

CS 398: Application Development

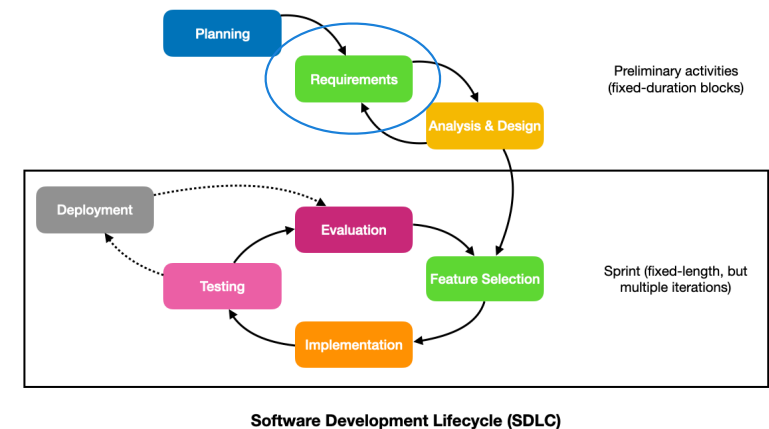
Week 02 Video: Requirements

Phase 2: Requirements

The requirements stage is where we explore the problem that we wish to address, and determine what to build. The output of this phase is a detailed list of product features that you would implement.

Attitudes shifted starting in the mid 1990s, towards trying to improve the overall customer experience. Product *design* became more important, and there was a major shift towards having dedicated product *designers who focused on users, their needs and what was desirable to them.*

This focus on users-first is a critical tenet of [User-Experience Design \(UXD\)](#).



What is User Experience Design (UXD)?

In the mid-1980s, Don Norman suggested that design should focus making products more usable for the people that use them. His book, The Design of Everyday Things became the foundation for the UX movement that followed.

Through the 1990s and early 2000s, significant research in HCI and related disciplines contributed to a whole-scale shift contributed to the widespread acceptance of human-centric product design. Today, it is widely accepted that our best practices in requirements and design are human-oriented*.

“User Experience Design (UXD or UED) is **a design process whose sole objective is to design a system that offers a great experience to its users**. Thus UXD embraces the theories of a number of disciplines such as user interface design, usability, accessibility, information architecture, and Human Computer Interaction.” — www.usertesting.com

* UXD is not just about design, it also applies to determining requirements.

Understanding Users

Why the focus on users?

Our first step is to gather information that will help us understand our users and their problems, and determine what features we need to design and implement for them.


Software development is all about solving problems for people. You want to make sure that you understand:

- Who are my users?
- What is important to them?
- What are they trying to accomplish?
- How do they relate to the problem that I'm trying to solve?

There is a difference between:

- **Stakeholders:** people that have some interest in your product.
- **Users:** people who could potentially use your product.

Your stakeholders are invested in your project for reasons that may be unrelated to actually using the product. Your users are the ones most invested in your product since they'll be the ones using it!



You want stakeholders to be happy, but it's your users that you really want to satisfy.

Interviewing Users

Interview 5-6 users [Nielsen 2000] and ask open-ended questions about the problem.

- Use followup questions to to make sure that you understand their goals and motivations.
- The outcome from interviews should be detailed interview notes.

Good interview questions:

- Describe the problem that you have.
- How would you accomplish X?
- Would X be useful to you? Why or why not?
- What do you like about this system? What do you dislike?
- How would you expect X to work?

Outcome: Personas

One issue with trying to understand users is that we always want to design from our own perspective. You don't want that - instead you want to be able to empathize with your users, and anticipate what *they* would want.

A **persona** is an archetypal description of a user of your product. Creating personas can help you step out of yourself... [1]t can help you to identify with the user you're designing for.

Types

1. **Goal-based persona:** examine their attitudes, their situation, and their goal in that moment. Consider design from the perspective of how they can reach their goals.
2. **Role-based persona:** focus on the role that the user plays in the organization, and design from the perspective of expectations of that role (somewhat goal-driven as well).
3. **Engaging personas:** model a complete personality that you can engage with, and "bounce ideas off".

Data to Capture

- **Personal info:** Name, job title, company, job description, family status, and more. Include any details that can help your team understand the persona better.
- **Demographic info:** Age, gender, income, education, location, and other background information.
- **Image:** Images speak louder than words. A vivid image, sketch, or cartoon can help your team picture your users clearly, and help you establish a consistent understanding of the target.
- **Goals & motivations:** You should also make it clear what your audience wants to get or to do with your product.

