactions that are pointed to use version with “test” alias. As the version is tested, the alias is changed to “production”. The transactions that have already started before the new version is taken into use will continue with the old transaction process version.

State

A state expresses the state a transaction process can take. All states must be accessible by a transition, except for the initial state. Delayed transitions are tied to states.

Transition

Transitions define what states can be accessed next from one state, taking the transaction forward. They define what happens at each step of the process as a list of actions (e.g. create booking). Transitions can be only made by certain actors defined in the transition. Actions are tied to transitions.

Delayed transition

Delay transitions are transitions that happen after a certain amount of time has passed after the transaction has arrived to a certain state. It's possible to express times like "6 days after 'request'", "1 day and 12 hours before the booking start time" or "the earliest of the booking start time and 3 days after 'request'". In these transitions, actor is always the system, meaning that the transitions happen automatically. Delay transitions are tied to transitions.

Actions

Each transition defines a set of actions that can take place during that transition. Actions can, for example, calculate the total price and commission of the transaction,or create a payment.

Parameters

Actions need specific parameters to function correctly. For example, commission percentage is needed to calculate the commission, and transaction price is needed to create Stipe events. Parameters can also be used elsewhere, for example in notifications.

Notifications

Notifications are messages, currently only emails that are sent to defined users in certain points in the process. Notifications are tied to transitions. They can be sent immediately after a transition or scheduled to be sent at a certain time of the process, similarly to delayed transitions.

3.3 Research methodology

ISO 9241-210 (2010) defines the process for user-centered design to include six steps that happen in an iterative process. These steps are:

1. Plan the human centered design process
2. Understand and specify the context of use
3. Specify the user requirements
4. Produce design solutions to meet user requirements
5. Evaluate designs against requirements
6. Designed solution meets user requirements

In this thesis, section 3.2 concentrates on explaining the context of use. This understanding is based on information gathered by Sharetribe sales and technical support personnel.

Section four (*User research*) will concentrate on specifying the user requirements by conducting user research among Sharetribe customers. The aim of this research is to better understand the requirements these users have and to further develop an understanding of the context of use. Additionally, it was needed to know what kind of pre-existing schemes

the users had when it came to visualizing their processes. This included gathering an understanding of how the users visualize their transaction processes, what components they find important and what styles of visualization they find natural for them.

The chosen methodology for user research was user interviews (introduced in section 3.3.1) and user sketching (introduced in section 3.3.2).

User interviews were conducted to better understand the user needs, and user sketching for a better understanding of the pre-existing schemes. As secondary research questions define two main user groups within Sharetribe customers as marketplace operators and marketplace developers, the user research was conducted within these two user groups.

Table 2. User interviews

Research type	**Participant type**	**Number of participants**
User interview	Operator	5
User interview	Developer	4

For the user interviews, five marketplace operators and four Sharetribe developers were chosen. The interviews were semi-structured, with a set of predefined questions and a possibility for open questions and clarifications (see appendix A and B).

The questions presented for the operators concentrated on creating an understanding of their team and how their development process works. When it came to the transaction process, the goal was to understand how familiar they are with the transaction process, what are their needs considering customizing it, and if they have done any customizations to it.

The questions presented for the developers concentrated on what they think about the development process and communication with the marketplace operators, how familiar they are with the transaction process, if they have done any customizations to it and how they felt about the customization process. The main goal was to understand the developers' needs and requirements for fluent customization of the transaction process.

Out of these nine interviews, user needs and requirements for the transaction process customization tool would be drafted. For categorizing the needs and requirements, a framework called Jobs to Be Done would be used. It is introduced in section 3.3.2.

Table 3. User sketching

Research type	**Participant type**	**Number of participants**
User sketching	Operator	1
User sketching	Operator with developing experience	1
User sketching	Developer	2

For user sketching, four users were selected. Out of these users, one was a developer and three were marketplace operators, out of which one also worked as a developer. For this interview, the users were asked to explain how their transaction process works at the moment, and after this given 10 minutes to draw the process on a blank sheet of paper (see appendix C). The goal of this sketching interview was to better understand how the users depict the process in their minds and what components they use in the drawing. These results would be used in the design phase.

3.3.1 User interviews

In his book *Observing the user experience: a practitioner's guide to user research (2003)*, Kuniavsky notes that "to really know the user's experience, you have to ask him or her about it." User interviews are part of almost every user research. User interview is a formal, standardized interview that seeks to reveal the user's experience and remove the perspective of the interviewer (Kuniavsky, 2003). Interviews are used in qualitative research in which facts, user insights, opinions, experiences, attitudes, and behavior are collected (Rowley, 2012).

In this thesis, semi-structured interviews were used for interviewing the users. The goal of semi-structured interviews, according to Wilson (2013), is to "gather systematic information about a set of central topics, while also allowing some exploration when new

issues or topics emerge". Wilson further introduces semi-structured interviews to include predefined questions and also space for open-ended exploration similar to unstructured interviews.

Wilson (2013) recommends using semi-structured interviews when there is already an understanding of the topics to be interviewed, but further details are still needed. The interview follows an interview guide that includes a list of topics and questions as well as probes and prompts, and can take from several minutes to a few hours.

The basic structure for most user interviews includes an introduction to the subject, asking questions about the structured topics, having time for general questions and open dialogue, and finally wrapping up the interview (Wilson, 2013).

3.3.2 Sketching and mental models

Sketching is a method often used in design work. User sketching is a design method in which the user is asked to draw on paper, other surface or in a drawing application a sketch of the idea, process or design they have. The sketches give the designer a more concrete understanding of what the user has in mind and a point of reference for further questions.

Tohidi, Buxton, Baecker, and Sellen (2006) suggest that giving the user a possibility to make sketches of their idea generates reflective user feedback, opposed to reactive user feedback that most usability testing methods offer. According to them, enabling users to sketch their ideas and thoughts facilitates reflection on the task at hand and provides a rich medium for communicating ideas. Tohidi et al. suggest user sketching as a complementary method to more traditional user testing methods to provide rich information in a time and cost effective way. The traditional way of user testing is that the user explores ready made sketches by thinking aloud or answering questions orally. This is most likely to cause reactive critic rather than reflective suggestions for improvement. In their study, Tohidi et al. found out that when the users are asked what they would change about designs, the common reaction is an inability to come up with suggestions. However, when the users were asked to sketch solutions, they were able to find out several motifs and patterns.

Analyzing user sketches offers a possibility to more in-depth analysis in shorter time. Analyzing sketches allows the researcher to find answer questions that were not even recognized in an interview setting, but were revealed in users' sketches. (Tohidi, Buxton, Baecker & Sellen, 2006.)

Sian Townsend describes in her article *Understand your users' mental model*" (2016) how she used sketching in her work with Intercom. The research showed that not only the employees had confusion on using different terminology while talking about the product, their mental model of it varied from each other. Sketching helped people to recognize and verbalize problems they had and differences in their thinking.

Townsend (2016) offers simple guidelines on conducting a sketch research. The first thing is to decide the goal and focus of the research. After this, the research questions should be designed to be open ended in order to prompt the participant to start sketching. Ask directive or clarifying questions while the sketch is processing in order to help the users think and come up with new directions for their sketches.

Michelle Fitzpatrick introduces in her article *How users understand new products* (2017) how matching system and user models help users to take the system into use. Her primary thesis is that if a user's mental model matches the system model, the users find the user interface more intuitive. To understand users' mental models, Fitzpatrick suggests user interviews and asking users to sketch how they believe the system works. After this, one can look for commonalities. Another aspect to take into consideration is the language people are using when they talk about their sketches. For example, if a person talks about sending a message, the same word should be reflected in the user interface. The nouns that they use are components they expect to be in the interface.

3.3.3 Jobs-to-Be-Done (JTBD)

Jobs-to-be-Done (JTBD) is a theoretical framework and tool used to describe users' needs and thriving factors when buying products. It is mainly in the fields of marketing and innovation. Even as the tool is used on product management, academic research that applies JTBD on product development is rare.

There is no unified opinion on the exact definition of JTBD, but the theories are rather similar. Christensen, Anthony, and Roth (2016) introduce JTBD as based on the theory that when a user buys a product, they "hire" it to do a specific job for them. Ulwick's (2016) view on the subject is rather similar, as he sees that "people buy products and services to get a job done". Klement (2016) has a slightly different viewpoint, defining JTBD as a process that people go through when they use a product to make their life better. In this thesis, we take the approach that Christensen et al and Ulwick offer.

An example of a situation for using the JTBD framework could be a customer buying coffee in the morning. Applying JTBD theory, the coffee is filling a need they have - in this case, most often, a way to feel more awake. Therefore, the same JTBD could be solved by a cold shower or a bright light lamp. However, if someone would just observe people buying drinks in the morning, they might make the mistake to try to sell them flavored water or cacao, not answering the actual need (or job) they have.

Christensen et al (2016) argue that understanding the jobs that customers want the product to solve, developing new products is more successful than using user segmentation. The user jobs concentrate on the needs users have instead of situational or demographic factors. With this focus, it is possible to find underlying needs that are common over the demographic factors.

However, jobs to be done alone might offer insufficient data for making design decisions. For example, the technical skill level of the user must be taken into account while designing technical products. The job can be similar to many groups of people, but their ability to use the product that solves a particular job may vary. Even if demographic knowledge does not reveal customers' needs, it reveals their ability to use the products offered. The need for understanding users' competence and their ability to use products are considered essential in human-centered design. Therefore it is important to remember that JTBD is a tool amongst others and suitable for addressing certain problems.

To define the JTBD for a certain product one must gather knowledge on customers' needs. Ulwick (2016) proposes that this can be done by using any of the traditional interviewing methods used in user research. These include personal interviews and ethnographic methods such as observation. What Ulwick considers important is to consider different types of job executors such as the end user, the purchase decision maker, and the product support team. For this thesis, the main job executors are operators, developers and Sharteribe's support team.

To further define the users' needs, Paul Adams introduces Jobs Stories, a tool invented within Intercom to turn JTBD into usable format, in his essay Abandoning personas: the story behind Job Stories, published in book *Intercom on Jobs-to-be-Done* (2017) by Intercom. Intercom had considered using user stories but found them too engineering driven instead of customer driven and not based on research. Therefore, they created a process of their own concentrating on situations, motivations, and outcomes.

[When _____][I want to _____][so I can _____]

The first part of the formula focuses on the situation, the second part on the motivation and the last part on the expected outcome. The formula leads the users to thinks about the situation when the problems are encountered, what is the user's motivation for solving it and what they achieve by solving it (Intercom, 2017). It offers a way for the designers to evaluate if the goal is achieved without restricting how the problem itself is solved.

Alan Klement further describes how to define Job Stories (Intercom, 2017; Klement, 2013). He suggests starting with a high level job and then identifying smaller jobs which help to resolve the higher level job. He suggests examining how people solve the problem currently to define the current job story. After this, it is possible to create the job stories and then the solution based on the stories.

One way to translate the job stories to solutions within the field of software development is to consider the way they translate to user interface components or views. The user needs can also be answered and supported by using support guides, offering customer service or changing the company business process.

Even as job stories are created to scrutinize customers' needs, it is possible to approach them with solutions in mind. This causes the job story to be biased towards the direction of the solution.

Consider the following job story: "When I'm customizing my marketplace, I want to choose the transaction process, so I can better meet my customers' needs." The verb after "I want to" defines the customer's needs and also gives an implication on how they want to approach the subject. Choose means that they want pick from existing processes. If the verb would be "customize", it would mean that the customer wants to create or modify a process matching their needs. The job story already includes an inclination to the solution and can, therefore, guide the design work to predefined direction. It needs close scrutiny and wording to make the stories match actual user needs and not already existing ideas of the solution. Creating specific customer stories based on user research guides us to better outcomes.

The job stories function well when analyzing the design process and possible solutions. If the proposed solutions do not match the user needs, it is easy to find them unsuitable. If a proposed solution does answer the user's needs, at least one step of the process has been validated.

3.4 Design evaluation

The evaluation of the created artifact is part of the design science research framework. Hevner et al. (2001) write about design evaluation, stating that the purpose of it is to prove the efficiency and quality of the product by using validated methods. According to them, the goal is to validate that the artifact solves the problem it was created to solve, and the business environment defines the requirements upon which the evaluation is based. Similarly, the fifth point of iterative user-centered design point according to ISO 9241-210 (2010) is to "evaluate designs against requirements".

In this thesis, the artifact is evaluated by using analytical evaluation. Analytical research refers to research that involves critical thinking and evaluation of facts and information already available and analyzing this information to make a critical evaluation of the material (Kothari, 2004). In this thesis, quantitative and mixed-methods are used to support analytical evaluation (Onwuegbuzie & Combs, 2011).

Quantitative analysis deals with data in the form of numbers, which can be examined by mathematical operations (Walliman, 2017). The primary purposes of quantitative analysis are to measure and test hypothesis (Walliman, 2017). The quantitative analysis process consists of determining the questions to be answered, determining the sample, selecting the methods needed to answer the questions, selecting the analysis tool and interpreting the results. (Holton & Burnett, 2005)

Qualitative data analysis is based on data expressed in forms of words rather than numbers. In qualitative analysis, a theory can be evaluated by collecting qualitative data and analyze it by using selected tools. In qualitative analysis, coding can be used to organize the data under labels or tags. Coding is an analytical process that includes recognizing what the data is about and categorizing it under a describing label. (Walliman, 2007)

Quantizing is a method of translating, transforming, or converting qualitative data into numerical (Sandelowski, 2009).

The scope of this thesis includes background and user research, designing the tool and evaluating the designs. The scope does not include the actual development and user testing. Therefore, the designs were evaluated by using analytical evaluation rather than empirical evaluation.

In the user research phase, two distinct areas were found: informing the users about their transaction process and customizing the transaction process. The designed solutions were evaluated separately.

The designed solutions were successful if the users' needs in both categories were fulfilled. In the first category, this meant that the users were able to access needed information by themselves, without reaching to Sharetribe support. If this was true, the number of support tickets should decrease and their quality increase. In the second category, successfulness meant that the customers were able to add those process functionalities that they wished to be able to add to their process. This was evaluated by inspecting if the customer built processes could also have been built with the designed solutions.

The evaluate how well the users are informed about their transaction process, the quantity and quality of transaction process related tickets was evaluated. The hypothesis was that the quantity of the tickets would decrease while the quality would increase. If these requirements would proof true, the solutions could be considered successful.

The first question to evaluate was how the amount of transaction process related support tickets has changed. The data set used is transaction related technical support tickets from January 2019 to July 2019 and it was evaluated by quantitative methods. The tickets were categorized by using coding methods, and the amount of tickets was calculated on a monthly basis. These tickets were categorized to general tickets and tickets considering the transaction process, and the transaction process related were categorized further to better reflect the problems they were related to. One of these categories was "tickets requesting for information". It was observed how these types of tickets were affected by the release of the designed solutions. Finally, it was evaluated how the amount of tickets had changed during the inspection period.

The second question to evaluate was how the quality of transaction process related support tickets requesting for information had changed during the inspection period. The same set of tickets was evaluated by qualitative methods and tickets were coded by categorizing them according to their quality. These categories were quantized by giving each ticket a numerical value corresponding to the quality. In this way, the monthly average quality was examined and it was evaluated how the designs had affected the ticket quality during the inspection period.

