

Rapid Ideation and Creative Problem Solving

Project Scope & Use Cases

Matt Franks & Lauren Serota



ac4d

Setting Project Scope

Think of project scope as the focus for your product.

Scope will determine key requirements, as well as who your primary and secondary users are.

Setting Project Scope

Think of project scope as the focus for your product.

Scope will determine key requirements, as well as who your primary and secondary users are.

What are some examples of tightly scoped products?

What are some examples of broadly scoped products?

Setting Project Scope

For this class, you will be designing a product based on your findings from IDSE 101.

Your product's primary component must be an application for the web.

It may have additional service components, however this class will focus on development of a web UI as the product's core offering.

What is scope creep?

Setting Project Scope

What is scope creep?

Scope creep refers to uncontrolled changes in a project's scope. This can occur when the scope of a project is not properly defined, documented, or controlled. It is generally considered a negative occurrence, and thus, should be avoided.

How can we avoid scope creep?

Use Cases

A use case is a description of steps or actions between a user (or "actor") and a software system which leads the user towards something useful (a goal). The user or actor might be a person or something more abstract, such as an external software system or manual process.

Use cases treat the system as a black box, and the interactions with the system, including system responses, are perceived as from outside the system. This is deliberate, as it forces the creator to focus on what the system must do, not how it is to be done, and avoids making assumptions about how the functionality will be accomplished.

Use Cases

A use case should:

- Describe what the system shall do for the actor to achieve a particular goal.
- Include no implementation-specific language (nothing about “how” it’s done).
- Be at the appropriate level of detail.
- Not include detail regarding user interfaces and screens. This is done in user-interface design, which references the use case and its business rules.

Use Cases

Developing use cases is an iterative process - one that will be edited and referenced through the entirety of the project.

Use Cases: How to do them

1. Identify your actors

All the peeps or other systems/entities/institutions that will be using your system

Example:

Online Payment Platform

Primary Actors:

Payer
Recipient

Secondary Actors:

Bank
CC Company

2. Identify your primary use cases (also known as “sunny day” or “happy path”)

Imagine all is well in the world, and your users are able to successfully complete the key tasks they’re trying to accomplish with your product.

What are those key tasks?

“Use the 80/20 rule -- if you write an exhaustive list of all possible use cases, typically 20% of the use cases will account for 80% of the activity. The other 80% of the use cases would support 20% of the activity.”

[-http://www.gatherspace.com/static/use_case_example.html#3](http://www.gatherspace.com/static/use_case_example.html#3)

Example:

Online Payment Platform

Payer:

- Create Account
- Manage Account
- Send Payment
- Check Payment Status

3. **Identify your edge cases** (also known as “rainy day”).

An edge case is a problem or situation that occurs only at an extreme (maximum or minimum) operating parameter. It can break an otherwise good experience.

Edge cases are very important to document, however **should not be designed around**. Identify them as you go along, and document them somewhere to reference later.

Example:

Online Payment Platform

Payer:

- Report False Charge
- Cancel a Payment
- Retrieve Lost Username or Password

4. Create a use case diagram

Use case diagrams are used to represent the functionality of your system from a top-down perspective, so at a glance the system's functionality is obvious, but all descriptions are at a very high level.

Use case diagrams have 4 components:

- actors interacting with the system or product
- the system or product itself
- the use cases (services, tasks) the system or product knows how to perform
- relationships between these components

