**User Requirement Analysis**

Table of Contents

[Establishing Requirements 3](#_Toc63529526)

[User Requirement Templates 4](#_Toc63529527)

[Requirement Types 5](#_Toc63529528)

[Example 6](#_Toc63529529)

[Data Gathering 7](#_Toc63529530)

[Data Interpretation and Analysis 9](#_Toc63529531)

[Task Descriptions 10](#_Toc63529532)

[Personas 10](#_Toc63529533)

[Scenarios 12](#_Toc63529534)

Most successful system development is user-centred, meaning the opinions of the user is very important. We want to ensure that the user is satisfied with whatever system is being developed. This is why user requirement analysis is important.

We need to understand as much as possible about the users, their work and the context of that work, so that the system we make can support them achieve their objectives. This involves understanding their demographic, their age group, their work, how they would use the system, in what environment and at what times, etc. This process is called ‘identifying needs’.

Through this process, we can create a set of functional and non-functional requirements so that we, as well as our users, have a clear picture of what we are trying to achieve. This is done through a few steps:

* Data Gathering Activities
* Data Analysis Activities
* Expression as Requirements

The user requirement stage is where failure is most likely to occur. It is very easy to have miscommunications between the users and the project team members, which leads to a domino effect of problems. Even after the initial communication, every person in charge of the different parts could take away a different interpretation of the users demands, making the final product wildly different from the desired product. Worse still, users, especially non-technical ones, tend to be unable to properly communicate their desires, making the process more difficult. Thus, it is extremely important that we make the utmost efforts to understand our users and their demands properly from their perspective.

## Establishing Requirements

A requirement is a statement about an intended product that specifies what it should do or how it should perform.

The first thing we need to do is understand what the user wants and what the user needs. The difference between these two things is important. Needs are things that our system should most definitely have since they will help our users achieve their objectives. Not having these will cause our system to not work properly. Wants are not critical. We should try our best to include them, but we can also manage without.

Requirements should also be very concise so as to avoid difficulties understanding them which would result in errors while documenting the requirements. The requirements can be presented using things like charts and graphs to show responses from users. The requirements need to be justifiably connected to the data we collected. For example, if one of the requirements of our system is that a particular interface have an animation, we should be able to show with concrete data that the users actually asked for this feature and it is not something we are including based on some assumption that the users will want this.

## User Requirement Templates

The following is one of the templates used to report user requirements. It is called a Volere Shell template.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement No.**: 75 | **Requirement Type**: 9 | | **Event/Use Case No.**: 6 |
| **Description**: The product shall issue an alert if a weather station fails to transmit readings. | | | |
| **Rationale**: Failure to transmit readings might indicate that the weather station is faulty and needs maintenance, and that the data used to predict freezing roads may be incomplete. | | | |
| **Originator**: Road engineers | | | |
| **Fit Criterion**: For each weather station, the recorded number of each type of reading per hour shall be within the manufacturer’s specified range of the expected number of readings per hour. | | | |
| **Customer Satisfaction**: 3 | | **Customer Dissatisfaction**: 5 | |
| **Priority**: High | | **Conflicts**: None | |
| **Supporting Materials**: Specification of Rosa Weather Station | | | |
| **History**: Raised by GBS - 28 July, 2005 | | | |

Notice how there is a requirement number, a requirement type, a short description of the requirement, what would happen if the requirement is not met, the general source from which the requirement was collected, ratings for how satisfied or dissatisfied a customer will be if the requirement is met or not met and the priority of the requirement.

## Requirement Types

In software engineering, there are essentially two types of requirements – functional and non-functional.

Functional requirements are very specific. These are things that can be achieved with a particular function designed to perform that particular task. Functional requirements tell us specifically what the system should do. For example, a smart fridge that is able to tell when the butter tray is empty. This is a very specific requirement that can be met with some function that we can design. Functional requirements are gathered by asking users about what things they specifically want.

Non-functional requirements are more generic, for example, the system needs to be secure. This is not specific and cannot be achieved using a particular function. It is actually a vague requirement that could involve a large number of features. If a specific feature is specified instead, such as verification of a pin number, then this is a functional requirement.

There are other requirements such as physical requirements about the amount of lighting and sound from the system, but those fall under the category of interaction design, which we will not be discussing for now.

Requirements are also heavily dependent on our potential users. We need to take into consideration what their abilities are, what their backgrounds are and how technically competent they are. For example, novice users need step-by-step, clear instructions while expert users need a wider range of control. Frequent users need shortcuts to get things done quickly, while infrequent ones need clear indications about how to use the system as they are likely to not remember.

### Example

Consider a self-service cafeteria that allows users to pay using their credit cards. Functional requirements could include things like notifications about pending charges or insufficient funds, a pin system, the ability to cancel orders, etc. Non-functional requirements include making the system in a way that users are familiar with and making sure that the payment system is secure. Environment requirements could include something like a small barricade to the physical machine so as to prevent others from viewing the user’s pin.