Jeff D. Instructor

BIO

Jeff is a faculty member in the Cheriton School of Computer Science. Previously, he worked in industry as a software developer, and engineering manager. He is passionate about teaching, and will evaluate any tool through the lens of "how it might make him a better teacher".



GOALS

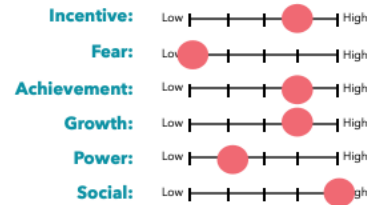
Find better ways to organize information. Improve efficiency.

FRUSTRATIONS

Poorly designed software.

Hates vendor lock-in. Ironically, Apple user.

MOTIVATIONS



QUOTES

"Using your computer should be pleasant, even *joyful*."

"Build something that will make you proud. If you wouldn't show it off to your parents, rewind and try again."

Age: 50

Occupation: Lecturer

Status: Married

Location: Waterloo

Archetype: Educator

Personality: Friendly;
Helpful.

DOMAIN AWARENESS

Self-proclaimed "productivity geek".
Has strong opinions in this domain.

TECH KNOWLEDGE

Expert in software development.
Broad general knowledge.

Extracting Requirements

Outcome: Affinity Diagrams

The outcome of your interviews will likely be a lot of qualitative data -- opinions, suggestions and ideas that the user expressed. The challenge is extracting ideas, themes from this narrative that you can use in your design.

We use thematic analysis to do this, and the mechanism that we will use is the creation of an **affinity diagram**.

Use an affinity diagram to expose critical ideas or themes from the user's ideas. (You can even do this across multiple users and look for which themes occur more frequently!).

To create an affinity diagram:

1. Record all notes or observations on individual cards or sticky notes
2. Look for patterns in notes or observations that are related and group them
3. Create a group for each pattern or theme
4. Give each theme or group a name
5. Create a statement of what you learned about each group (provide your analysis or key insight)

Outcome: User Stories

From your user interviews, you should have identified some basic tasks that you will need to address. Often, these will come in the form of stories or narratives from your users.

User-Stories are a description of activities that are performed with a system to perform a particular task. A user story is often simple text, and is intentionally describing the problem at a high level:

“As the HR manager, I want to create a screening quiz so that I can understand whether I want to send possible recruits to the functional manager.”

“As a user, I can indicate folders not to backup so that my backup drive isn’t filled up with things I don’t need saved”.

Once you’ve collected a number of user stories, you might arrange them into **User-Story Maps** to show their larger relationships.

User-Story Map: Mobile App Feature for Depositing Checks

1. Activities:

High-level tasks users can do in the digital product



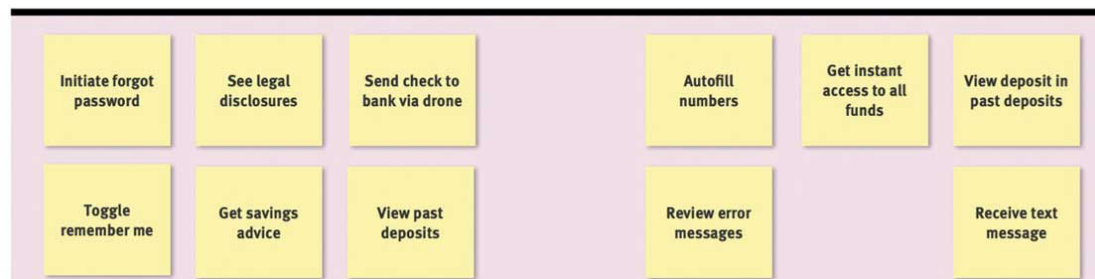
2. Steps:

Steps users go through to complete the activity above



3. Details:

Granular, discrete interactions to complete the step above

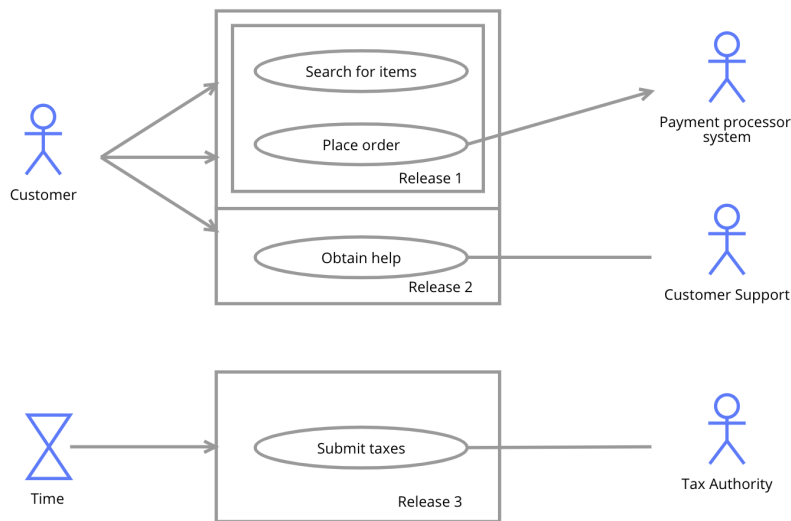


<https://www.nngroup.com/articles/user-story-mapping/>

Outcome: Use Cases (Optional)

A **use case** is similar to a user story, in that it describes a particular action or set of actions by a user. However, where a user story is high-level and leaves out a lot of details, a use case is much lower level and attempts to describe all of the required details.

