Software Requirements

CS 346 Application Development

https://student.cs.uwaterloo.ca/~cs346

Contents

Getting Started 2
- The Product Vision 3
- Defining a Software Product 5

User-Centred Design 6
- What is UCD? 7
- Personas 9
- Scenarios 14
- User Stories 17
- Planning 19
- Scenarios vs. User Stories 20

Applying UCD 21
- Feature Identification 22
- Recording Features 23

Bibliography 24

Getting Started

The Product Vision

The starting point for product development is an idea, often articulated as a Product Vision Statement.

Product visions statements are simple statements that define the essence of the what you are developing. They should answer three fundamental questions:

- What is the product that we want to develop?
- Who are the target customers / end-users?
- Why should customers buy this product, instead of a competing product? What makes our product unique?

> Example: **What's for dinner** is a meal-planning application that takes the pain out of deciding what to cook for people who don't have time or energy to meal plan! Use your phone to take a picture of your refrigerator, and our AI agent will compile a list of quick and easy recipes that you can make based on the ingredients you already have.

The Product Vision

This template shows one possible structure for a Product Vision [1]:

- FOR (target customer)
- WHO (statement of the need or opportunity)
- The (PRODUCT NAME) is a (product category)
- THAT (key benefit, compelling reason to buy)
- UNLIKE (primary competitive alternative)
- OUR PRODUCT (statement of primary differentiation)

This isn't meant to be prescriptive, but it's a useful example that reinforces what is important: users, their needs, and product differentiators.

Defining a Software Product

In the early stages, you want to understand:

- Who are your users, and what needs are not being met.
- What product features address their dissatisfaction, and
- What they like and dislike about the products that they use? What improvements can we make to existing solutions?

These drive the design of software products:

- Business and consumer needs that are not met by current products.
- Dissatisfaction with existing software products, business or personal.
- Changes in technology that make completely new types of product possible.

User-Centred Design

What is UCD?

User-centered design is a design process that focuses on *identifying and meeting a user's needs*.

- Based on work by Rob Kling (1977) [2] and Don Norman (1986) [3].
- A process that considers *meeting user needs* to be paramount in product development.

We'll use three core concepts from UCD to help us define our features.

- **Personas**: an abstract representation of a user of a system.
- **Scenarios**: a high-level description of how a system will be used.
- **User Stories**: brief, informal descriptions of software features told from the perspective of the end-user.

These will help us decide what product to build, and what features will be most useful for our users.

What is UCD?