The second evaluation category concentrated on the customer's ability to customize their processes. As the command line tool is able to satisfy all user needs within the possibilities of the technology, the initial process selection tool was evaluated.

As developing the designed tools for this was out of the scope of this thesis, the evaluation was done by evaluating how the designed tools would have answered to the need of the marketplaces built during the last six months. The evaluated question was how many of the features that customers had built in their transaction processes during the last six months

could have been built with the initial transaction process selection tool. The hypothesis is that it is possible to find similarities in the customers' requirements considering their transaction process. The tool would be successful, if it could satisfy half of these needs.

Here the analyzed data set was 27 transaction processes build during the last six months. The structure of these transaction processes was analyzed by recognizing what kind of functionalities that differed from the Sharetribe default transaction process the process had, and these functionalities were categorized under labels. Finally, it was evaluated how many of these functionalities would have been possible to build with the designed tools.

4. User research

4.1 Target user group

The current target group for Sharetribe Flex is operators (entrepreneurs, startups or companies) with developing resources. This means that the customer should have or they should be able to frequently, if not at all times, have access to a developer or developers who are able to do modifications to the code base. With this target group, people without continuing access to development resources are not considered in the designs.

This division was done as Flex as a product is directed to entrepreneurs, startups and companies that are preferably past their early stage and have already tested their marketplace idea or are otherwise ready to commit to the project. Building a marketplace with a headless approach means committing to a software project and therefore also into having a developer or a development partner in the project.

This leads to some expectations on how work is distributed between various parties within the company. The operators are expected to run the daily business and operate the marketplace functions through Console. The Console is a place for managing the marketplace and sharing information about the marketplace and its functions. The operator can do slight modifications to their marketplace, for example, change the email contents or search rules for their marketplace. Any changes to the UI and functionality will require development experience.

The developers are expected to have certain competency when it comes to tools commonly in use in the occupation. This means familiarity with tools such as command line interface, software such as GitHub or similar, and development environments. It is expected that the developers have existing ways of working and tooling in their development practice.

The customization of the product is done within the developer's own development environment and not in Console. No extensive modifications can be done through Console, but it is rather a place for retrieving information and doing minor modifications. This also prevents possible mistakes that are caused by modifications done by people with insufficient understanding of the requirements. With this division, it is not possible, for example, add new email templates within Console, as they should be tied in the Flex backend in ways that require development skills.

At the current moment, the targeted customer group is strictly limited to those who have access to developers. It is possible that this limits the customer group extensively, as development resources require money and time. This is especially the case when all changes require a developer. Out of the 20 first customers that launched their marketplace in 2019, 11 had developing resources in their team from the beginning. This means that having access to developing resources constantly raises the likelihood of successful onboarding and launch of a marketplace. Sharetribe seeks to connect customers without development resources to development partner with expertise of building on top of Sharetribe products.

Currently, the user is able, for example, edit the email contents without a developer as they have access to email templates in Console. This, however, will not be possible with transaction processes or transaction process notifications. The transaction process is the engine that runs the marketplace. Changes to this process should be made with care, as they affect the core functionality of the marketplace. However, it is somewhat unlikely that frequent changes are needed after the initial launch of the marketplace.

4.2 Defining user needs

To answer the research questions on users' needs, user interviews were conducted. The aim of these interviews was to get a proper understanding of our customers and their teams, and to acquire information on what kind of needs they had considering customizing transaction process.

4.2.1 Operators

Five people from marketplace operating teams were interviewed to gain an understanding of the level of technical understanding and requirements of use for the transaction process tool. The roles of these people varied from marketplace managers to product managers. At the time of the interviews, Flex was still in an early stage as a business with only a few customers.

In addition to the interviewed people, a further understanding of the customer base was gathered by following customer sales calls, which provided a more extensive cut of the customer base. Based on this material, the technical understanding of the customer varies from rudimentary to proficient. For many understanding complex technical language caused difficulties. Often these people were considered less probable to establish a

successful marketplace, and were recommended a development partner. This points to the fact that our current customer base is adept with technical notation, but in order for the company to grow from the current status, it might be necessary to provide more understandable instructions and ways to hire a developer for the team.

The team size of the interviewees varied from one to seven people. Only one out of five customers had a dedicated developer with a permanent contract, as the others had hired either freelance developers or hired developers from consultancies. The design and product decisions were done in house, either by the marketplace operator or within the team. In all of the cases, developers and other team members were not co-located. Communication was mostly text-based and done either via email, online communication platforms such as Slack, or by calls.

During the interviews with the operators similar issues and needs around transaction process editing arose. The findings are gathered into sections below.

Finding 1: It is unclear what is meant by the transaction process and what can be solved by modifying it.

The term “transaction process” is not understood in a unified way. The users were asked what they think the marketplace transaction process included. Two out of five interviewees connected the term only as related to payment process and it did not rise an image of the transaction flow as a whole. When an operator was asked what do they think the term transaction process means, one interviewee answered “Do you mean the payment process?” This indicates that the user connected the term only to the payment process, and not to the whole flow.

The three users that had worked longer with Sharetribe, were able to connect the term to the booking flow as a whole. When asked, one operator described their process step by step starting from search all the way to returning the goods, including transaction process related steps such as notifications. However, they also included steps that were not included in the transaction process, such as the search. This means that even though the operator was able to connect the term to the whole process, it was not completely clear what was included in the process and what was not.

It was similarly unclear to interviewees what is related to the transaction process and what does it affect, hence also what kind of possibilities modifying it offers. It was common to all of the interviewees that they were not able to connect the problem they faced to being connected to the transaction process. Consequently, it also did not occur to them that the

problems could be solved by modifying the transaction process. For example, one marketplace operator wanted that the marketplace customer is sent an email that reminds them about their booking, which could be done by modifying the transaction process notifications, but the operator didn't recognize this possibility. Another operator wished that they would be able to add a customer identification to a customer's profile. This could be done through extended data added to a transition in the transaction process, but the user was not aware that they could request this from the Sharetribe team.

As the transaction process includes not only the different states of the process and different actions the actors can make, but also the technical actions (e.g. releasing the money to provider), notifications and delayed transitions (e.g. expiring requests) it is understandable the not all of the various functionalities are connected to the same process description in users' minds. This restricts the users' understanding of how they could approach their problems and how they could take advantage of the full potential of their marketplace transaction process

Finding 2: It is difficult to comprehend the transaction as a whole, and therefore it is difficult to communicate.

Closely related to the first finding, users had difficulties in comprehending the transaction process as a whole and thus also communicating their problems or needs relating to the process. This was common to all of the interviewees, and previous examples apply also to this finding.

The difficulty in understanding the process as a whole was partly due to insufficient documentation and communication from the company's side. At the time of writing this thesis, the transaction process documentation included only a graphical description of the default process and few paragraphs explaining the basic functionality, though not in detail. There was no description of how the process works or what it includes.

When one of the operators was asked how familiar they are with their transaction process, they answered that they "don't understand what is going on in the black box". The operator felt that the transaction process was a "black box" that somehow handled all considering marketplace's transaction without them actually knowing how. This underlines how difficult it felt to understand it for the customer.

As in creating service blueprints or customer journeys, it is common to think only about one successful route and the end-user's actions as a single line of actions. When asked, all of the users described only one successful booking flow. However, it must be noted that

the users were describing the process vocally, and therefore might not to concentrate on details, but rather describe the process as simply as they can.

Only two out of five interviewees had had any changes done to their transaction process. In these cases, the requests considering modifications to the transaction process were communicated in text to the Sharetribe team. It both cases for the provider and Sharetribe developer to reach a mutual understanding of the requested changes, multiple conversations and corrections were needed. Two of the interviewees suggested that having access to a graphical description of the transaction process would help them to better communicate their needs.

Finding 3: The needs for modification were similar, but details varied.

The needs and requests the interviewees had considering the transaction process related changes were similar. Common requests were canceling opportunity for buyers (3/5), having user ID stored (3/5) and possibility for rescheduling (2/4) and deposit (2/4). All of the interviewees expressed a desire and need to be able to change the transaction process related emails.

However, smaller details such as parameters used in notifications, exact delays of transitions, and collected extended data varied. Also, one of the customers had made significant changes to their transaction process that made the process radically different from the original and from others' processes. These changes presented various ways of answering to the same problems users had. Also, how the user wanted their problems to be solved varied. For example, users had varying ideas when and where the ID could be stored.

None of the users had regular need to make changes to the transaction process, but rather the need arose when the interviewees had had their marketplace running for some time and they were able to learn what was needed. All of the interviewees agreed that it was probable that the transaction should be modified according to needs when the marketplace was built, and there shouldn't be much need to change it afterward. As an exception, changes to commission percent was thought to be more regular.

This would suggest that even though some of the changes required were similar, many of the details varied significantly. It would seem that even though many of the marketplaces require small changes or additions, some are interested in modifying the process further to meet complex needs that differentiate profoundly from the original process.

Finding 4: A change in transaction process inevitably causes changes to the marketplace user interface.

As the transaction process defines what happens on the backend, it has to have corresponding elements in the user interface. For example, if a possibility to cancel a request is added, there will also have to be a way for the user to trigger the action. Another example would be storing information in the user's extended data - to store that information, there must be some way to provide it through the marketplace customer facing part. It is rarely possible to do changes to the transaction process without it requiring user interface changes.

It is unlikely that any changes to the transaction process would be done without a developer. Additionally, there are many parts of the process that are interlinked in ways that require developing skills.

Summary:

The operators' needs concentrate on understanding how transaction processes work and how they can best use them to their advantage. The operators' needs for transaction processes vary, and all the details cannot be predicted. As the changes affect the user interface and data structure, it is clear that the transaction process changes always need the help of a developer.

4.2.2 Developers

For this research, four people from Sharteribe's development team were interviewed. At the time of writing this thesis, there was no possibility for people outside Sharetribe team to modify the transaction process. All the modifications were made by contacting Sharetribe's support and requesting a change. Making the requested changes was the responsibility of the person who was having the technical support sift. The changes were done manually and no proper tooling existed, but rather only some commands to push and pull correct files. There was no proper validation, but the validation was on the responsibility of the maker.

Even though Sharetribe development team makes the changes to the transaction processes, the consequenting needs for user interface changes are on client's responsibility. When a Sharetribe team member has updated the transaction process and pointed the suitable alias to it, they inform the customer on the changes they've done, including the required information for front-end changes.

The people interviewed have done varying amount of work with transaction processes. One of the interviewees had done only one transaction process change and the most experienced one had been developing the transaction process engine. The common consensus was that the process was hard to comprehend in the beginning, but got easier over time. As most transaction processes are similar to each other, learning to know the default process helps understanding the modified versions and thus decreases the time needed for familiarizing oneself with them. Still, the current EDN does not support fast learning or easy comprehension.

All agreed that making changes to the processes was tiresome. This was mostly due to inadequate tooling and the insufficient validation. There is a risk of semantic mistakes as well as logical errors. The current validation is only partial and doesn't validate e.g. action parameters. Also, logical errors such as transitions to unexisting states are not validated. It takes time and careful scrutiny to notice errors that relate to the graph's rightful construction.

Finding 1: It is difficult to comprehend the transaction process description

It was commonly agreed that even though the EDN offers an exact description of the process and is not difficult to read in itself, it fails to offer an easy way to comprehend the process in its entity. Especially the relations between different states and transitions are difficult to follow.

One developer said that "it still requires a lot to understand, what can be included in the process description". They continued by saying that "if the process is very mysterious and has to be examined more thoroughly, I have to draw it". Another developer put it more directly: "It is hard to get a comprehensive picture."

The transitions and states are described apart from each other in the EDN description. To follow the information flow from one to another, the developer has to move in the text file from state descriptions to directed (asymmetric) graph description to find out the possible following transitions and from there to transition description. This is found to be time consuming and making the process unnecessarily difficult. One developer mentioned that "the states and transitions are located in different places in the EDN, so it is hard to find which is related to which."

There was a frequent request to visualize the relations between states and transitions, as this helps with both comprehending the process and the possibilities it offers. It was though that it is easier to notice the relations between various states and transitions as well as the possibilities for adding states or transitions.

One developer felt that having a graphical description would help them to understand the process description more quickly and would help to understand what changes they could make. Another developer wished that they could have the EDN and graphical description next to each other. They both suggested that having a graphical description would lessen the possibility for mistakes, as it would be easier to notice if e.g. a transition would not lead to a state.

Finding 2: Editing transaction processes is prone to errors

As the transaction process defines the process of booking or buying a listing, it is in the core of marketplace functionalities. A mistake in the transaction process description can cause the transaction to fail. This means disrupting the core functionality of the marketplace and can break down the whole process. Therefore editing the transaction process must be done with utmost carefulness. Two of the four interviewees described editing transaction process as "scary" due to this need for flawless execution. One developer said that "making changes is scary, as you cannot be sure if you are doing the right things."

There were two common problem areas: semantic mistakes such as typos, and logical mistakes such as adding a transition without proceeding state. The validation in the current editing process is existent, but inadequate to catch all possible mistakes. It does inform on most syntax errors, but not for all. For example, action parameters are not validated.

There are also many possibilities to make logical errors, which aren't validated at all. These include making transitions where they cannot be done, adding states without having transition that leads to them or making actions that need parameters from other actions but which have no access to them. On interviewee offered the following example:

A developer makes transition AB from state a to b which includes action *book_slot* for booking a particular time slot. After this, they make transition BC from state b to c where they include action calculate_booking_cost that calculates the cost of this booking. After this, in transition CD from state c to d, they make an action stripe_create_charge that makes a payment with a payment provider called Stripe, which needs the result of action calculate_booking_cost to function. Later, they add transition BD from state b to d that

skips the state c and therefore transition BC, which includes action for calculating the booking cost. Therefore, the action for creating a Stripe charge cannot function.

Finding 3: Customer's awareness of their processes varies

As operators make requests for transaction process changes, their awareness of the process and its implications vary. In some cases, operators are not aware that the problem they are encountering is related to transaction process. Rather, they contact support with a problem in mind and the technical support person connects this to be related to transaction process.

This kind of requests take time for two reasons - they require communication with the customer to make sure that the problem has been defined correctly and they also require work to discover what solutions are possible. At the other end of the spectrum are customers that are familiar with the EDN description and request exact changes to the transaction process. This communication work takes approximately half of the time needed for a change, excluding the time that it takes from the customer to answer inquiries.

One developer described that most problems in communication were due to the fact that "the customers do not know their own process too well" and "they do not know how changes affect". They suggested that visualizing the process could help the users to understand their process better. They noted that some of the customers had been provided with process descriptions previously, and these customers were able to talk with exact terms.

Finding 4: Developers need to know what parameters are required to user interface changes

After a new transaction process has been released to production, the front end developers need to know the required parameters and transitions to make UI changes. Having access to this information is crucial for the front end developers to correctly build their user interface. To be aware of the possible changes, the developers also require information on what version of the transaction process they have in use.

One developer complained that "the actions and their parameters have not been documented too well", even though they are required in almost every change.