Use cases are documented using UML templates. e.g. "Search for items", "Place order" are use cases.



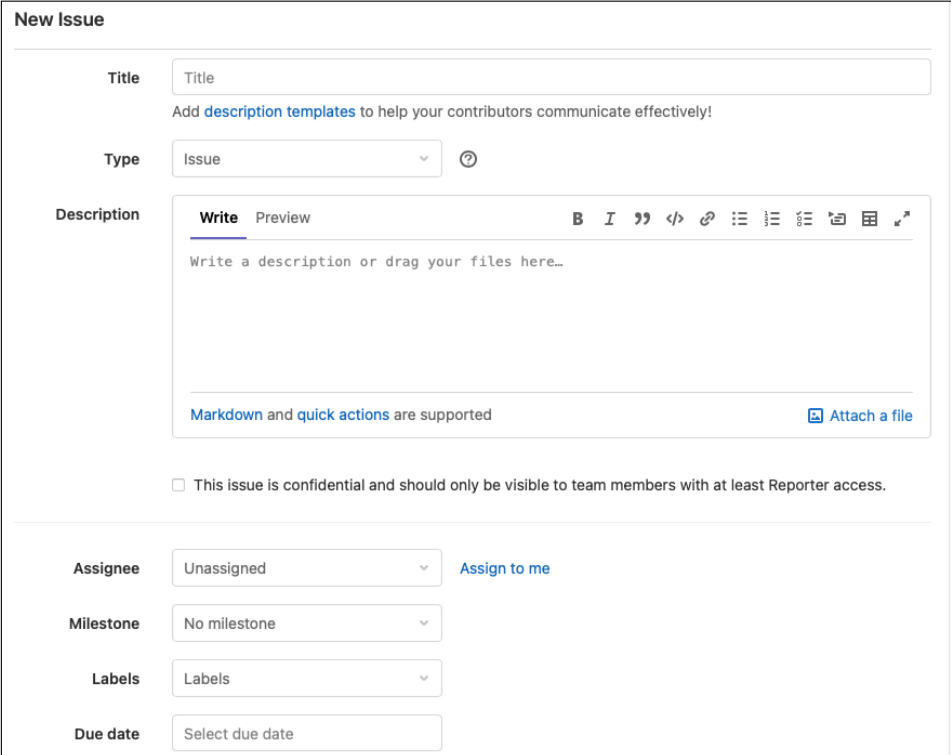
Name	The use case name. Typically the name is of the format <action>+<object>.
ID	A unique identifier.
Description	A brief sentence that states what the user wants to be able to do and what benefit they will derive.
Actors	The type of user who interacts with the system to accomplish the task. Actors are identified by role name.
Assumptions	Any statements about the system assumed to be true.
Frequency of use	How often this use case is executed.
Triggers	Actions executed by the an Actor to start the use case.
Precondition	Conditions that must be met before starting the use case.
Ordinary sequence	Steps that will be executed by this use case.
Exceptions	Alternate steps if errors.
Postcondition	Conditions that will be met after the use case.

Final Step: Product Backlog

Your final step should be to convert these requirements into actionable items.

- Log each item into your GitLab project as an Issue (better to have a custom type?)
- Don't assign anything yet!
 - Assignee: Unassigned
 - Milestone: No milestone
 - Due date: Unselected

These unassigned issues constitute your Product Backlog. You only schedule this during Sprint planning, and only if the team agrees to take it on.



The screenshot shows the 'New Issue' form in GitLab. It includes a 'Title' field with the placeholder 'Title', a 'Type' dropdown set to 'Issue', and a 'Description' field with a rich text editor. The description field has a 'Write' tab selected and a 'Preview' tab. Below the description field, there is a checkbox for 'This issue is confidential and should only be visible to team members with at least Reporter access.' At the bottom, there are fields for 'Assignee' (set to 'Unassigned'), 'Milestone' (set to 'No milestone'), 'Labels' (set to 'Labels'), and 'Due date' (set to 'Select due date').