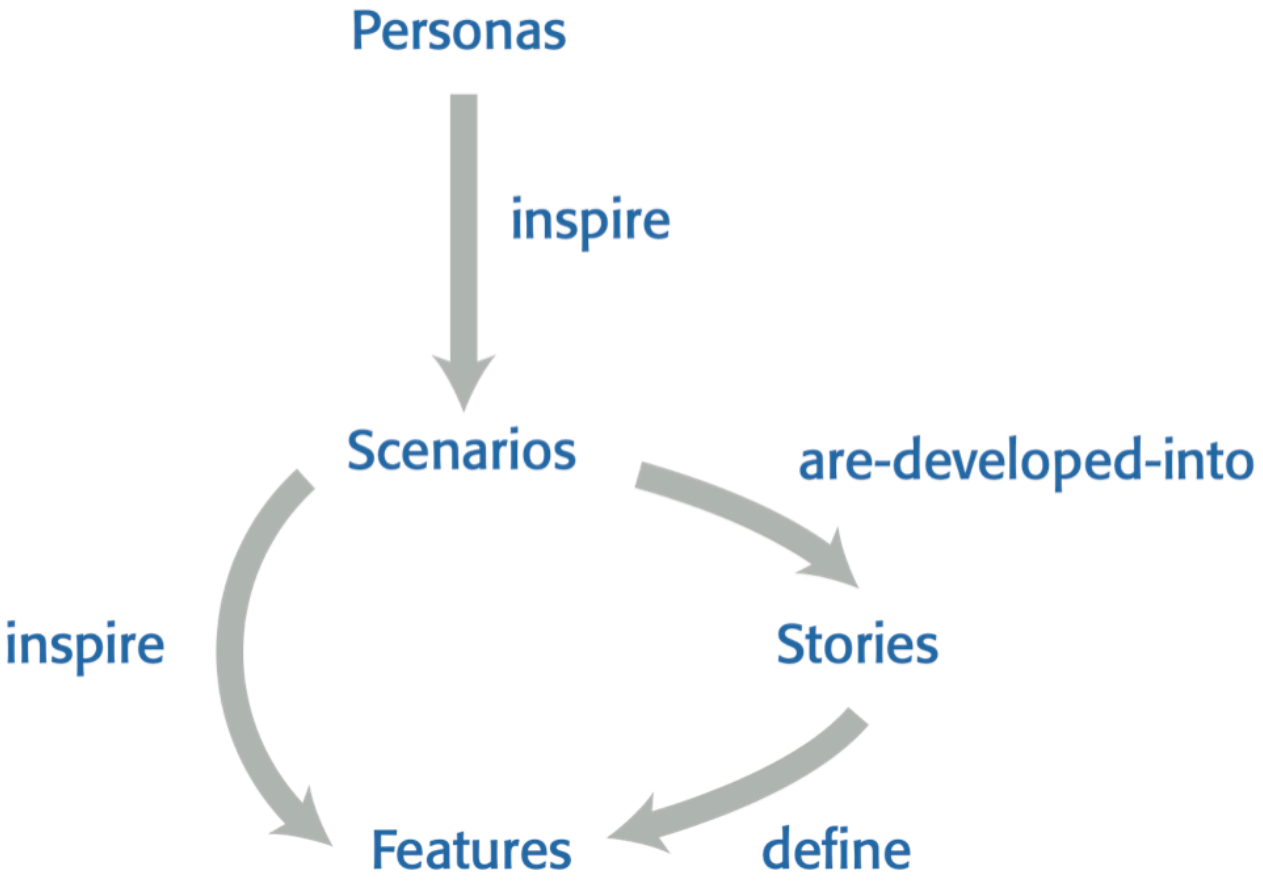

Figure 1: The core of user-centered design is developing stories that describe how users interact with your product. Diagram from Sommerville, Engineering Software Products. 2021.

Personas

You need to understand your potential users to design features that they are likely to find useful and to design a user interface that is suited to them.

Personas are “imagined users”, or portraits of a type of user that you think might use your product.

- e.g., if your product manages appointments for dentists, you might create a dentist persona.

It’s common for there to be more than one type of user. You may need to define multiple personas to model all of the other interactions!

- e.g., your dental appointment software might also need a receptionist persona (someone who books appointments) and a patient persona (if you want to send them reminders).

Personas

The Benefits of Personas

The main benefit of personas is that they help you and other development team members empathize with potential users of the software.

Personas help because they are a tool that allows developers to “step into the user’s shoes”.

- Instead of thinking about what you would do in a particular situation, you can imagine how a persona would behave and react.
- Check your ideas to make sure that you are not including product features that aren’t really needed.
- They help you to avoid making unwarranted assumptions, based on your own knowledge, and designing an over-complicated or irrelevant product.