Summary:

The developers' needs concentrate on understanding and learning how the transaction process works, learning how to best use it, finding ways to effectively communicate with the operators and being aware of the changes and how they affect the front-end.

4.3 User sketches

As one of the goals was to make the transaction process more understandable and discussable, it is of utmost importance that it is understood how users comprehend the process. To gain further understanding on how the users model the process or how they think about changes in it, four customers were asked to draw their own process to us.

The users consisted of two programmers, one marketplace operator and one marketplace product owner. The task was kept simple. First, it was explained to the users what is meant by the term "transaction process" and what is included in it. After this, they were asked to draw down their own marketplace transaction process. Three of the interviews were conducted locally and one by using video call. The user who was interviewed by using a video call was given the task and had prepared the sketches beforehand.

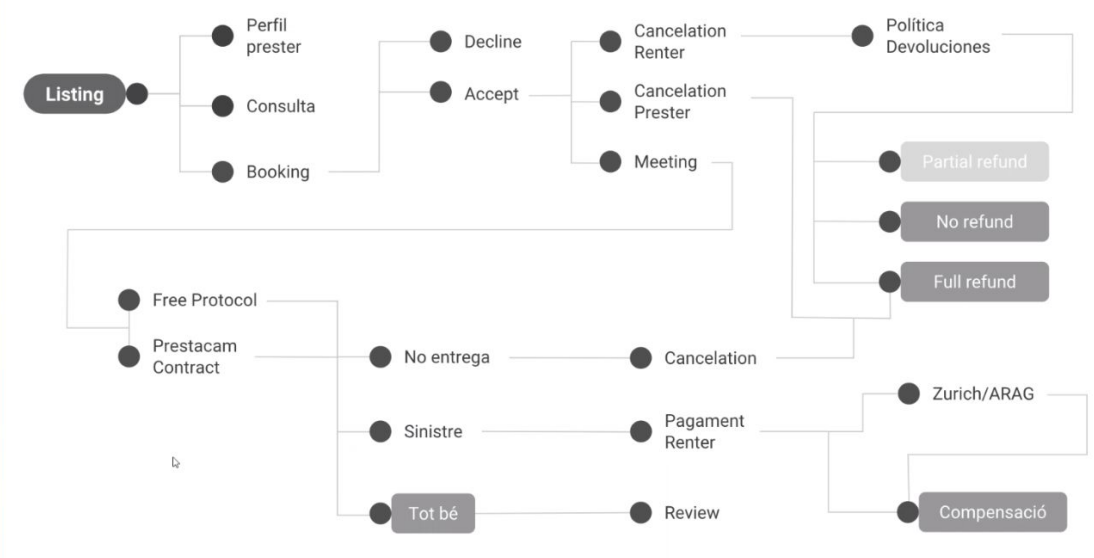

Figure 10. User 1: Marketplace operator.

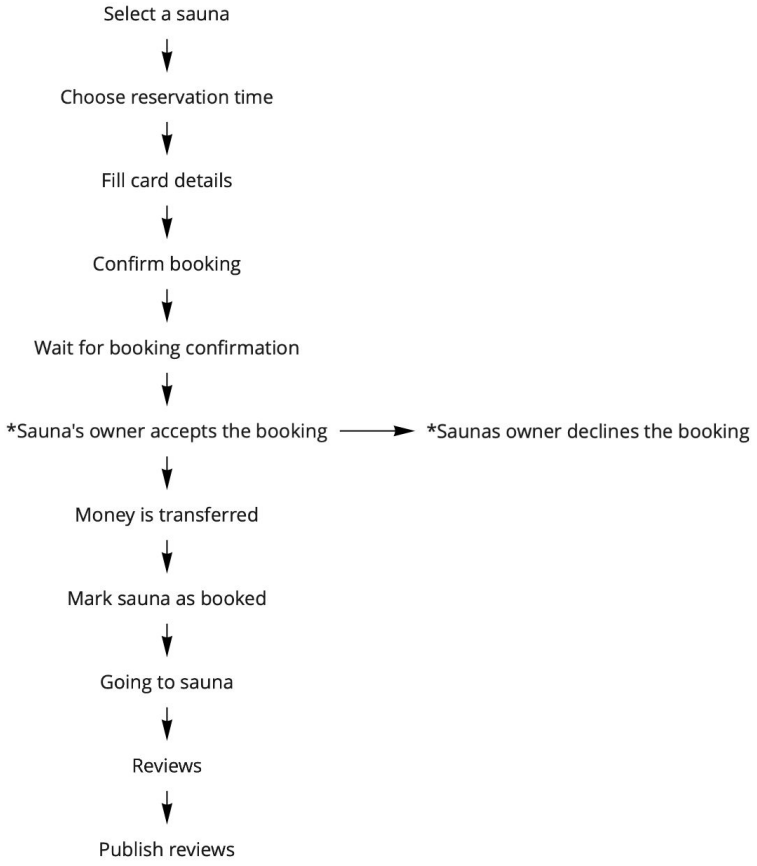

Figure 11. User 2: Developer.

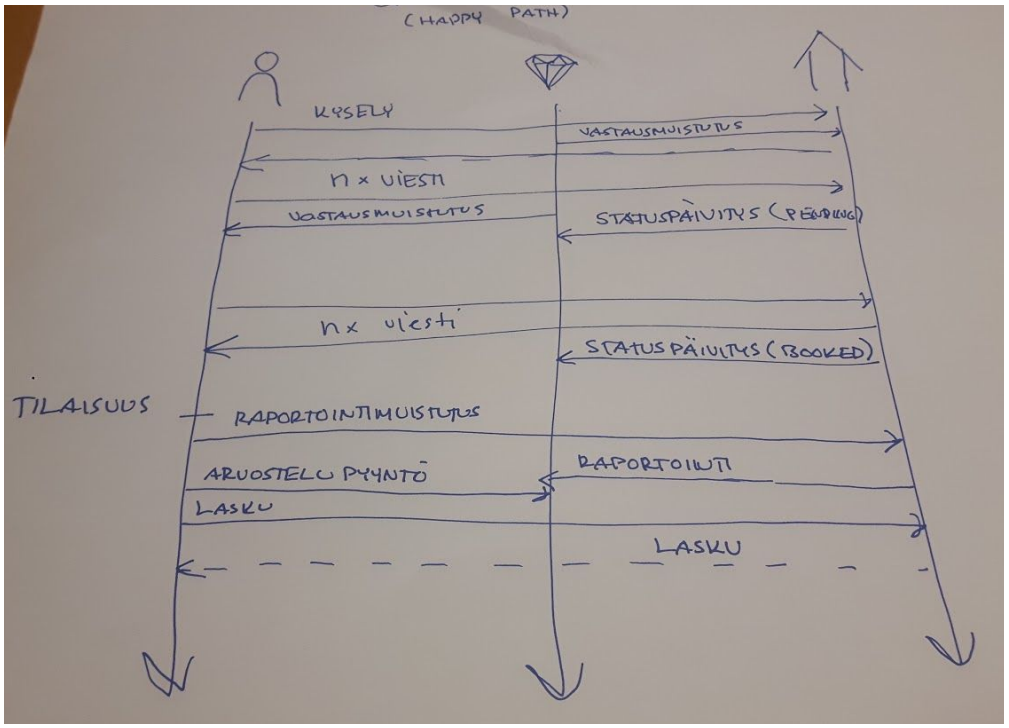

Figure 12. User 3: Marketplace product owner and developer

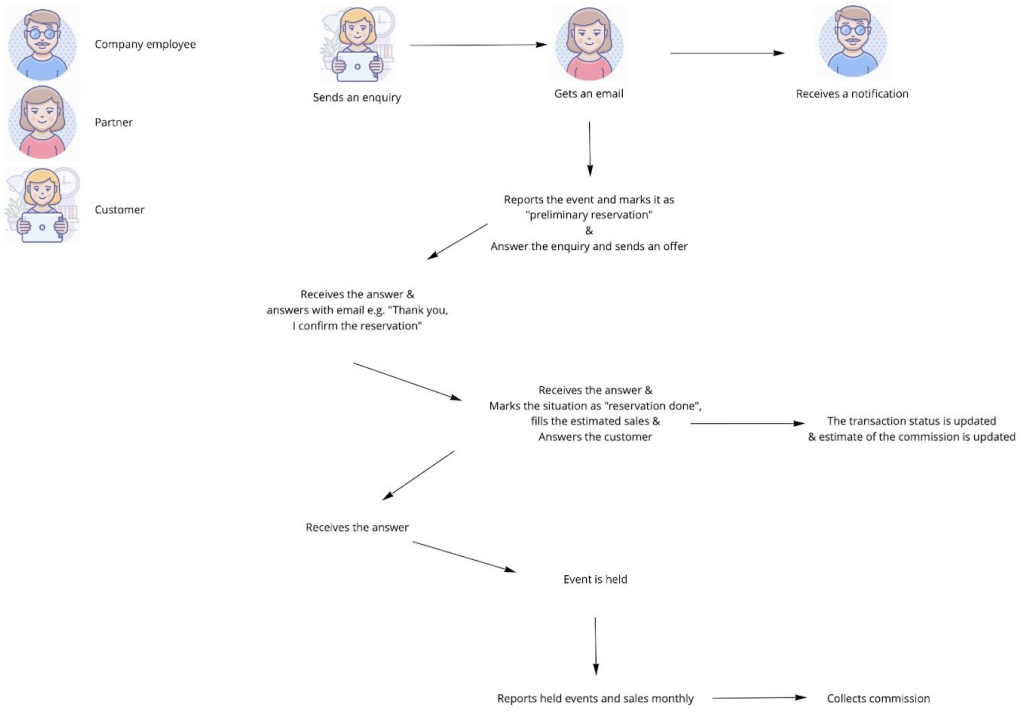

Figure 13. User 4: Marketplace company's CTO.

There are certain similarities in all of the sketches.

1. **Timeline is on the vertical axis**

In three out of four cases, the timeline is depicted to be vertical. The events are listed from up to down in chronological order. Sometimes there are multiple lines for different actors, but the basic timeline composition stays the same.

2. **The users' actions are central**

The users' actions are depicted central in all of the drawings. In all of the cases, the actions are depicted as text describing the action shortly in natural language. The descriptions varied from one to ten words. The mode was two words and medium three. In three of four cases, the actions were depicted as nodes to which and from which the arrows connected.

3. **The actions are connected by arrows**

The actions were connected by arrows or lines in all of the cases. The arrows depicted the flow of the actions and were always directed to one direction. In one case of the four, the user actions were connected to the arrows instead of nodes.

4. **Technical details are not included**

In none of the cases, any technical details were mentioned. The closest mentions were mentions about filling online forms or the money being transferred. even though many of the actions revolved around technical components such as authentication and user interface components to execute actions, they were not mentioned. Everything happened "on the background".

Other observations

Two of the users referred to different user types with icons positioned to the top, and stacked actions to lanes beneath the icons depicting the actor. In the two other cases, the actors were not unequivocally mentioned, but could be deducted from the context or had mentions in the text used in drawings.

In all but one of the cases the sketch included only "the best" route. This means that the drawer had chosen to depict the ideal route for their marketplace and not to include other options, e.g. routes where the seller or provider would not answer the buyer.

As one of the interviews was done on video call, the user had had a possibility to prepare by drawing the sketch beforehand. This particular sketch (Figure 10) includes all of the

possible routes instead of only “the best” route. Therefore, it can be speculated that the amount of detail was connected to the amount of time and preparation the users had had.

The technical implementation was referred to, but no technical details are mentioned. The technical aspects are seen through actions conducted on user interface components (“Accept”; “Answers the message”; “Confirm booking”). There is only one actual referral to how the actions are executed (“answers with email”) and in this particular case, the components used were not related to the marketplace’s user interface. The users seem to expect that information on how the actions are executed or what happens in the background is not needed in the sketch. This might be again due to the limited time or to the fact that the users did not think of the transaction process from technical perspective but rather as “customer journey”.

The most noted interactions are the messages the marketplace users send and changing the transactions status. There are no notions on notifications considering these changes but rather their existence is implied (e.g. “wait for confirmation”; “reminder to answer”). The notifications are expected to be related to actions and not in need of separate mentions.

There are only few mentions of automated events such as “money is transferred” and “reminder to answer” or “reminder to report”. This is probably related to the previous three observations. The users did not include how actions happen but expected everything to work out somehow and concentrated only on the customer journey.

As a final observation, none of the sketches was very complex, but they were all relatively simple, including only three elements: The action, the direction of the action flow and the actor.

The users spend approximately ten minutes on the draft, excluding the user who had done the task beforehand. Three of the users did the sketches by hand and one on computer by Power Point. Drawing the sketches in a short time by hand affects level of detail and absence of illustrations. The task that the users were given was to “draw your transaction process”. The transaction process was described as “all that happens between the seller and the buyer, including but not only booking, conversations, and money transfers. This has definitely affected what has been selected to be put on the sketches and the level of details in them.

4.4 Existing user data

Within Console there is an existing view for editing email templates. This view consists of a listing of the existing templates, a preview function and an HTML editor for editing the templates. The usage of this view has been studied by using Hotjar, a user tracking software that offers heatmaps, visitor recordings and more (Hotjar, 2019).

In the email template editor the user can edit the existing emails by editing their HTML. The company offers an HTML editor in console with basic validation. However, by following the visitor records, it was discovered that most users did not use the HTML editor as expected, but opened the editor and copy-pasted the HTML from somewhere outside the view and only saved it in the editor. The common way to use the editor was to copy some information, e.g. context variables or content text, from the existing templates and pasted this to a file somewhere else. After this, the users pasted the text to email template editor, replacing the previous text.

From this behavior it can be deducted the users were more comfortable in using their own HTML editors and using the template editor only as a tool for submitting the changes. It can be assumed that as professionals, the developers have existing ways of working and are used to work with their own tools.

5. Design

5.1 Job stories for design

Based on the user interviews and interviews within Sharetribe's team, the following job stories were identified. The job stories are in order of importance.

1. **When I am developing my marketplace purchase flow, I want to be aware of what transaction processes I have in use, so I can manage changes in my marketplace.**

This story is bringing forth the basic need of informing the marketplace operator and developers on what transaction processes and versions they have in use. As there is the possibility of having multiple transaction processes, the user should also be aware of what transaction process version and aliases they are using. This helps them to understand possible changes in their marketplace and development processes.

2. **When I'm planning on customizing the transaction process of my marketplace, I want to understand how my transaction process works, so I can understand what happens between marketplace users.**

This story is focused on the need to inform the user how their transaction process works at the moment, what is included in the transaction process, and how it works. This should give the Sharetribe team, operators and developer a common ground on talking about the transaction process in use and help them to understand each other. Having the transaction process description available also helps the operators and developer check if changes have been done correctly.

3. **When I'm designing my transaction process, I want to understand my options and best practices, so I can get an idea of what's possible and what would work for my marketplace.**

This story aims to bring out the problem that marketplace operators and developers are not completely aware of their possibilities when it comes to making changes to their transaction process. As customers are not aware of the different functionalities and possibilities that transaction process engine offers, they might not be aware that customizing the transaction process might solve their problems. Answering this story gives the operators and developers an understanding of how the transaction process can be edited and how to do it.