## Data Gathering

Now we need to understand how to gather the data we need to establish our requirements. If we are able to provide solid data about what users want, for example by stating the percentage of users from a certain demographic who want a specific feature, we can ensure that that feature is included in the final system.

The purpose of data gathering is to collect sufficient, relevant and appropriate data so that a set of requirements can be produced. Thus, we need to have a large userbase from which to collect our data, so as to avoid biases and we need to ensure the data we are collecting is related to the system we are developing.

Data can be gathered through:

* **Questionnaires** – These are good for collecting specific information from a very large and diverse group of users. We can collect both quantitative data, such as a measure of user satisfaction on some scale, and qualitative data, such as a description of the users’ experience with the system. The problem with questionnaires is that there may be very few responses and the responses are not always useful.
* **Interviews** – These refer to physical interviews with users, which can be time-consuming and somewhat intimidating for the user. However, interviews allow us to explore issues with users in detail and present different scenarios and prototypes to gain a lot of insight. The data gathered is more likely to be qualitative than quantitative. There should be a report of the interview.
* **Focus groups and workshops** – These are just group interviews. A group of users is selected to meet up and discuss and brainstorm different possible requirements. Focus groups are good for gaining a consensus view and multiple viewpoints. Some quantitative, but mostly qualitative data can be gathered.
* **Researching similar products**
* **Direct observation** – This involves actually observing users as they work (slightly creepy). Since the job is undertaken by a member of the team, it can take a lot of time and effort, but since the team member likely knows what to look for, it can result in a huge amount of useful, but qualitative, data.
* **Studying documentation** – This is a good source of data about the steps involved in an activity. We can study reports and system documentation to understand the system better and gather a lot of quantitative data. However, this method should not be used on its own, but rather in addition to other data gathering techniques.

Most of these techniques can be used together to create a better picture of our userbase.

For interviews specifically, we need to prepare beforehand and establish what we want to talk about with users. We need to decide who we want to interview and give them some details about what questions we will be asking beforehand so that they can prepare themselves. Questions can be open-ended, which the users can answer however they want, such as what they like or dislike about the system, or closed ended, which have very specific answers, such as the number of times they had to face some problem with the system.

## Data Interpretation and Analysis

Once we have gathered data on user requirements, we still need to represent that information in such a way that share-holders are able to understand the data easily. To do this, we can use several methods, such as:

* Data flow diagrams
* Class diagrams
* Sequence diagrams
* State chart diagrams
* Entity-relationship diagrams
* Use case diagrams

We will come back to these techniques and how to implement them later on.

## Task Descriptions

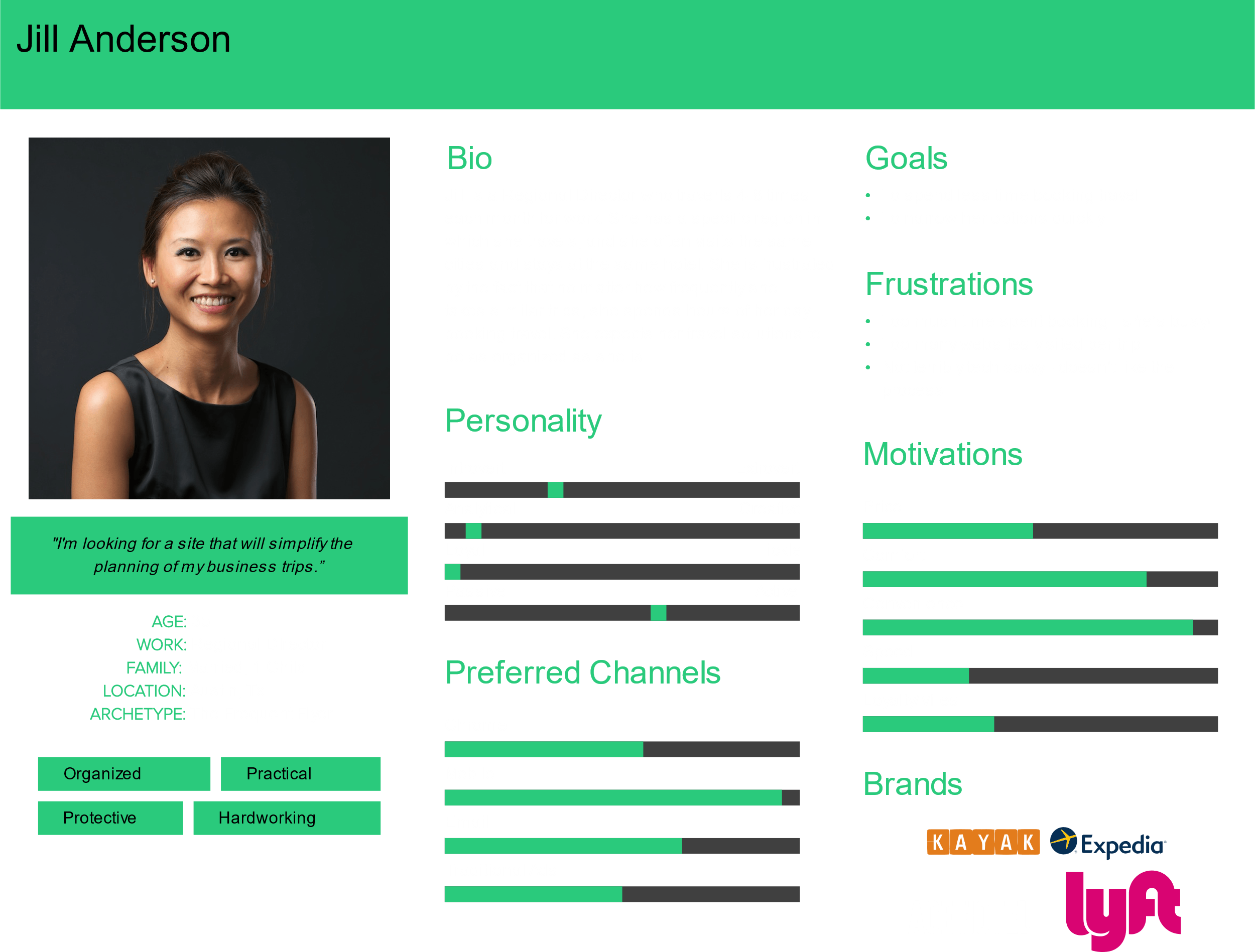
Task descriptions are used to describe the tasks we are about to perform so that users and share-holders are able to understand them. To do this, we will be using scenarios and use cases. To begin with, we will discuss scenarios, but before we can get to that, we need to discuss personas.