4. Create a use case diagram

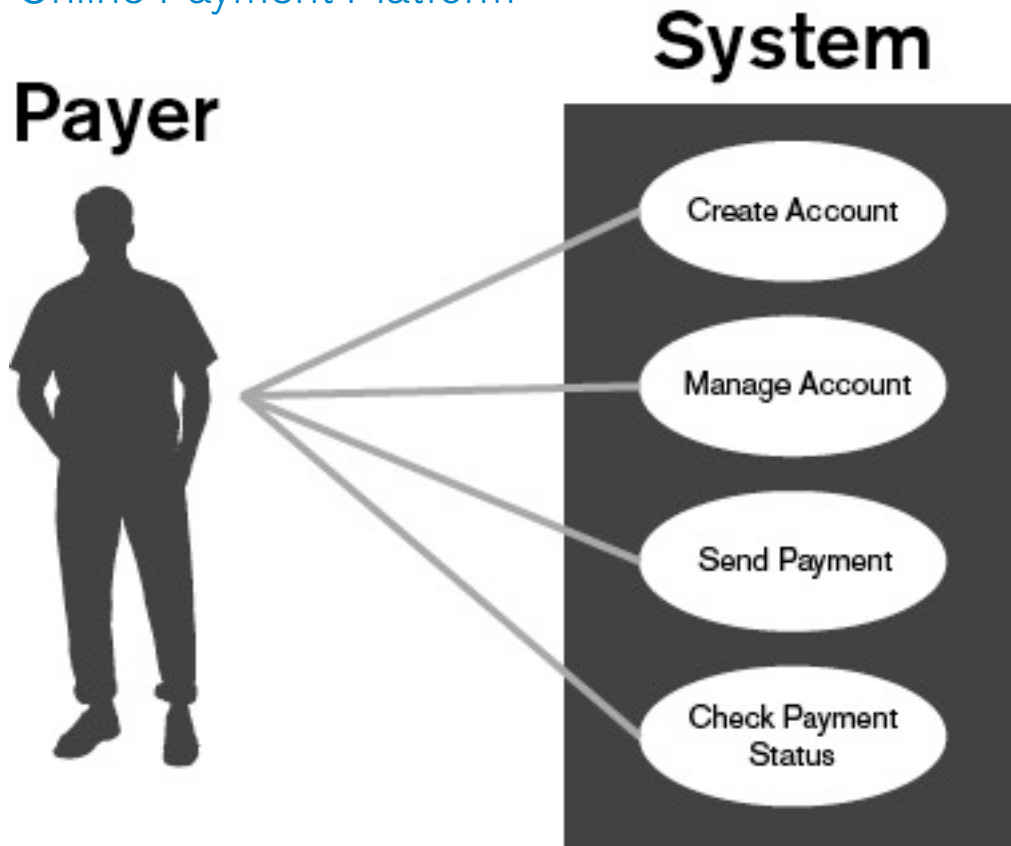
Use case diagrams only show interactions that happen between users and your system. They do not show interactions that happen between user of your system outside of it, or what happens in your system independent of the user.

They also do not show sequence of actions. Storyboards and flowcharts do those well - and we'll do those later in the class.