4. **When I'm implementing a client app, I want to know the parameters of the API call to initiate a transition, so I can make the correct API call from the client.**

This story aims to bring out the more technical need of informing developers on the technical information they need to make changes correctly in the marketplace user interface.

5. **When I want to change the content or appearance of emails sent as part of the transaction process, I want to make edits and see the results immediately, so I can control the quality, outlook, and accuracy of the emails sent.**

This story brings out the need for being able to modify and control the emails that are related to the transaction process.

6. **When I'm developing my marketplace, I want to customize my transaction process, so I can answer to my user needs better.**

This job story is a high level story aimed to answer the need for customizability.

6a. When I'm customizing my transaction process, I want to add all the interaction possibilities and functionalities I've planned, so I can better answer to my users' needs.

This story describes how the users have varying needs on how they wish their marketplace to function. These needs concentrate on their wish to create interaction possibilities to the marketplace customers and providers. The functionalities again include actions such as storing information about the transaction.

6b. When I'm customizing my transaction process, I want to be sure that my process is error-free, so I won't break anything when deploying the changes.

This story describes the user's needs to make sure that the changes they make do not break anything in the existing marketplace.

5.2 Versions for iterative building

5.2.1 Listing processes, versions, and aliases

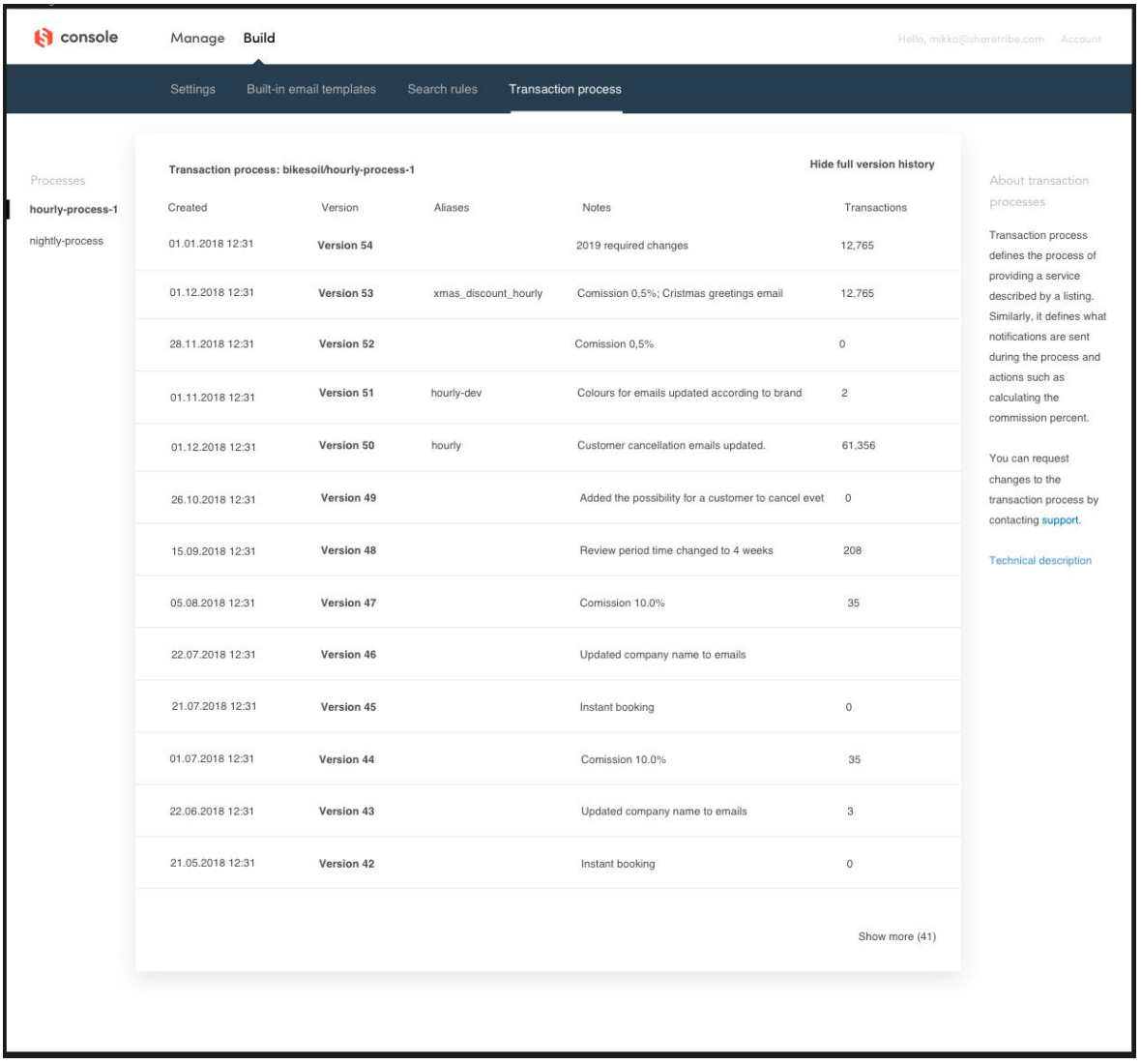

Figure 14. Transaction process listing

The minimum valuable product that would bring additional value to the customers would be one that satisfies their immediate needs. As there is an existing process on updating the transaction processes, the product should concentrate on the problems that are not related to editing the process itself. As the users receive customized transaction processes, they still have to do the front-end customizations by themselves. For successfully managing their marketplace customization and process changes, the users have to be aware of what transaction process they have in use and how to use them. This design seeks to answer the job story 1.

To answer the users' need for information considering their transaction processes and how to use them, a documentation collection was made and published under Sharetribe.com/docs. This documentation consists of a background article and series of how-to guides. The background article sheds light on what is transaction process, what components it includes, and how to best take advantage of it.

A view including transaction process listing with aliases and versions was added to Console. The goal of this design is to give the user the possibility to browse through their processes and their versions. The user is able to see the process name, version and possible aliases with one look. The process date, notes, and process transaction are additional information that give the user a better understanding of when and why the process is taken into use, and how many transactions have been started with this process version.

The information needed to make a distinction between transaction processes are version numbers and aliases. It is common to have one to two different transaction processes, each containing multiple versions. At the time of making this thesis, the number of existing versions was between one to ten, but as the marketplace gains age, it is probable that the number might be in tens or hundreds. For customers, it is important to recognize which versions they have currently in use - in other words, which versions are pointed by an alias. Additionally, it is important to know if the latest version is been pointed to, as this is usually the version that is wanted to be taken into use. The versions are distinguishable from each other by the version number and creation date.

5.2.2 Graphical representation

Motivation and goals for the design

Most of both operators' and developers' needs regarded being informed about the transaction process of their marketplace. These needs were related to understanding how their transaction process works or understanding how the transaction processes work in general. This includes how the transaction process defines what is happening between the marketplace users as well as what actions are taken at each step of the process (Job story 2). Similarly, the users were in need of understanding what is possible to achieve by modifying the transaction process, and what are the best ways to achieve the changes they need (Job story 3).

These problems were approached with two different solutions. One is to offer the users general information of the transaction process and its functions in a written form. The

second one is to offer the users a graphical representation of the transaction processes they have in use.

The background documentation was added to Sharetribe Flex documentation and includes information on the transaction process, its components and how to change it. Additionally, how-to guides were added to give examples of how to use the processes. These include complete examples on how to use the transaction process and, e.g., guides on how to change transaction process setup, how to change the transaction process type, and how to add functionalities (such as custom prizing).

The graphical descriptions were added to Console and they are accessible through the transaction process listing. The view includes a visualization of the selected transaction process version and an information container offering additional information about the selected transition. The graph is designed to inform the users how their transaction process works and from what kind of components it consists of. The information container includes information about the actors, notifications, delay transitions, actions, and parameters. This will offer the user additional information on how their process works and also information needed in API calls (Job story 4).

Additionally, the design helps Sharetribe's support and sales. The support has easy access to see how the client's transaction process looks like and how it works, which offers valuable information useful in support tasks. The sales team has the possibility to show the client or a possible client how the transaction process looks like and use the visual representation to explain the components used in the process. This helps the sales team to communicate the value of composable transaction flow.

The drop-down navigation enables the users to quickly navigate between different process versions, both customers and Sharetribe's support are able to see differences between the graphs. This makes it easier to recognize changes in the process, which helps on discovering possible errors or helps support to track possible changes in the process.

Design

The graph is designed to convey information at a glance. It emphasizes the connections between the different states and relations between states and transitions, something that the EDN description clearly lacks in communication. As the relations are clearly stated, following the graph and flow of actions doesn't need any particular skills or knowledge.

As EDN is more technical oriented description, the graph is meant also for more business and process oriented viewer. However, the graph corresponds to the data model to avoid confusion in the development process. The terminology and graph elements are in line

with the EDN, even though other terminology might have had more natural feel to it, and could have helped faster recognition for people who are not familiar with process modeling. This helps the developer and operator to reach common terminology and mental model of how the process works. It will also help the Sharetribe's support to recognize possible changes and errors in the customer's processes.

The graph is designed to decrease the time that is used to comprehend the current process structure. The user can easily see what actions the marketplace user can take in each state and what are the overall possibilities for a transaction to take. This helps with discussions and planning the process as the users do not have to keep in mind the process steps, but can use their cognitive capacity on creating new. It is easier for a developer experienced with transaction process descriptions to see possible modifications as they affect the graph form

Being able to visually try out possibilities also decreases the possibility of mistakes. From a graphical description, it is easy to recognize if a transition doesn't have a following state and therefore an arrow ends in empty space. It is also simple to notice if there is no arrow leading to a state. This will help in the development phase to plan the process and to notice mistakes.

Functionality

To help the user to acquire additional information about the process, the user is able to click transitions, causing information to appear to the right side container. In this way, the user is able to discover relevant information about that exact transition in compact form. This information includes the actor for the transition, the actions, required and optional parameters, notifications, and delay transitions. This information (apart from actors) is not included to the graph itself for the sake of clarity, but it is important for the user to be able to understand the functionality of the process. The data in the right side container is divided under subtitles. This reduces the cognitive work as it is easier to memorize where certain parameters can be found and reduces the need to look for needed information.

The data presented on the right side is also essential information for implementing possible UI modifications correctly. As the relevant information for a single transaction is put into one place, the developer no longer has to scroll through the whole EDN to find the relations between transitions and states or corresponding action parameters. This makes working with the EDN faster and also less wearing. Showing the transition details in the container supports discovery, insights, and creation especially when the graph is used while making changes to the EDN. The EDN groups different factors in the transaction process description according to type - by status, transition or notification. However, when planning the process, is more natural to think about the process according to its flow. In

this case, the user plans the next transition and what actions happen at this transition. For example, when the user plans that the next step will be canceling the transaction, they also decide what notifications are needed at this state and what actions must happen so that the cancellation is complete. In the same way, verifying that all is as it should is easier when the information is collected according to transition, not the type.

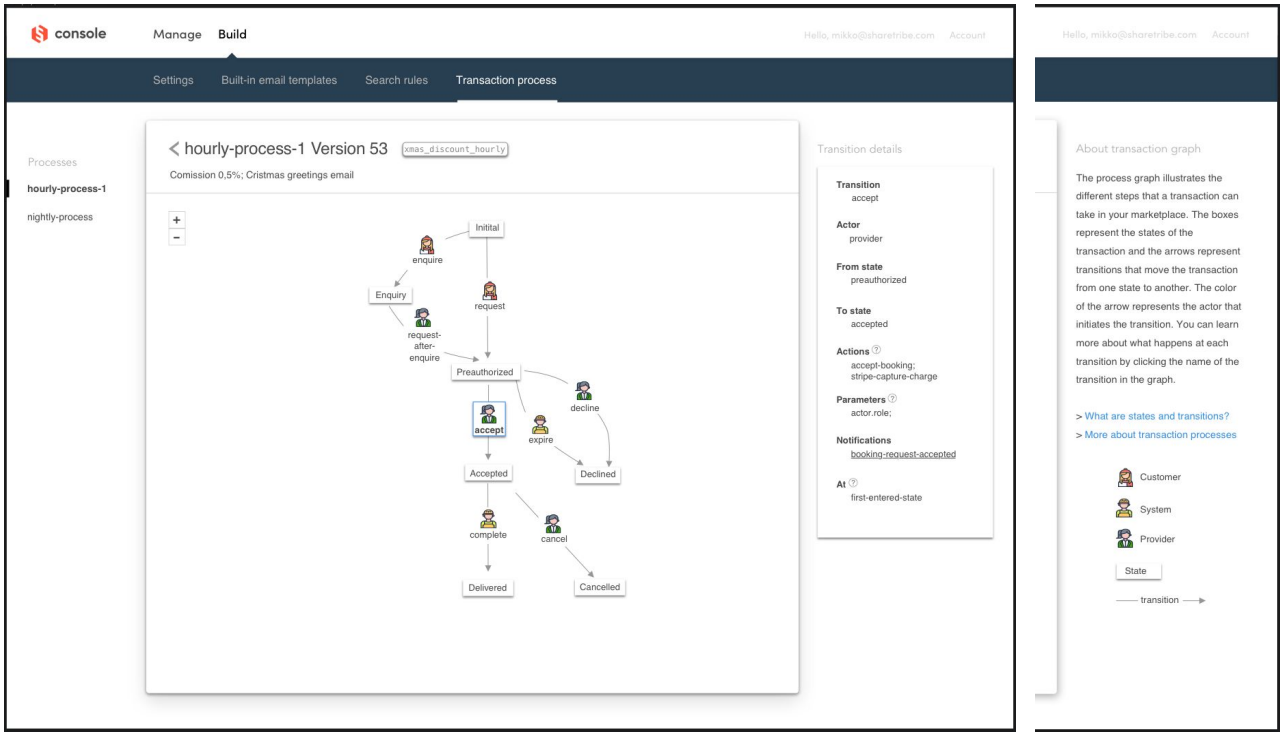

Figure 15. Transaction process graph and details

The graph

The transaction process is described as a directed graph in the data model. This means that there are certain limitations on how customers can build their transaction process. However, it is technically possible that the graph is of any width and length the customer has build the process to be. It is more likely that the graph gains more length (more steps in the process) than width (more alternative options in the process) as for complex alternatives it might make more sense to create another transaction process. The page has been created so that it can continue freely vertically but is divided into three columns horizontally. The graph itself can be panned and zoomed in to. By default, the graph is fitted to the reserved space.

As the user research showed, most of the customers were comfortable with “boxes and arrows” kind of description when it comes to process visualization. However, the current visual process description was described as intimidating. Therefore it is possible that in this case, the experience of being intimidating is due to the style, not the elements. The user research revealed that the important factors in the visualization are.