Personas

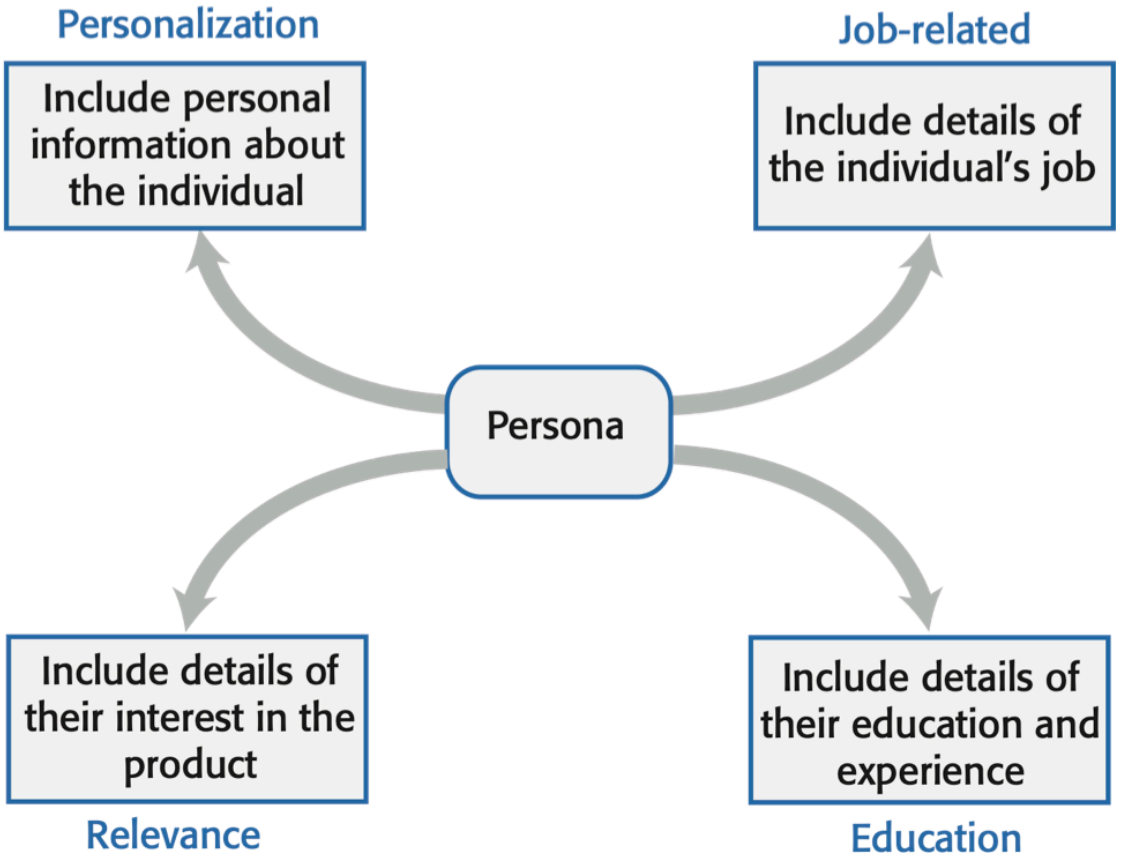

Figure 2: What to include in a persona. From Sommerville, Engineering Software Products. 2021.

Personas

Example

Dr. Candace Chou is a **dentist** starting up a new office on downtown Waterloo. She would a scheduling system to track her daily appointments. She is very busy, and values something that keeps her organized while at work and home.

Max is her **receptionist**. He's a trained dental receptionist, woth 10 years of experience, and he's quite knowledgable about dental and medical treatments. His job is hectic, and he often needs to juggle phone calls, email and interact with patients directly. He is very computer savvy, and prefers something that is quick for data entry.

Janice is a **patient**, who has recently signed up with Dr. Chou. She is 33 years old, and has excellent dental health. She has dental coverage through work, which means that she is able to book and elective appointments. She's tech savvy and likes getting reminders.