Use Cases: How to do them

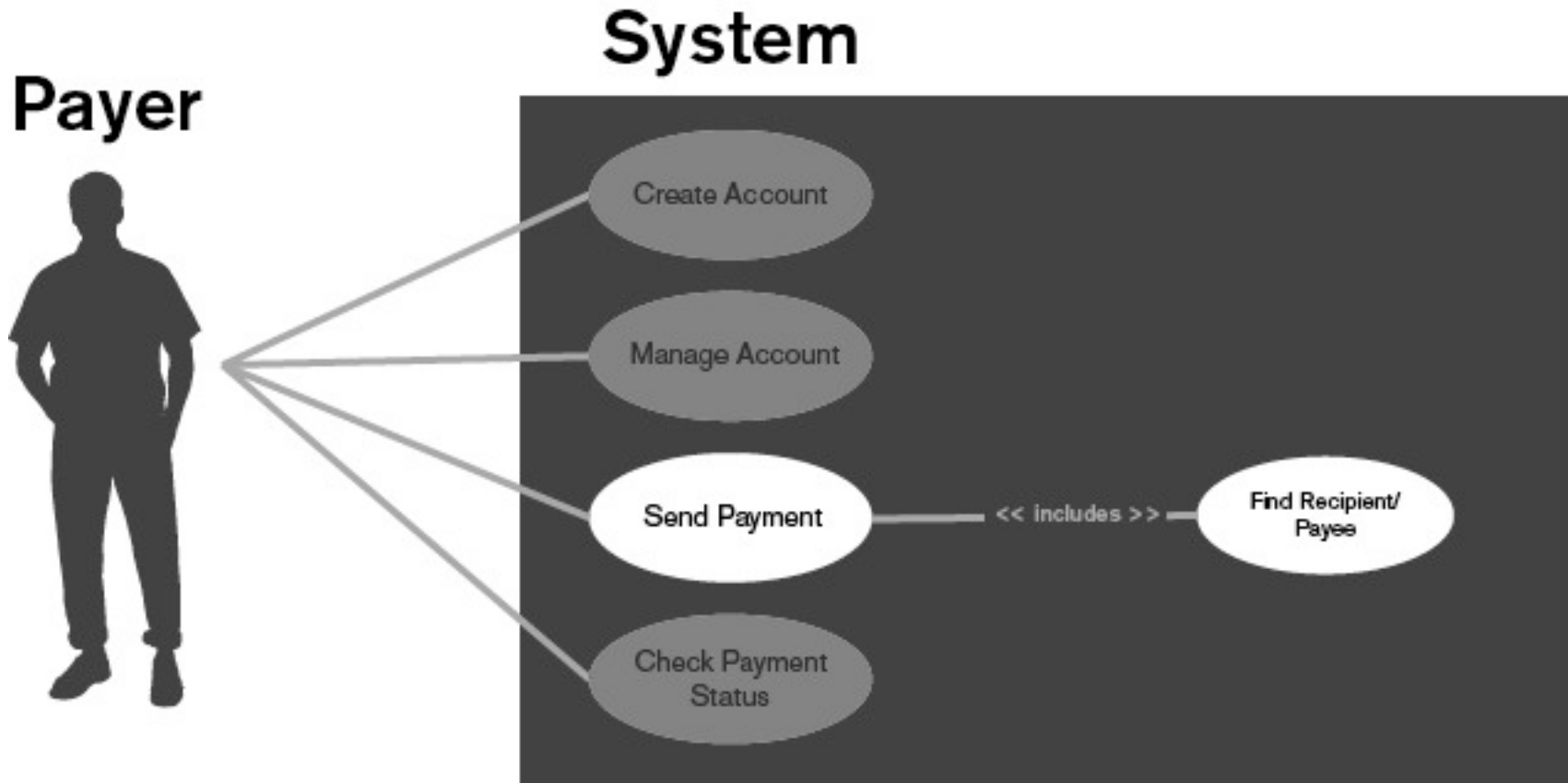
4. Create a use case diagram

Example:
Online Payment Platform



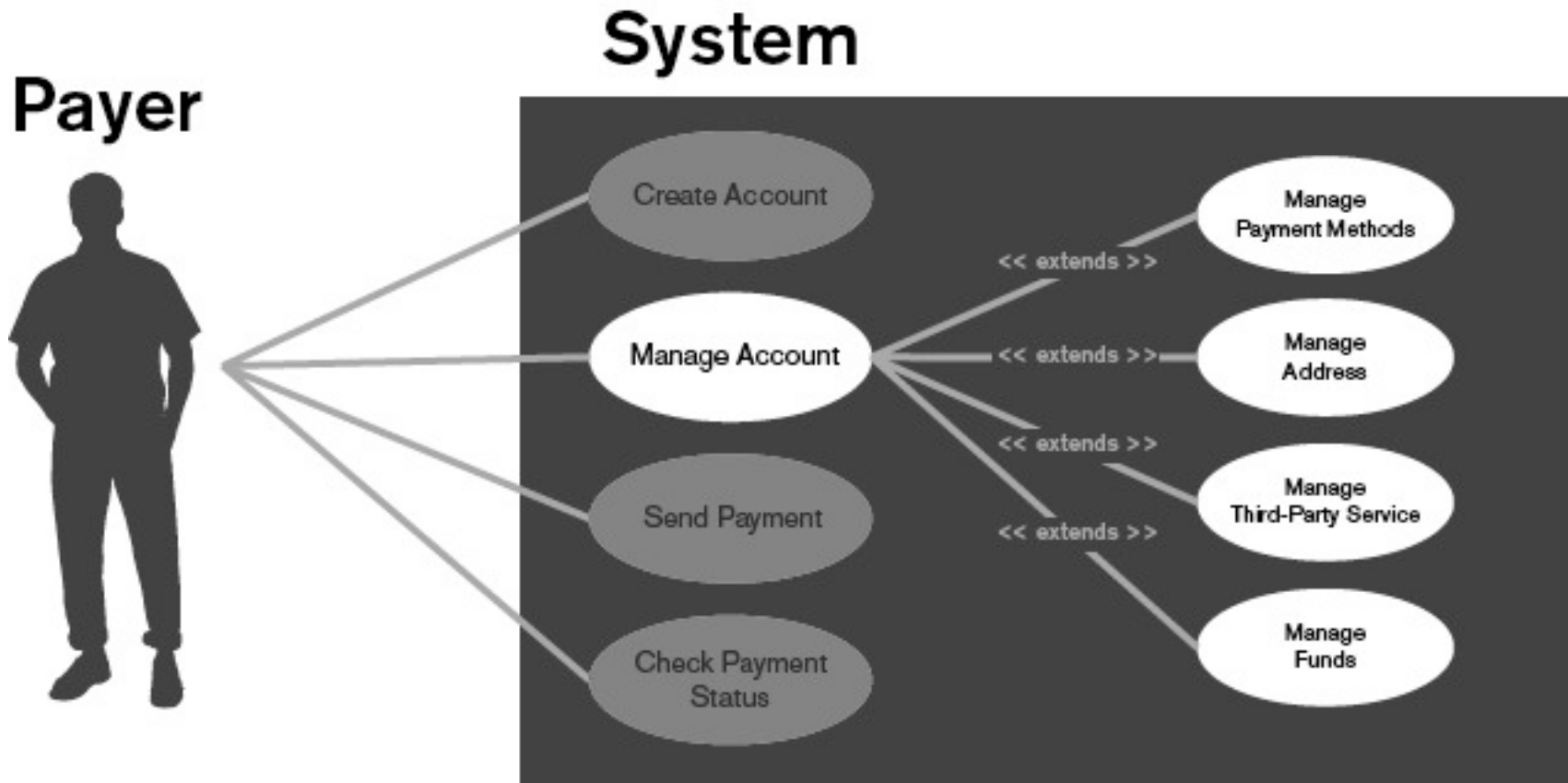
4. Create a use case diagram, adding detail

<< includes >> indicates that the base use case explicitly incorporates behavior from another use case.



4. Create a use case diagram, adding detail

<< extends >> indicates that the base use case can incorporate other use cases at certain points.



5. Write a description for each use case

Here's where the element of time comes more into play. Now that your use cases have been broken down further, you can begin to describe the sequence of events for each.

Online Payment Platform

Actor: Payer

P6

Name: Send Payment

The payer has decided to send a payment to an individual recipient. The payer locates the recipient within the system (includes: Find Recipient/Payee). The payer selects the appropriate recipient. The payer specifies the amount of money to send. They decide where they want the money to come from, choosing between their banks, credit cards and any money stored in the system. The payer enters all information into the system and the system verifies and validates all the recipient and payment details then submits payment to the recipient. The system provides feedback to the payer that their payment is complete, with notice of any delays or other important information.

Alternate 1:

.....

Use Cases: Formality

As you can imagine, more complicated systems require more complicated and numerous use cases. Numbering and indexing your use cases is always a good idea, as it allows you to quickly edit and scale your use cases to accommodate changes or growth to your design.

I recommend using a databasing tool (like Excel) to write your use cases. You can even keep your use case diagram on a separate sheet within the file!

Use Cases: Let's do one!

Our system is a DSLR Camera

Use Cases: Resources

[CMU Unified Modeling Language](#)

[Gatherspace](#)

[Agile Modeling](#)

[Top Ten Use Case Mistakes](#)

Assignment 1: Use Cases & Use Case Diagram

Due Tuesday 11/8

Go through the steps we just went through and create:

- 1. Use Cases** These use cases will represent the aggregate of your iterations and refinements. You will deliver at least 3 use cases (see above), with each primary use case containing at least 1 alternate.
(60 points)
- 2. Use Case Diagram** Use case diagram(s) that describe the functions your product supports to one level of detail. You will turn in one high-level diagram, and one per use case showing additional detail where applicable.
(40 points)

ac4d