1. Timeline is on vertical axis
2. The users’ actions are central
3. The actions are connected by arrows
4. Technical details aren’t included

It is plain that the visualization should not include excess elements but only those needed to clarify the process. Therefore, the graph is designed so that the most important information is pushed to the front and emphasized. This information includes the transitions, clearly stated user actions and the flow of actions. To prevent information overload and to keep the visualization uncluttered, the technical information is presented only in the right side container

Apart from the user research, another motivation for using a graph as a visualization is the easiness of implementation. The graph is easily made from the EDN description and corresponds it completely. In the directed graph notation, transitions are depicted as arrows - mathematically called as edges - between states. This is how most libraries used for rendering directed graphs function. The natural way for people to think user actions in not to think for them as arrows, but as points where arrows leave from. This causes a conflict between two models of thinking.

During the design process, it was also considered if the statuses should be removed. This would have been technically very difficult, as most libraries that are used to render graphs are tied to nodes and edges and changing this would require a great amount of work. As there was no clear preference between the interviewed customers between the models with statuses and model without statuses, it was decided to include both states and transitions (nodes and edges) to the visualization.

The details are situated on the right side of the graph on the same level. In this way, the user can see the details at the same time with the graph. The container is sticking to the top so it is visible even when the page is scrolled down.

Help on the view is offered on the right side. This includes information about the graph and its legend. There are links to background information about transaction processes and

Transaction API references. This help will be covered by transition information container when a transition is selected. The transaction information container includes help texts for actions, parameters and delay transitions. These helps are accessible by hovering over the help icons next to labels.

5.2.3 Editing the EDN in command line interface

Motivation for the solution

At the starting time of writing this thesis, all modifications to transaction processes were done by Sharetribe Support team. If a customer wanted to change something in their transaction process, they contacted Sharetribe Support, requested a change and waited delivery. To offer the users a possibility to edit the transaction process, two options were considered: a visual editor and offering the developers the possibility edit the EDN directly.

In section Existing user data we considered the data that was acquired from previous development phases. In this research, it was noticed that even though the company offered an editor for editing email templates, the users preferred to use their own tooling and the editor remained mostly unused. This was most likely due to the fact that developers have their own tooling that they are accustomed to use and prefer to use for development purposes. Usually these editors include a set of features specific for development work, and building those tools to Console would not make sense. From this observation it was deducted that offering an editor for EDN in Console would meet a similar destiny.

The decision to let the users use their own tooling for editing the EDN was backed up with the notably lower amount of resources that would be used compared to creating an editor with graphical WIMP interface. As Sharetribe is, as a company, in a state in which the amount of features is highly important for the functionality of the product, opting for the fast and easily made solution is a sensible choice. Directly editing the EDN is the least restricting solution for enabling process changes, giving the users freedom to make changes as they wish.

In user interviews, the developers expressed needs related to learning how the transaction process works and communicating with the operators about the desired changes. When editing the transaction process was in question, the most pressing problems were the lack of validation process and lack of information that prevented them to get started or made starting feel difficult. The editing itself was not considered difficult, but rather making sure that the editing process was done correctly.

Keeping the editing process separate from the Console also prevents the possibility of people without required understanding trying to make modifications. As making changes to the transaction process always goes hand in hand with the need for doing changes to the UI, development resources are needed almost in all cases when making changes. This supports the decision of limiting editing to developers only.

The interviewed operators agreed that they would rarely have the need to make changes to the transaction process after the process had been modified for the marketplace's needs. This also supports the decision that the marketplace operators don't have to have a way to modify the transaction process without the help of a developer.

The solution background

Editing the transaction process in a local command line tool offers a solution to the job stories 5 and 6, the need for customizability of the transaction process and its notifications. The job story 5 includes customer's need to be able to modify and test transaction related notifications. The job story 6 identifies the customer's need to modify the transaction process of their marketplace to meet their customer's needs. This includes being able to add functionalities and interactions to meet the needs, and being able to implement them correctly.

The process should also support stories 2, 3 and 4, the needs for information sharing and learning to work efficiently. The job stories 2, 3 and 4 identify the needs for understanding how the transaction process works and how it defines the action possibilities in the marketplace. These jobs concentrate on the user's ability to understand the process and its components, and understanding the possibilities it offers.

The targeted customers are either professional developers or teams including professional developers. In creating a solution for these users, the main concern is to use tools that they feel comfortable to use in this context. Developers have various working habits commonly used within to the field. There are tools, ways of receiving and distributing information, and ways of learning that are generally adapted within the profession. To create a tool useful and adaptable for these users, it should resemble the existing working habits.

Solution description

Developing Flex happens on local environment. The developer is free to use any development tool or editor they wish. The templates and updates are pulled to their local development environment and the changes made are pushed back to Flex backend. Using a command line tool for editing transaction processes naturally fits this sequence of actions while the user is able to use their own development tools. The command line tool adds a

supportive element to the use flow. It is common for developers to run programs they use or have built and execute commands directly from command line.

Not only does a command line tool resemble the existing working habits, it offers a highly effective and flexible way of working. The command line is an effective tool, enabling the user to run programs, do modifications and navigate using only short text commands. The developers are able to use their own tools and methods of working with the assistance of the tool. It brings no limitations or boundaries to what one is able to do. This is highly important when it comes to the value proposition of Flex as a product: flexibility is the core of the product. Almost any graphical tool would come into its way or become highly complex in trying to offer flexibility.

The editing flow will resemble the editing flow that the company technical support uses at the moment. The editing process in command line tool will include a verification process to prevent and expose errors. Even though the EDN was not felt to be too difficult to understand, a plan was made to make it simpler to better suit the skills of an junior level developers. This included changes to the process description itself and better documentation.

Design principles

The goal of developing the command line tool for editing transaction processes is efficient, flexible and error-free customization of the processes. To reach these, in this thesis the same usability principles as Nielsen (1993) defines in his book *Usability Engineering are used*. These usability principles are learnability, efficiency, memorability, low error rate and user satisfaction.

The learning experience in enhanced by providing the users examples of transaction processes that can be viewed both in EDN and graphical form, offering help in the “help” section as well as in documentation, and supporting learning by doing. Learning by doing is supported by giving instant feedback in the form of graphical process descriptions, rendered notification templates, and process simulations.

The defining feature of command line tools are their efficiency. This tool offers vast flexibility to the developers by giving them free hands on customizing the process (within the scope of used technology). The users are free to modify the processes, their aliases and push and pull them freely. As there is no graphical interface, the commands are in text form, and even complex tasks are possible in just a few rows.

The memorability aspect is addressed by using commands close to natural language and having the helper function readily available. The process simulation, graphs and ability to

compare processes to each other help lowering the error rate. This is furthermore supported by process validation command, that validates the process.

Altogether these aspects offer the users efficient, flexible and error-free process customization experience that will result in user satisfaction.

Solution functionality

The CLI tool is installed by using npm package manager. The tool allows the user to pull a remote process to a local disk and push process from local disk to remote. Additionally, it gives tools for listing and examining existing processes, modifying them and creating new ones. Also creating, updating and handling aliases is possible. While modifying the processes, the user is able to explore them by comparing one process to another, to print a process graph, simulate transitions and transactions and validate processes. The use flow can be divided into four different phases: Learn, explore, modify and validate. Ideally the first time user goes through all the phases, and a recurring user can start directly from the part they need to.

In the learning phase the user is given the opportunity to learn how the transaction process is constructed and how it functions. This is done first and foremost by informing the user about their possibilities by offering the information and examples. The command line tool is supported by extensive documentation, including an architectural decisions document, how-to-guides, examples on how to customize the transaction processes and examples of different types of transaction processes the user can compare and try out.

The motivation for offering a selection of ready made templates for transaction process is to give the user an understanding of the possibilities that the process offers and also show them the best ways to implement these functionalities. The user can open a selection of transaction process templates, view them in EDN and graphical form, and this way discover insights on how to built and implement transaction processes. By using these examples as a basis for their own work, the user can be more confident about the quality and lack of errors in the process they make, as the example templates have been created and tested by the Sharetribe team.

The command line tool itself offers the user the opportunity to dive into the selected transaction process and get further information on the transitions, notifications, and actions it includes. The tool offers the user a listing of the process transitions, states, and notifications. By selecting a specific transition, state or notification the user is able to view more information on it. From transitions, the tool lists the previous and following states, notifications, delay notifications, required parameters, optional parameters, and actions.

Notifications open to a new window and the user is able to see the HTML code or render the email contents.

In the exploration phase the user can dive further in the transaction process and experiment with it. This phase includes the possibility to make changes to the transaction process and see how they affect the process functionality and outcomes. The user can open, view, change and simulate any transaction process they have in use or any of the examples. In case they wish to make changes to their own, they can choose to base the changes on the templates given. This way user can safely get feedback on the changes they have made and see how they affect the transaction. The user can also open the transaction graph to another window, which will be automatically updated when changes are run.

The transaction process simulation can be used for evaluating the outcomes and changes of the customized transaction process. The user can select any process to make changes to it, and see how they affect the process outcome. As there are several paths the process can take, the user must select the transitions they wish to be included into the simulation. The simulation runs through the code and prints the process outcome. This includes the transitions run, money transfers, commission and sent notifications and their delays. For example, if the user wants to see how will a certain email, e.g. receipt, look like, they can simulate the transaction until this specific transaction has been completed. Without simulation, it would be difficult to generate a receipt with correct information in it.

To see a visual representation of the process, the user is able to open the transaction graph to a new window. The process looks and functions similarly to the one shown in Console. The graph will automatically update to correspond the code anytime the code is changed. Being able to see the transaction graph helps to spot disconnected nodes (states) and to comprehend the transaction process as a whole.

The modification and validation phases work parallel. In the modification phase the user will make the desired changes to the transaction process. In validation phase the changes are tested for any syntax or logical errors. Part of this is done automatically and part of it lies on the user’s responsibility.

Any syntax errors are validated automatically. The CLI tool offers tools for inspecting the changes that have been made and their validity. These include seeing the transaction graph that is generated from the code as well as being able to compare the transaction process to the previous version. Being able to compare the transaction process to its previous version gives the possibility to quickly see the changes that were made. Also simulating the transaction process outcomes is a way to offer feedback and aid learning.

Simulating the process is a way to check if all the parameters come through correctly or if the process is using parameters that were not given on the route. For example, a state can be accessed through more than one transition, it is possible that some actions might try to use parameters that were given in one of the leading transition but not in the other. This also helps to see if the notifications come on right time, if the delays are in place and if there are any other logical mistakes.

Notifications related to transaction processes can also be edited by using the command line tool. The wanted notification file is pulled to the local environment, edited and reviewed. The user can view the changes locally and render the email. The user can also simulate the transaction leading to sending the notification. This shows the users if correct parameters have been passed.

5.2.4 Choosing the initial transaction process

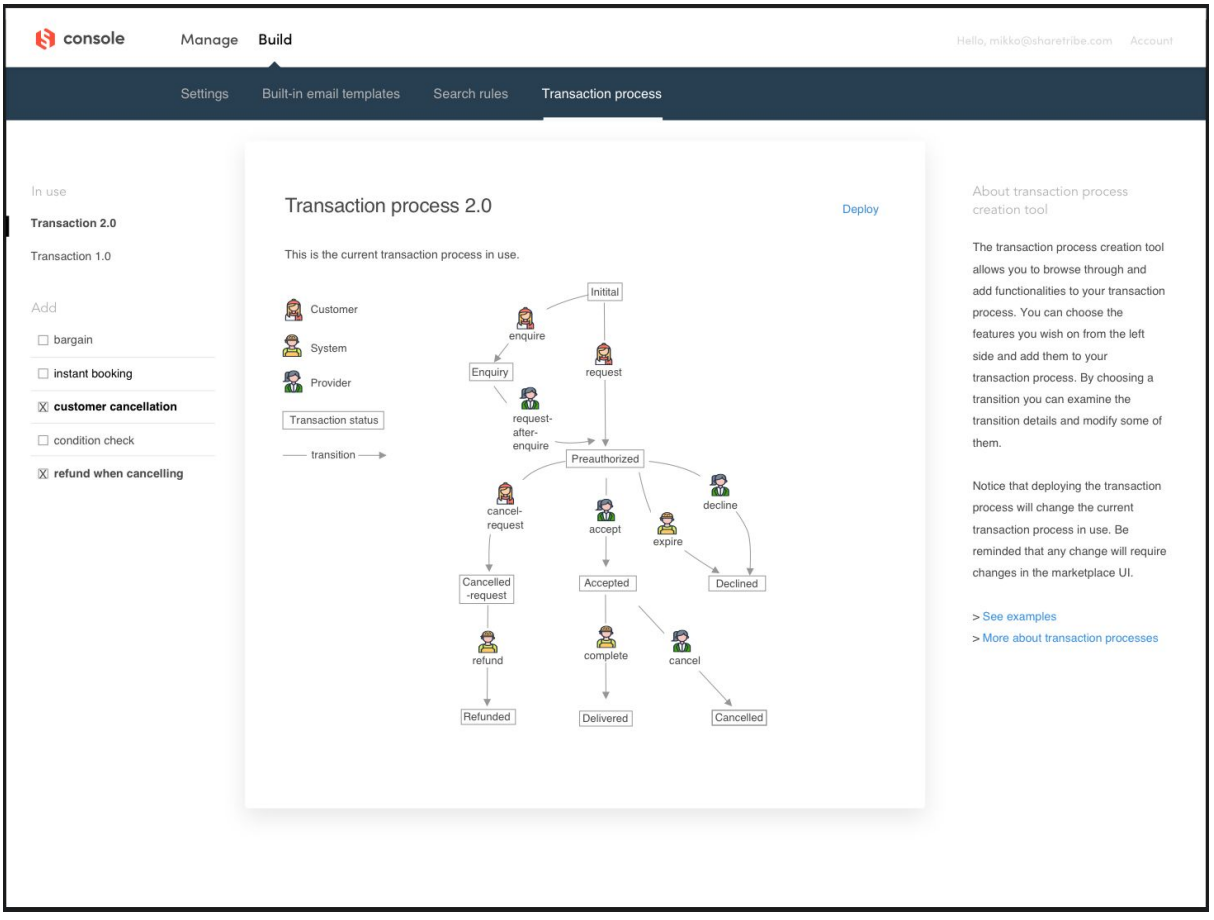

Figure 16. Adding functionalities to the transaction process

The user requirements for different marketplaces vary, but it can be estimated that along time certain patterns in these requirements will appear. Customers who want to rent sports gear, bikes or cameras might have similar needs when it comes to checking the gear condition. The customers who want to book beauty services or photoshoots might have similar needs when it comes to customer cancellation. These similar needs will emerge over time and amongst them it is possible to find patterns that can be used when modifying the transaction process. The emerging of similar marketplace requirements was already visible in the user interviews, where the customer's wishes resembled each other.

The hypothesis behind this design solution is that a significant portion of the marketplace requirements can be satisfied by offering readily made options or configurable "building blocks" to the initial transaction process.

Offering an easy way to customize the transaction process, even if not to great extent but only by enabling or disabling few basic functionalities, would help those customers who have limited access to development resources or do not have a full-stack developer. As the transaction process can feel difficult to approach and understand as a whole, offering an easy starting point can inspire operators and developers to make changes that they otherwise would not consider due to the workload.