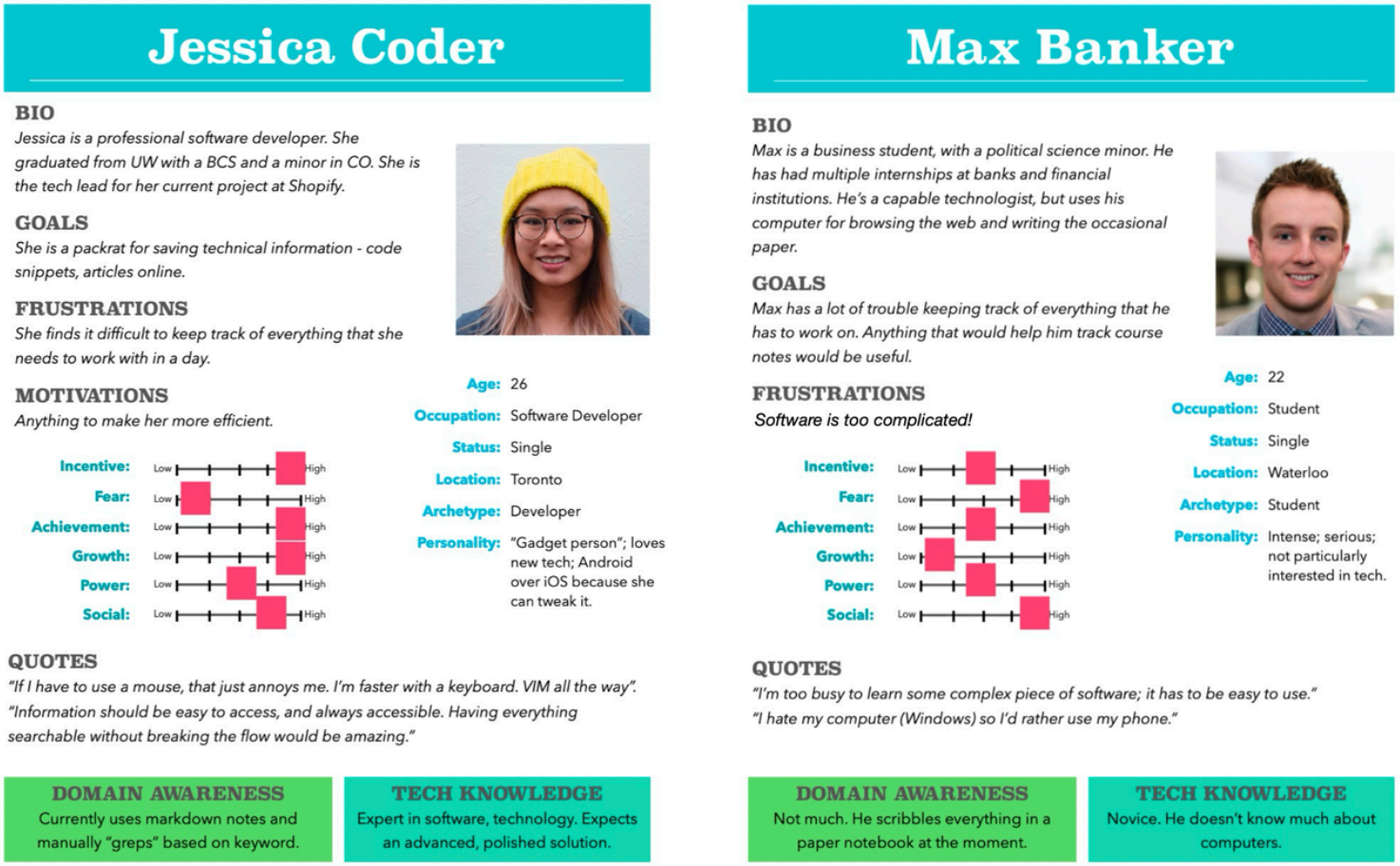

Figure 3: Personas are often narrative, but they don't need to be! Some people prefer summary cards like this for a quick reference.

Scenarios

A `scenario` is a narrative that describes how a user, or a group of users, might use your system.

A scenario is a description of a situation where a user is using your product's features to do something that they want to do.

- A scenario isn't a complete system specification.
- Scenario descriptions may vary in length from two to three paragraphs up to a page of text.
- You will likely have multiple scenarios!

Scenarios

Example

Booking an Appointment
- Overall Objective: Create a new dental appointment for a patient.
- Personas: Receptionist, Patient, Dentist

Janice needs to book a dental cleaning (i.e. a non-emergency dental appointment). She calls in and speaks to Max, the receptionist, who confirms her contact details, checks her dental history, and her coverage.

He confirms that her insurance requires her to wait 6 months from her last appointment, and suggests an upcoming date. She confirms, and Max books her in the scheduling system. He double checks her her contact information, including her phone number, so that the system can send out a notification later.

Once confirmed, the system automatically updates Dr. Chou's calendar.