### Personas

A persona is a fictional user for our system. Each persona is specific to the system we are building. It captures a set of user characteristics, such as the name, age, their likes and dislikes, hobbies, etc. We have to bring it to life using characteristics, goals and a good background. This is not a real user of course, but one created using the information collected from our target user base. For example, the name will not be a real name, but the age will be a random age from the target user base, the hobbies will be hobbies that are common to the user base and so on.

The persona we create should not be idealized either. It should not be some perfect user, but should accurately represent the user base, including any flaws they may have.

An example template of a persona is given below. Templates like this one can be found [online](https://xtensio.com/user-persona/) and can be customized to the specific system we are creating.



Notice how there is a biography of what the person is like, to give the feeling that this character is real. The system being created here is a site that simplifies business trips. Information relevant to this system has been included, such as what this user wants, what problems she faces, what features are important to her and so on.

Remember that the data has to reflect the user base we are targeting, and we can do this by using the general data that we receive. The persona is imaginary, but not completely.

### Scenarios

Scenarios are like informal narratives or stories. They are simple, natural and personal. They cannot be generalized. With these scenarios, we will be using personas. For every scenario, we will be talking about different personas. The persona is the ‘who’, and the scenario describes what the character we created is doing, when they are doing it, where they are doing it. Essentially, we will describe the series of activities the character goes through to accomplish the goal.

Establishing scenarios is the first step in establishing requirements. Once we understand how a user would interact with the system, we can easily establish the requirements based on the scenario we have written.

Note that the scenario should not mention specific software or technology. We are trying to keep things very simple in a scenario.

One of the templates used to create scenarios is called the Baskinger template.

Say we want to develop a software that will serve as a holiday planner. An example scenario for this software is given below.

The Thomson family enjoy outdoor activity holidays and want to try their hand at sailing this year. There are four members of the family: Sky who is 10 years old, Eamonn who is 15 years old, Claire who is 35, and Will who is 40. While out on a shopping trip they call by at the travel agents in their local town to start exploring the possibilities. The travel organizer is located in a quiet corner of the agents’ office, where there are comfortable seats and play things for young children. They all gather around the organizer and enter their initial set of requirements — a sailing holiday for four novices. The standalone console is designed so that all members of the family can interact easily and comfortably with it. The system’s initial suggestion is that they should consider a flotilla holiday, where several novice crews go sailing together and provide mutual support for first-time sailors.

Notice how the example starts off by giving some details about the personas created earlier and some background about the group. Then it jumps into the details of what specific tasks they must finish, such as calling a travel agent, and contacting the travel organizer. It also gives some details about the environment in that situation, the fact that there are comfortable seats and toys for kids available in the agent’s office. Remember that we are trying to understand what they are doing, where they are doing it and when. The scenario needs to be able to answer these questions. After this, we also mention specific features that are available, such as the console that takes their initial requirements and the suggestions it gives them. This is the system that they actually work with.

One of the most important parts about the scenario we create is the goal we give to the persona. The goal defines what the persona needs to do, and the motivation behind their action. The scenario ends when the goal is reached.