Offering a way to configure the transaction process functionalities would help these customers to get started with their marketplace and learn what are the exact changes they would need to have their transaction process. Learning the exact needs of one's customers is a process that needs time. Therefore, it's more likely that the more advanced and detailed changes would happen later in the marketplace's lifecycle, rather than at the very beginning while establishing the business. This tool would also help the developers in the initial phase of building the marketplace to add features quickly without worrying about making mistakes in the process.

Solution

This design seeks to answer job story 6 about enabling the users to customize their transaction process.

In this design, the user is offered add-on features to their transaction process. These additional features and functionalities have been noticed to be commonly requested amongst customers. These additional features are adding an instant booking option, bargaining option, adding customer cancellation possibility, adding a refund possibility, and adding condition check for the product. These are requests that have commonly risen in customer interviews and sales calls. In the future, Sharetribe continues to collect user's

wishes about the transaction process functionalities, and new additional features can be added according to this data.

All users have a transaction process that their marketplace uses and the process is visible in the transaction process listing in Console. Below this, the user is offered the possibility to "Add new" transaction process.

The user is directed to a view with a graph of the Sharetribe initial transaction process. The page includes a right side feature listing, a graph in centre and on left side information about actions and parameters and a help section.

From the right side help section the user is directed to documentation with a collection of transaction process templates and descriptions on how they work and in what kind of projects they are useful. This documentation helps the user to understand the full potential of the transaction processes and how to use them.

From the lefts side user can choose the features they wish to add to their transaction process. Every feature includes configurations for time delays, notifications, and actions. It is noteworthy, that all of these configurations are created by Sharetribe team beforehand, and the user is only offered a limited amount of customizability in them. In this way, a new branch will be added to the transaction tree. Examples of modifiable configurations are the length of the time delay and commission percentage

This functionality is only used in the beginning of marketplace creation, and is not used after the initial transaction process model has been modified. The main motivation is to serve customers in starting their marketplace by offering them a low-cost way to customize their marketplace's transaction process and to develop their marketplace in the lean way.

Most of transaction process changes need some elements to be added in the UI to function properly. This means that even though the transaction process is changed in the Console and can be done without development skills, changing the UI still requires development work. However, this tool will shorten the development time and remove the need for a developer with a more advanced skill set. Later on when the customer wants to continue developing their marketplace and they have better understood their user's needs, they can hire a development partner to make more exact changes to their transaction tree.

6. Evaluation

The evaluation will be done by reflecting on how well the offered solutions will solve the research question:

What kind of tool for customizing transaction process offers additional value to operators?

The additional value is defined through the needs of the users - in this case, the operators and developers. Their needs were researched and put into the form of job stories. Solving these job stories and answering to the users' needs in an effective way will bring additional value to the operators. Based on these job stories and to the understanding of the users gained through user stories, incremental solutions to their needs were introduced in this thesis. This evaluation will reflect on how well the designed solutions will be able to satisfy the users' needs and bring the desired additional value.

Out of the designed solutions, listing of the transaction processes and aliases, and graphical presentation of transaction processes were finished at the time of writing this evaluation. These solutions answered to the job stories 1 to 4, and were focused on informing the user on how the transaction processes work and how their own transaction processes work or could work. These solutions are evaluated by reflecting how the offered solutions have affected users behavior during the last seven months.

The job stories 5 and 6 concentrated on being able to customize the transaction process notifications and the transaction process itself. The proposed solutions for these stories - editing the EDN in command line interface, customizing transaction process notifications and adding functionalities to the transaction process - were not finished at the time of writing this evaluation. Therefore, in this evaluation, it is reflected how successfully the transaction processes that users have built during the last seven months could have been built with these tools. This evaluation gives as an indication of how well these solutions could answer the users needs.

6.1 Informing customers about their transaction processes

In this evaluation, the focus is to understand if the changes made have helped the users to understand what processes they have in use, how their transaction process works, how transaction processes work in general, and what parameters are needed for API calls.

All the job stories are related to gathering information about the transaction process. They focus on the customers' needs to gain information about the process. Before the implementation of transaction process listing and graph, the only way user could get information about their transaction process was through Sharetribe customer support or the technical documentation accessible in GitHub, which only contained a limited amount of information.

Evaluating if the customers' need to understand the process has now been better addressed by the solution proposed in this thesis, the quantity and quality of support questions related to transaction process was observed. The premise is that the implemented solution will affect the amount of transaction process related questions and their quality.

The Sharetribe support works exclusively through a ticketing system, which collects emails and instant messages into topics. These topics can be sorted by date, counted and their contents evaluated. It is possible that the implemented solution will decrease the amount of support questions related to transaction processes as the new features will inform the users better. Another opportunity is that they will increase, but the content will change in quality. While people become more aware of the transaction process and the possibilities it offers, they will also get inspired on changing it and acquiring more knowledge. Therefore, it is not only enough to study the quantity of the questions, but also the quality.

The new features seek to answer the basic questions considering transaction process (such as "what transaction process I have in use", "where does this alias point to" or questions relating to the possibility to implement functionalities such as booking cancellation). Therefore, it can be assumed that after they have been published, the content of the support questions shifts to more precise or more personal questions.

Hypothesis: The amount of customer support tickets requesting for information related to transactions process will decrease, and the quality will sift from basic questions to more precise and development oriented questions.

Other aspect that affects the quality and quantity of the tickets is the amount of new customer Sharetribe gains and new features published. The amount of new customers grows, and as the customer base is rather small, the amount of work that one new customer causes has a significant impact on the overall amount of work and tickets. Also, some new features require transaction process changes to work, which might cause an increasing amount of transaction process related tickets.

Table 4. Updates with an effect on transaction processes published during the observation period. The solutions designed in this these are bolded.

Date	Feature
February 11th	**Transaction process listing and transaction process documentation**
March 21st	Custom prizing engine
March 26th	**Transaction process visualizer**
April 17th	Time-based availability management
July 3rd	Strong customer authentication

Table 5. Total amount of tickets and transaction process related tickets

Month	Tickets*	Technical Flex tickets** (change from previous month %)	Transaction process related tickets*** (change from previous month %)	Transaction process tickets from technical tickets
January	1,944	56	32	57,1%
February	1,879	63 (12,5%)	17 (-46,87%)	27,0%
March	2,298	56 (-11,11%)	36 (111,76%)	64,3%
April	2,035	85 (51,79%)	22 (-38,89%)	25,9%
May	1,950	87 (2,35%)	26 (18,18%)	41,1%
June	1,556	99 (13,79%)	32 (23,08%)	32,3%
July	1,802	112 (13,13%)	39 (21,88%)	34,8%

*All tickets include Go-related tickets

**All tickets that have been appointed to Flex technical support

*** Tickets that are related to transaction process, either requesting for information or modifications related to transaction processes

Table 6. Transaction process support tickets and amount of customers

Month	Transaction process related tickets	Live customers	Users in development phase	All active customers*
January	32	10	16	26
February	17	12	17	29
March	36	14	17	31
April	22	19	23	42
May	36	23	25	48
June	32	27	32	59
July	39	28	37	65

* Active customers means all customers that have had development activity during the last two weeks or have live sites

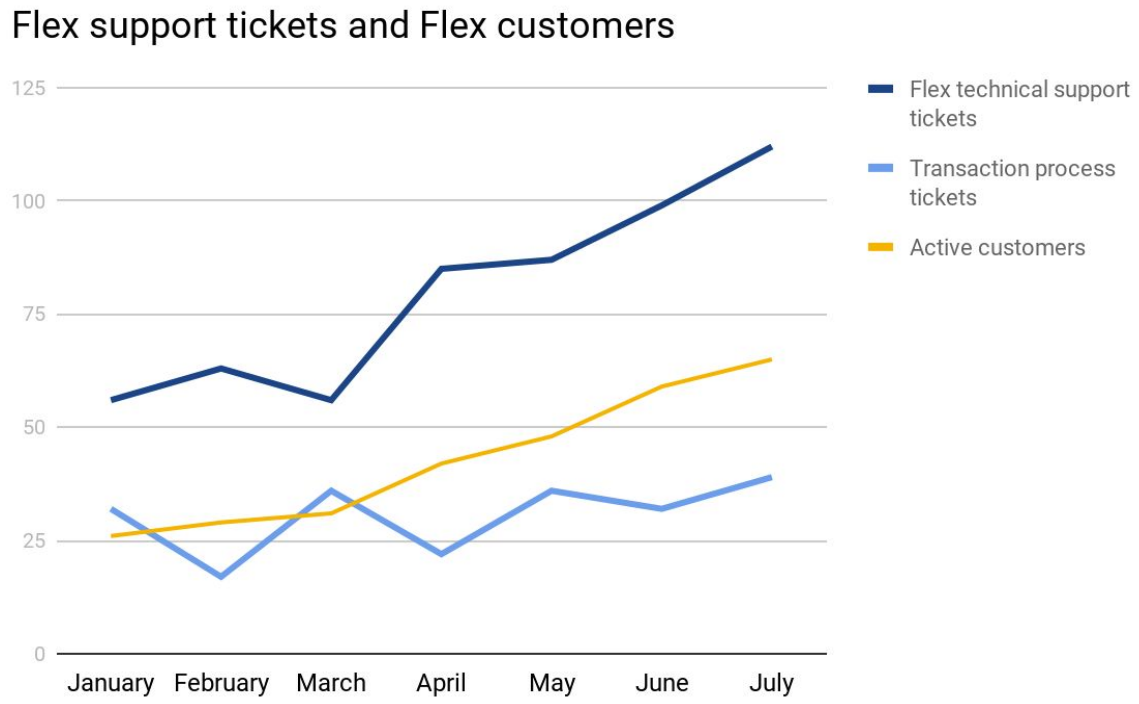

Figure 17. Flex technical support tickets and Flex customers

The Figure 17 shows that during the first seven months of 2019, the number of Flex customers and Flex technical support tickets have increased steadily in correlation with each other. It can be expected that the amount of work continues to rise with the amount of customers. However, The Table 7. shows that the amount of technical support tickets per active customer dropped steadily from 1,2 to 0,6 technical support tickets per customer. This means that customers start less conversations with the support team.

Table 5 shows that the number of technical support tickets have doubled since the beginning of 2019, while the number of transaction process related tickets has not changed significantly during this period. The transaction process related tickets include requests for information about the transaction process or any requests that require changes in transaction processes. This might be because people are requesting less for changes, they are requesting less for information or they have more precise requests. The latter two might mean that the customers are better informed about how they can change their process and are able to offer more precise information in, e.g. graphical form.

The amount of process change requests fluctuates month to month, being higher every other month. This probably follows the natural rhythm of customers making changes and has evened out during the second quarter of the year, likely due to growing number of customers

Table 7. Amount of individual customers asking for support

	Transaction process related tickets	**All active customers**	**Transaction process tickets per all customers**	**Individual customers asking for support**	**Transaction process tickets per individual customers**
January	32	26	1,2	15*	2,1
February	17	29	0,6	16	1,1
March	36	31	1,2	12	3,0
April	22	42	0,5	15	1,5
May	36	48	0,8	16	2,3
June	32	59	0,5	18	1,8
July	39	65	0,6	24	1,6

* The amount of individual customers is greater than amount of active customers as these exceeding customers were not considered active as they didn’t have API activity

Table 7 shows that the amount of technical tickets per developing customer has decreased while the amount of transaction process related tickets per developing customer has stayed relatively the same. While the amount of active customers has increased steadily per month, the amount of individual customers asking for support per month has only slightly increased.

This is most probably due to the fact that transaction process changes are something that are executed during the development phase and not regularly customized. The data shows that most of the individual customers request transaction process changes during a period of one to two months and the stop. Therefore the customer group requesting changes renews periodically.

However, the amount of individual customers asking for transaction process changes should still increase in relation with the amount of active customers. Even though there is a slight increase, it could be assumed that the amount should be higher. These results lead as to the same possible scenarios as previously. It might be possible that new customers don't request transaction process changes, or they do not request for help from support. The former is supported by later findings that show that several new customers don't make changes to their transaction process. To verify these hypothesis, the support tickets per request type are examined.

Table 8. Flex transaction process requests by type

Ticket type	January	February	March	April	May	June	July
Requesting for information	16	12	11	5	8	3	15
Commission % change	0	3	2	1	4	0	1
Email template change	1	1	2	7	4	9	7
Add protected data	6	1	1	5	3	2	1
Change process model	3	4	3	1	3	3	3
Add a feature to the process	0	2	1	5	3	1	8
Edit process	7	13	11	6	8	8	10

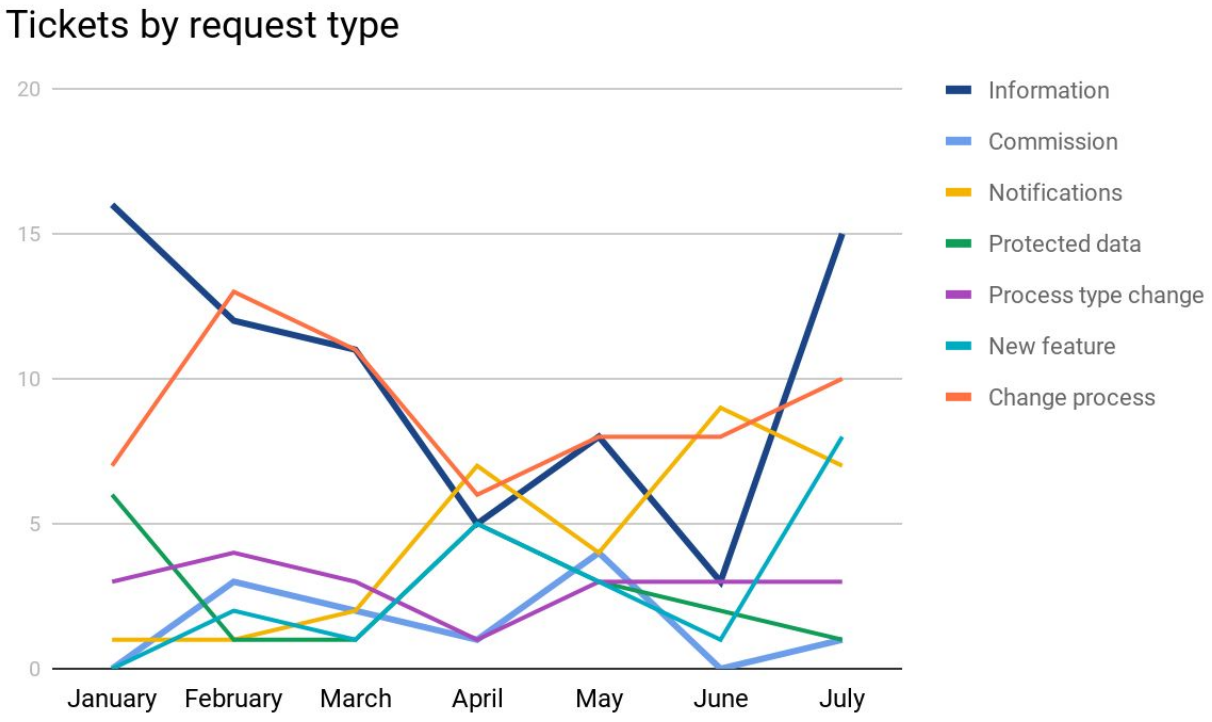

Figure 18. Flex transaction process requests by type

Table 8 and Figure 18 show that the amount of tickets asking for information about transaction process has decreased notably. This gives strong support on that the work done on informing customers on transaction processes has worked. This is the likely reason for why the amount of transaction process related tickets has not increased as the amount of technical support tickets has. The transaction process and version listing was published in early February, and the graphical representation was published in late March. In the graph it is possible to see a decline of tickets requesting for information from January until April, where there is a small increase and again decrease in ticket numbers. There is a spike in information and feature updates in July. This is because the strong customer authentication was taken into use, causing mandatory process changes for many European customers.