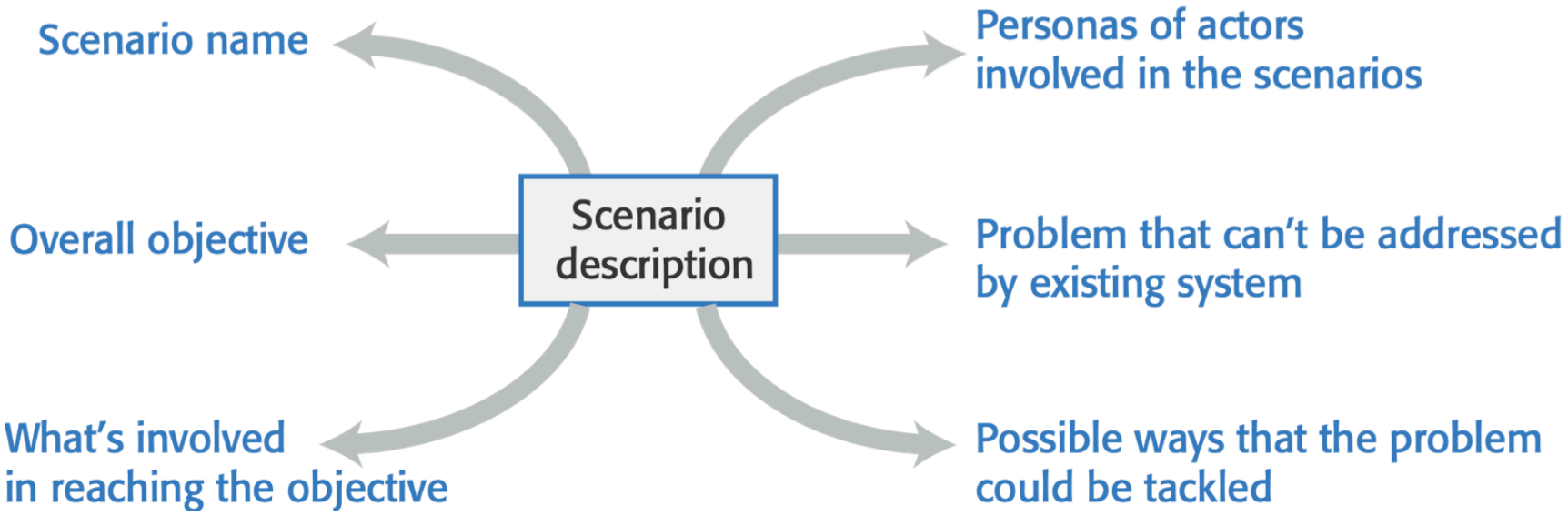

Figure 4: What to include in a scenario [4].

User Stories

Scenarios are high-level stories of system usage. They describe context.

User stories are finer-grain narratives that set out in a more detailed and structured way a single thing that a user wants from a software system; they *emerge* from scenarios.

Standard format of a user story:

- As a <role>, I <want | need> to <do something>
  - e.g., As a student, I need to be able to upload photos that I have found.

A variant of this standard format adds a justification for the action:

- As a <role>, I <want | need> to <do something> so that <reason>
  - e.g., As a teacher, I need to be able to report who is attending a class trip so that the school maintains the required health and safety records.

User Stories

Example

Your scenario could expand into multiple user stories. e.g.,

“As the receptionist, Max needs to be able to search for customer information and quickly bring up relevent records, including contact and insurance details.”

“As the dentist, Dr. Chou needs to be able to glance at her calendar and see her next appointment, including patient information and location.”

Planning

In Scrum, your product backlog is a collection of user stories.

- User stories should focus on a clearly defined system feature or aspect of a feature that can be implemented within a single sprint.
- During planning activities, you can expand into features aka issues.
- You are always working towards addressing the features for a user story.

If the story is about a more complex feature that might take several sprints to implement, then it is called an epic.

- e.g., As a system manager, I need a way to backup the system and restore either individual applications, files, directories or the whole system.
- There too much functionality associated with this user story!
- It should be broken down into simpler stories with each story focusing on a single aspect of the backup system, which can then be implemented.

Scenarios vs. User Stories

“If can express all the functionality described in a scenario as user stories, why do I need scenarios?”

- Scenarios are high-level and missing details.
- User stories provide those details.

Scenarios are helpful as a starting-point for the following reasons:

- Scenarios read more naturally because they describe what a user is doing with that system. People often find it easier to relate to this style of narrative.
- Scenarios often provide more information about what the user is trying to do and their normal ways of working. They provide context for the user stories.

Applying UCD

Feature Identification

Your goal in the initial stages of the project should be to create a list of user stories.

- Each user story contains one or more features.
- You complete the user story by delivering all of the associated features.
- All of your features are described this way.

Features should be independent, coherent and relevant:

- **Independent**: Features should not depend on other system features outside of the user story. Within a user story they may be related.
- **Coherent**: Features should be linked to a single item of functionality, and they should never have side-effects.
- **Relevant**: Features should reflect the way that users normally carry out some task. They should not provide obscure functionality.

Recording Features

In GitLab, you should be tracking the following:

Project proposal
- Personas
- Scenarios
- User stories
- Prototypes (TBD)

Milestones
- one milestone per demo
- each milestone contains 1+ issues
- each issue corresponds to a user story
  - subtasks represent features

Bibliography

[1] G. A. Moore, *Crossing The Chasm*. Harper Business Essentials, 1991.

[2] R. Kling, “The Organizational Context of User-Centered Software Designs,” *MIS Quarterly*, vol. 1, no. 4, pp. 41–52, 1977.

[3] D. A. Norman and S. W. Draper, *User Centred System Design*. CRC Press, 1986.

[4] I. Sommerville, *Engineering Software Products An Introduction to Modern Software Engineering*. London, UK: Pearson, 2021. [Online]. Available: https://iansommerville.com/engineering-software-products