In Figure 18 is visible that the requests concerning process type changes, commission changes and updates on protected data have stayed low and steady during the research period. Request for changing transaction process notifications have increased. This might be due to the fact that customizing the notifications according to branding is a basic change that many wish to do. The process change requests have peaked in February to March and then decreased slightly, slightly increasing from April forward. This indicates that either the new customers have not asked for process changes or their request are more exact and better informed.

To be able to find out if the customers have asked better informed questions, the qualitative data from the support tickets is examined. The customer tickets considering transaction process changes send during the period January to July. The evaluation was done monthly, and the tickets were evaluated against the following question: “How well the user is aware of the basic functionalities of the transaction process?” The results were divided into three categories: Poor (equivalent of value 1), satisfactory (equivalent of 2), excellent (equivalent of 3).

Tickets that were evaluated as “poor” showed the lack of using correct terms when speaking about the transaction process. The question showed that the user wasn’t properly aware how transaction processes works and how their request could be achieved.

Tickets that were evaluated as “satisfactory” showed basic understanding of the terms and functionality of the transaction process. The customers had a basic understanding of what they were wanting to achieve and how, even though they couldn't give technical details.

Tickets that were evaluated as “excellent” showed that the customer had a good understanding of how the transaction process works and what is possible with it. The customer used correct terms and visual aids in their communication, and requests were precise.

Table 9. Evaluation of users’ understanding of transaction processes

Month	**Average result**	**Characteristics**
January	1,6	Asking for aliases, parameters and what is possible
February	1,5	Basic questions are “how can I do this?”
March	2,2	Customers start to use correct terms.
April	2,5	Many people use graphs and correct terms in their communication.
May	2,5	Most people use graphs and correct terms in their communication.
June	2,7	Most people use graphs and correct terms in their communication.
July	2,9	All but simple process change requests were communicated with a graph. Many problem shooting tickets appeared, were users asked for help with bugs they’ve found.

Examples of questions from January 2019:

“How do we setup a new transaction process?”

“What’s the alias for the default behaviour?”

“It's possible to build new transaction processes from the console?”

Examples of questions from June 2019:

“Is it possible to remove transition "review-1-by-provider" from the process “test-process-1?”

“Could you add action that allows sending protected data in transitions "instant-booking" and "booking-request" to transaction process on "marketplace-site-test" for me

“Please update the transaction process for us using below diagram. Let me know if you have any questions on that.”

During the first period support requests varied from simple questions searching for information to questions inquiring if certain changes were possible, and if so, how they could be done. The questions showed that the user lacked basic information on their transaction processes - what processes they have in use, what aliases they have and how they function. The requests made were often ambiguous, as the users were not able to use exact names for e.g. transitions.

During the second quarter support most requests included clearly defined questions that showed that the customer had an clear understanding of how their transaction process works and what they want to change. User regularly used graphical or EDN descriptions of their process to communicate their change requests. Most user were able to provide exact names for transaction process elements such as transitions. The customers were ready to discuss workarounds and approached the team with detailed process descriptions. The conversation has clearly sifted from inquiring if something is possible to directly requesting changes.

The Table 9. shows a steady increase in the quality of support tickets. The results show that the implemented ways to offer users more information worked. Users have better understanding of their marketplace processes, their details and how they work. The users were able to find answers to basic questions on how the transaction process works and how it can be used for their advantage.

From this it is possible to deduct that publishing information on the users transaction process versions and aliases along the transaction process documentation helped to reduce the support ticket load. Publishing the transaction process visualization helped customers to actively develop their marketplace and to plan their transaction processes, giving them tools to communicate the changes better.

Furthermore, the transaction process visualization is in active use when communicating with the customers. Over half of the users used either the graphical description or the EDN description to communicate change requests.

6.2 Enabling users to customize their transaction process

This chapter seeks to compare what kind of transaction process changes do customers request and how well do our solutions answer to these needs. This section evaluates how well do the command-line interface tool and tool for adding functionalities answer to customer needs, and if it is possible to find similarity in the users' requirements for their transaction processes.

Table 10. Amount of transaction process related tickets by type

Ticket type	**Total amount of transaction process related tickets January to July**	**% of all**
Modify transaction process	77	33,0
Asking for information	56	23,9
Email template change	31	13,2
Change transaction process model	20	8,5
Add a published feature to transaction process	20	8,5
Add protected data	19	8,1
Commission % change	11	4,7
Total	234	100

The Table 10 shows that transaction process modification requests make up one third of all transaction process related support tickets. These requests are unique change requests that change the structure or functionality of the transaction process and don't otherwise fall in to other the categories. Examples of this kind of changes are adding an instant booking option or a process for evaluating the product conditions.

The second most common change request is editing the transaction related notifications. Editing these emails to correspond the marketplace branding is commonly sought out.

Changing the transaction process type or adding functionalities or protected data are equally popular. These changes include adding readily made features and functionalities into transaction process, for example strong customer authentication. Changing the transaction process type means changing it to another type supported by Sharetribe, for example from day-based bookings to time-based bookings. Adding protected data again allows the customer to store information within transaction process.

In the Table 11 table is listed the transaction processes modifications made for customers - these correspond to the transaction process change requests.

Table 11. Customer's transaction processes and process modifications in order of launch

Customer	Processes	Process modifications	Changes to actions
A	1	No changes	-
B	1	Cancellation with refund. Cancellation without refund.	Custom pricing Commission changes
C	1	Customer cancellation	Commission changes
D	1	Simplified review process.	
E	1	Instant booking. Customer and provider condition reports. Customer drop-off report. Customer and provider cancellation.	
F	2	Price negotiation. Cancellation with refund. Cancellation without refund.	Commission changes
G	1	No changes	

H	1	No changes.	Commission changes
I	2	Price negotiation. Disputation.	Commission 0 process
J	1	No changes. (Protected data updates)	update-protected-data
K	3	Cancellation with refund. Cancellation without refund. Disputation. Pick-up and pick-off. (Complex processes)	update-protected-data
L	2	Instant booking.	
M	1	No changes (Custom pricing actions)	
N	2	No changes	Commission change
O	1	No changes. (Commission changed)	
P	2	Price negotiation.	update-protected-data
Q	2	Price negotiation. Disputation. Customer cancellation.	update-protected-data
R	1	No changes.	
S	1	No changes	
T	1	No changes	
U	1	No changes	
V	1	Instant booking. Customer cancellation. Pick-up and pick-off. Disputation. (Complex process)	
W	1	No changes	
X	1	No changes	
Y	1	No changes	
Z	1	No changes	custom-pricing
AA	1	Instant booking.	update-protected-data

Table 12. Frequency of transaction process changes made

Change	For how many of 27 customers
No changes to transaction process structure	15 (55,6%)
Disputation	5 (18,5%)
Instant booking	4 (14,8%)
Refund	4
Price negotiation	4
Pick-up and/or pick-off	3
Customer cancellation	3
Review process simplified	1

Table 11 shows changes done to the process structure. This means it does not include changes to to parameters (such as the commission percent), actions or to notifications.

The table 12 shows that 55,6% of the customers, a bit more than half, make no modification to the transaction process structure. This means that the results indicate that most customers do not have the need or will to change their transaction process in the early stages of their marketplace career. This would indicate that the need for low cost, quick changes is not significant, or that customers have not understood or willing to work for understanding how process change could benefit them, as this would need resources from a technical person. However, it is noteworthy that this number does not include changes in mail templates or actions, which also require transaction process changes.

From table 11 it is possible to see that the customers that have launched later have more seldomly made changes to their transaction processes. This means that the hypothesis that the amount of transaction process related tickets has not increased is partly due to the fact that new customers do not request as much customizations to their transaction processes. It is likely, though, that when the operators gain experience on operating their business, they realize that they have the need to change the transaction process.

Customizations such as disputation, instant booking and refund are done by 14,8% of all the customers considered. Ifonly those who have made changes to their transaction process (12 customers) are considered, then 33,3% of the customers have made these changes.

This is a considerable amount that gives us strong evidence on the similarity of customers' needs

There are patterns on what are wanted changes, and some changes clearly arise above others. In some cases the customers have had different ways to implement these features, but the basic functionality has stayed relatively same. In other cases, like in instant booking, the function has implemented in a completely same way.

The results show that a tool for choosing and adding certain features to their initial transaction process could benefit customers, as it is possible to find similarities in the processes that customers wish to have. The designed solution suggested that the customer could add features such as instant booking, bargaining option, customer cancellation, refund and condition check for the product in Console. Additionally, the user could decide their commission percent and other configurations. These additional features would have corresponded to 16 out of 24 (66,7%) structural changes customers have made in their processes. However, it is noteworthy that this doesn't include email template changes, changes in actions or later changes in commission. If these changes are considered, the actual number is lower.

This means that out of the initial changes, 66,7% could be answered by a tool for adding additional features to the initial transaction process, while the other 33,3% of the processes should have been edited with the command line tool.

7. Analysis and discussion

In the evaluation it was found that after transaction process listing, graphs and documentation were published, fewer customers supports requests considering transaction process related information were received. This implies that the customers were now able to find the information they needed without contacting support.

The qualitative analysis showed that the customers were able to ask more informed and more precise questions about transaction processes and were more informed on how transaction processes could be used to solve their problems. Customers were also able to use the right terms when referring to the transaction process and its functionalities. This is likely due to the transaction process graph that shows the user their transaction process details including transitions, actions and parameters. With this help, the customers were able to discuss their transaction processes in detail with the support.

Even though the job stories noted customizing transaction process emails as one significant user need, only 13,2% of the transaction process related support tickets considered transaction process notification changes. In the design phase implementing a tool for customizing transaction process emails in Console was considered, but the actual implementation was found to to be complex and lacking the ability to answer customer needs. As transaction process emails use parameters defined earlier in the transaction process, validating the emails would not have been possible as it would not have been possible to simulate the right process structure. The error margin in this implementation would have decrease the usability of the solution and might even lead to increasing amount of support requests. Also, as any changes to new these notifications requires creating a new transaction process version, any change would have required updating the aliases to point to the new version. Considering these aspects and the financial strain that implementing this tool would cause, it was decided to not to build such tool.

Adding functionalities to the transaction process was found to satisfy one third of all the customer needs that had been implemented during the observation period. It was possible to identify a clear set of transaction process features that were frequently requested.

The design suggested five functionalities that the customers could add to their initial transaction process. When compared to the processes customers had built during the observation period from January to July 2019, these five functionalities were able to cover 66,6% of all the functionality needs. However, it has to be considered that this number doesn’t include commission changes, changes in action or changes in notifications. It is also likely that not all customers are satisfied with the ready made functionalities, but wish

to modify them to exactly to their needs. This means that it is unlikely that the solution would answer two thirds of all the needs in reality.

The results show that 55,6% of customers make no changes to their transaction process, except possible changes in the notifications and actions. The tool for adding functionalities to the initial transaction process could help the users to understand the possibilities of the transaction process, and how they could best use it to their advantage. It's an easy start for new customers, and might inspire those who otherwise would not make changes to their transaction process. The tool brings additional value also by giving customers autonomy over their transaction process and enables modifying the transaction process also for those with less resources.

As more customers build their transaction process, it is possible to collect more data on the functionalities that are popular and develop the offering of readily made transaction process functionalities even further.

The command line interface for editing the transaction process EDN enables full customizability for all customers. The command line tool is especially useful for development partners who build transaction processes regularly. Together with the tool for adding functionalities, these enable a wide range of customizability for users with different resources and skills.

The users interviewed for the user research were early stage customers and therefore early adopters of the new product. These people have in common strong background in business or technology and often a competent technological lead in the team. It is unclear how well the needs of this group will correspond the larger customer base. It is likely that some or even most customers will not have as competent skill base.

The thesis evaluates how well the customers needs are and would be met by the solutions presented. The actual realization of the customization tools were not in the scope of this thesis and therefore no user evaluation was done

From resource point of view, implementing these changes at the moment of writing this thesis or in the near future might not make sense, as the workload on these changes is significant, but the actual improvement from customer's point of view is limited. However, when it comes to user experience, these changes could give the operator a sense of autonomy, especially for those without development resources.

All in all the process of collecting user needs through interviews and user sketches to categorize them into job stories and designing solutions based on this information has proved to be accurate process for designing efficient solutions to complex problems.

In section 2.1 Designing complexity, Hevner (2004) suggested addressing difficult design problems by dividing the problem into smaller subproblems and solving these iteratively. This was leading theme in the process presented in this thesis. Job stories were chosen as the tool to divide the user needs into smaller substories, and the final design was built in four iterative phases.

This approach allowed us to chunk the problem into more approachable pieces, and it was possible to find out two almost completely distinct problems areas: informing the customers about transaction processes and enabling them to customize their own process. This is in line with what Zmiro (2017) suggested in section 2.6 Case study: Intercom Chat Bot.

Visualizing the transaction process was one of the main challenges in the process. In section 2.1 Fitzpatrick (2016) noted that when users mental models match how the actual system model works, people find the user interface intuitive. This was basis on doing the interview with user sketching presented in User research section. These sketches provided us with information on users mental models and helped understanding the best practices when visualizing the process.

The user sketches revealed that three out of four users used flowcharts and one user used role action diagram - both introduced by Aguilar-Saven (2002) in section 2.3 Process modelling. It seemed clear that the users used process models that they had encountered most in their own area of work. The user using role action diagram was from operating position, and two of the three users using flowchart were from technical background.

The main takeaways from the section 2.2 Information visualization were from Keim et al (2018). These suggested that only relevant data should be represented and it should be grouped so that it creates meaningful entities and the information should only relate to the task at hand. As the transaction process EDN only shows information by type, the graph was to show information by state - as most tasks were related to understanding how specific transition works. This provided a natural way of grouping and showing the information.

In section 2.4 Developer experience Bohwmick (2018) advices to use tools that are already familiar to developers and warns from assuming that a graphical interface is necessary. This, together with his notion developers “can handle more complexity in their products than other users we might be used to designing for”, were strong indicators that using command line interface could be a suitable design solution. The user interviews reassured that the developers are familiar with command line use, and felt more need for visual explanation of the process than for visual components for the actual tooling.

In the section 2.6 Donhue and Shepard (2018) explained that they decided to go against WIMP tool as it would become overwhelming. Similarly, in section 2.5 Mathieson and Keil (1998) suggested that the tool should also match the complexity of the task. These notions led to using CLI as a primary tool for transaction process customization, as it would fit the complexity of the task and would offer the users the desired flexibility. Additionally the visual interface for choosing the initial transaction process was supported by Davis (1997) finding that users were most comfortable with the interface style they were familiar with. In this way, both developers and operators needs were met.

8. Conclusions

The research question and the sub questions were defined as follows:

What kind of tool for customizing transaction process offers additional value to operators?

> *1.1 What kind of needs do marketplace operators have considering customizing their marketplace's transaction process?*
>
> *1.2 What kind of needs do developers have considering the tool for customizing transaction process?*

Operator and developer needs were translated into job stories. Answering to these job stories in an efficient, usable and scalable way will bring additional value to marketplace operators. These job stories were divided into two groups: those considering information sharing and retrieval, and those considering transaction process customization. Incremental solutions were designed to answer these needs. It was discovered that the tool consisted of several incremental solutions that all brought additional value to the customers.

From these solutions, creating a listing of transaction processes and their aliases, and a graphical description of the customer's transaction processes were implemented. The designed solutions for modifying transaction process notifications and transaction processes were designed but not implemented during the time of writing this theses, but they were evaluated against the customers' implemented transaction processes.

Job stories 1 to 4: Information sharing

The job stories one to four reflect the customer's' need for information. The needs concentrate on the need to know what kind of processes the customer has in use, how they work and what are the best practises for customization. The solution designed to answer these needs consists of three parts: transaction process and version listing in Console, graphical presentation of customer's transaction processes with technical information on it, and an information package at sharetribe.com/docs including information on what is transaction process, how it works, how to take advantage of it etc.

The results show that the amount of tickets requesting for basic information declined over the observation period. Also, the amount of question asked per customer has been in decline. After the designed solutions were published in early February and late March, the

amount of tickets requesting for transaction process related information has declined steadily. This indicates that the users have been better informed.

The qualitative analysis of customer support tickets showed that customers were able to ask more informed and more precise support questions after the transaction process listings and graphs were published. The analysis showed that customers were more aware of the transaction process functionalities and terms, and were able to recognize their possibilities and problems better.

Considering the results against the evaluation criteria presented in section 3.4 (“the quantity of the tickets should decrease while the quality should increase”), it is possible to say that the designs were successful.

Job stories 5 to 6: Transaction process customization

The job stories five and six reflected the users’ need to customize their transaction process related notifications and the transaction processes itself in a flexible and error-free way. The solutions designed to answer to these needs consists of a tool for editing transaction process related emails, a tool for choosing and adding functionalities to user’s initial transaction process, and a command line tool for freely editing the chosen transaction process.

The results showed that even though most customers made no changes to their transaction processes, there were clear regularities in the functionalities customers implemented in their transaction processes during the observation period. 66,6% of the customers’ implementations could have been accomplished with the tool used in Console. The rest 33,3 could have been created in command line tool.

The tool for editing transaction process related notifications would have answered to only a small amount of transaction process related requests. However, it would not have worked too well with these requests and would not have met all the customers’ requirements. Therefore, the solution could be discarded and the modifications should be done by using the command line tool.

Considering the results against the evaluation criteria presented in section 3.4 (“The hypothesis is that it is possible to find similarities in the customers’ requirements considering their transaction process. The tool would be successful, if it could satisfy half of these needs.”), it is possible to say that the designs were successful.

The user research and categorizing the user needs by using job stories as a framework proved to be an efficient process for designing effective tools for the selected purpose. The evaluation proved that the tools were able to answer to the users needs. However, the evaluation was not able to answer how usable to the tools were, or how the users felt about using the tools. Also, the actual rate of use and the iterative improvement process were left out.

In further studies, the use rate between the graphical tool and the command line tool could be evaluated to better understand users' preferences. For future studies using similar process,usability evaluation would be beneficial and offer more in-depth information about the efficiency of the design process as well as confirm the design applicability.

References

Aguilar-Saven, R. S. (2004). Business process modelling: Review and framework. *International Journal of production economics,* 90(2), 129-149.

Alsever, J. (2013). The “mega trend” that swallowed Silicon Valley. *CNN Money.* Retrieved from http://tech.fortune.cnn.com/2012/10/03/the-mega-trend-that-swallowed-silicon-valley/

Basselier, R., Langenus, G. and Walravens, L. (2018, September). The rise of the sharing economy. *NBB Economic Review*, 57-78

Bhowmick, A. (2018, July 25th). *Designing for developers*. Retrieved from https://uxdesign.cc/designing-for-developers-e44280438e27

Bor D, Duncan J, Wiseman RJ & Owen AM. Encoding strategies dissociate prefrontal activity from working memory demand. *Neuron* 2003; 37: 361–7

Card, M. (1999). *Readings in information visualization: using vision to think*. Morgan Kaufmann.

Christensen, C. M., Anthony, S. D., & Roth, E. A. (2004). Seeing what's next: Using the theories of innovation to predict industry change. *Harvard Business Press*.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, 319-340.

Davis, S., & Bostrom, R. (1992). An experimental investigation of the roles of the computer interface and individual characteristics in the learning of computer systems. *International Journal of Human‐Computer Interaction*, 4(2), 143-172.

Donhue, B., & Shepard, D. (2018, August 15). *Embracing complexity: designing our Custom Bot builder.* Retrieved from: https://www.intercom.com/blog/embracing-complexity-designing-custom-bot-builder/

Fagerholm, F., & Münch, J. (2012, June). Developer experience: Concept and definition. *In Proceedings of the International Conference on Software and System Process* (pp. 73-77). IEEE Press.

Feizi, A., & Wong, C. Y. (2012, June). Usability of user interface styles for learning a graphical software application. *In 2012 International Conference on Computer & Information Science* (ICCIS) (Vol. 2, pp. 1089-1094). IEEE.

Fitzpatrick, M. (2016, May 11th). *How users understand new products.* Retrieved from: https://www.intercom.com/blog/videos/users-understand-new-products/

Fuller, R. B. (1992). Cosmography: A posthumous scenario for the future of humanity. The Estate of R. Buckminster Fuller.

Gall, J. (1975). *General Systemantics: An Essay on how Systems Work, and Especially how They Fail, Together with the Very First Annotated Compendium of Basic Systems Axioms: a Handbook and Ready Reference for Scientists, Engineers, Laboratory Workers, Administrators, Public Officials, Systems Analysts, Etc., Etc., Etc., and the General Public.* General Systemantics Press.

Hamari, J., Sjöklint, M., & Ukkonen, A. (2016). The sharing economy: Why people participate in collaborative consumption. *Journal of the association for information science and technology*, 67(9), 2047-2059.

Hasan, B., & Ahmed, M. U. (2007). Effects of interface style on user perceptions and behavioral intention to use computer systems. *Computers in Human Behavior*, 23(6), 3025-3037.

Heer, J., Van Ham, F., Carpendale, S., Weaver, C., & Isenberg, P. (2008). Creation and collaboration: Engaging new audiences for information visualization. *In Information Visualization* (pp. 92-133). Springer, Berlin, Heidelberg.

Hevner, A. R.; March, S. T.; Park, J. & Ram, S. Design Science in Information Systems Research. *MIS Quarterly*, 2004, 28, 75-106.

Hickey, R. 2018. *edn*. Retrieved from: https://github.com/edn-format/edn

Holton, E. F., & Burnett, M. F. (2005). The basics of quantitative research. *Research in organizations: Foundations and methods of inquiry*, 29-44.

Hotjar, 2019. Retrieved from: https://www.hotjar.com/tour

Intercom. (2016). *Intercom on Jobs to Be Done*. Retrieved from https://www.intercom.com/books/jobs-to-be-done

ISO 9241-210: 2010. (2010).. Ergonomics of human system interaction-Part 210: Human-centred design for interactive systems. International Standardization Organization (ISO). Switzerland.

Keim, D., Andrienko, G., Fekete, J. D., Görg, C., Kohlhammer, J., & Melançon, G. (2008). Visual analytics: Definition, process, and challenges. *In Information visualization* (pp. 154-175). Springer, Berlin, Heidelberg.

Kerren, A., Stasko, J., Fekete, J. D., & North, C. (Eds.). (2008). *Information Visualization: Human-Centered Issues and Perspectives* (Vol. 4950). Springer

Klement, A. (2013). *Designing features using Job Stories*. Retrieved from https://blog.intercom.com/using-job-stories-design-features-ui-ux/

Klement, A. (2016). *When Coffee & Kale Compete: Become Great at Making Products People Will Buy*. Retrieved from http://www.whencoffeeandkalecompete.com/

Kothari, C. R. (2004). *Research methodology: Methods and techniques*. New Age International.

Kraljic, A., Kraljic, T., Poels, G., & Devos, J. (2014). Business process modelling in ERP implementation literature review. *In 8th European Conference on IS Management and Evaluation* (ECIME) (pp. 298-308). Academic Conferences and Publishing International Limited.

Kuniavsky, M. (2003). *Observing the user experience: a practitioner's guide to user research*. Elsevier.

Lakin, R., Capon, N., & Botten, N. (1996). BPR enabling software for the financial services industry. Management services, 40(3), 18-20.

The Linux Information Project. (2004). *GUI Definition*. Retrieved from http://www.linfo.org/gui.htm

Mathieson, K., & Keil, M. (1998). Beyond the interface: Ease of use and task/technology fit. *Information & Management*, 34(4), 221-230.

Miller, G.A. (1956), The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. Psychological Review, 63, 81-97.

Neath, I., & Surprenant, A. M. (2003). In Taflinger M. (Ed.), *Human memory* (2nd ed.). Canada: Vicki Knight.

Nielsen, J. (1994). Usability engineering. Elsevier.

Nielsen, J., & Levy, J. (1994). Measuring usability: preference vs. performance. Communications of the ACM, 37(4), 66-76.

Onwuegbuzie, A. J., & Combs, J. P. (2011). Data analysis in mixed research: A primer. *International Journal of Education*, 3(1), 1.

Phalp, K. T. (1998). The CAP framework for business process modelling. *Information and Software Technology*, 40(13), 731-744.

Purchase, H. C., Andrienko, N., Jankun-Kelly, T. J., & Ward, M. (2008). Theoretical foundations of information visualization. In *Information Visualization* (pp. 46-64). Springer, Berlin, Heidelberg.

Richter, C., Kraus, S., Brem, A., Durst, S., & Giselbrecht, C. (2017). Digital entrepreneurship: Innovative business models for the sharing economy. C*reativity and Innovation Management*, 26(3), 300-310.

Rittel, H.J., & Webber, M.M. (1984). Planning Problems are Wicked Problems. *Developments in Design Methodology*. N. Cross (ed.), John Wiley & Sons, New York.

Rowley, J. (2012). Conducting research interviews. *Management research review*, 35(3/4), 260-271.

Sandelowski, M., Voils, C. I., & Knafl, G. (2009). On quantitizing. *Journal of mixed methods research*, 3(3), 208-222.

Sharetribe. (2019). Transaction process. Retrieved from https://www.sharetribe.com/docs/background/transaction-process/

Shneiderman, B. (1993). 1.1 direct manipulation: a step beyond programming languages. *Sparks of innovation in human-computer interaction*, 17, 1993.

Shneiderman, B & Plaisant, C. (2004) *Designing the User Interface, Strategies for Effective Human–Computer Interaction*. Reading, MA: Addison Wesley.

Simon, H. A. (1996). The sciences of the artificial. MIT press.

Thomas, D. R. (2006). A general inductive approach for analyzing qualitative evaluation data. *American journal of evaluation*, 27(2), 237-246.

Townsend, S. (2016, October 5). *Understanding your users' mental model*. Retrieved from https://www.intercom.com/blog/videos/understanding-your-users-mental-model/

Ulwick, A. (2016). *Jobs to be Done: Theory to Practice*. Idea Bite Press.

UML. (2003). Resource Center. Retrieved from http://www.rational. com/uml/index.jsp.

Vaishnavi, V., Kuechler, W., & Petter, S. (Eds.) (2004/19). "Design Science Research in Information Systems" January 20, 2004 (created in 2004 and updated until 2015 by Vaishnavi, V. and Kuechler, W.); last updated (by Vaishnavi, V. and Petter, S.), June 30, 2019. Retrieved from http://www.desrist.org/design-research-in-information-systems/.

Walliman, N. (2017). *Research methods: The basics*. Routledge.

Wang, C., & Zhang, P. (2012). The evolution of social commerce: The people, management, technology, and information dimensions. *Communications of the Association for Information Systems*, 31(1), 105–127.

Westerman, S. J. (1997). Individual differences in the use of command line and menu computer interfaces. *International Journal of Human-Computer Interaction*, 9(2), 183-198.

Wiedenbeck, S., & Davis, S. (1997). The influence of interaction style and experience on user perceptions of software packages. *International Journal of Human-Computer Studies*, 46(5), 563-588.

Wilson, C. (2013). *Interview techniques for UX practitioners: A user-centered design method.* Newnes.

Zmiro, J. (2017, November 15. *The hidden cost of design complexity*. Retrieved from https://www.intercom.com/blog/the-hidden-cost-of-design-complexity/

Appendix A - Interview template for operators

Interview questions:

Your team

- Would you like to tell something about you marketplace and your team?
- Who are the people in your team and what are their expertises?
- Could you tell more about your developers, where are they from (freelance, consultant) and skills?

Building your marketplace

- How much time do you spend on your project?
- What kind of budget do you have?
- What are your goals with your marketplace?

Transaction process

- How familiar you are with the transaction processes on your markeplace?
- Would you like to tell me about your marketplace's transaction process?
- How do you found the Sharetribe's default process suit for you?
- Is there something you would need to have?
- Is there something you would like to have?

Customization

- Have you customized the process?
- How do you customize the process at the moment?
- Have you added some emails etc.?
- How often do you have the need to do changes?
- What kind of changes?
- Why?

How the customization is done

- Would you like to be able to customize the process more fluently?
- What would help you to do that?
- Would you like to be able to do it yourself or with a help of a programmer?

Appendix B - Interview template for developers

Interview questions:

Background

- What is your personal background with web-development?
- Could you tell something about the project you are on?
- What is the budget? How long will you be working on it?

Communication

- How did you come to contact with the marketplace founder?
- How do you communicate with the marketplace founder?
- How do you find the communication? Is it easy to understand what they are after?

Transaction process

- Are you familiar with transaction processes?
- Would you like to tell me about the transaction process in the marketplace you are building?
- How does Sharetribe's default process works for you?
- Do you find it difficult or easy to understand? Why?

Customization

- Have you customized the process?
- How do you customize the process at the moment?
    - Have you added some emails etc.?
    - How often do you have the need to do changes?
    - What kind of changes? Why?

How the customization is done

- How do you find the customization process?
- Have you encountered problems in the development? What kind?
- What would help you to work with it?
- Do you have difficulties understanding what is happening in the code?

Communication on what will be done

- How does the communication on what is wanted happen?
- Would you like to be able to have the introductions as text or charts or visualizations of some kind?

Appendix C - Interview template and assignment for user sketches

Setup:

- 1 hour of time
- One A4 for each participants
- A few pens to choose from

Interview and assignment:

- Could you tell me about the transaction process you have at the moment?
- Now, here are some pens and paper - Could you draw me what your transaction process looks like at the moment? You have ten minutes.
- Could you explain to me what you have drawn?
- Is there anything you would like to change or add to your process?