CSE 403: Software Engineering, Spring 2015

courses.cs.washington.edu/courses/cse403/15sp/

Use Cases

Emina Torlak
emina@cs.washington.edu

Outline

- What is a use case?
- Terminology
- Styles of use cases
- Steps for creating a use case

What is a use case?

A written description of the user's interaction with the software product to accomplish a goal.

- It is an **example behavior** of the system.
- 3-9 clearly written steps lead to a "main success scenario."
- Written from actor's point of view, not the system's.

What is a use case?

A written description of the user's interaction with the software product to accomplish a goal.

- It is an **example behavior** of the system.
- 3-9 clearly written steps lead to a "main success scenario."
- Written from actor's point of view, not the system's.

Use cases capture **functional requirements** of a system.

Benefits of use cases

- Establish an understanding between the customer and the system developers of the requirements (**success scenarios**)
- Alert developers of problematic situations, error cases to test (**extension scenarios**)
- Capture a level of functionality to plan around (**list of goals**)

Qualities of a good use case

Qualities of a good use case

- Focuses on interaction
    - Starts with a request from an actor to the system.
    - Ends with the production of all the answers to the request.

Qualities of a good use case

- Focuses on interaction
    - Starts with a request from an actor to the system.
    - Ends with the production of all the answers to the request.
- Focuses on essential behaviors, from the actor's point of view
    - Does not describe internal system activities.
    - Does not describe the GUI in detail.

Qualities of a good use case

- Focuses on interaction
    - Starts with a request from an actor to the system.
    - Ends with the production of all the answers to the request.
- Focuses on essential behaviors, from the actor's point of view
    - Does not describe internal system activities.
    - Does not describe the GUI in detail.
- Concise, clear, accessible to non-programmers
    - Easy to read.
    - Summary fits on a page.
    - Main success scenario and extensions.

Use cases versus internal features

Use cases versus internal features

Use cases

- call someone
- receive a call
- send a message
- memorize a number

Point of view: user

Use cases versus internal features

Use cases

- call someone
- receive a call
- send a message
- memorize a number

Point of view: user

Internal functions

- transmit / receive data
- energy (battery)
- user I/O (display)
- phone-book mgmt.

Point of view: developer

Use cases and requirements

- Special deals may not run longer than 6 months.
- Customers only become preferred after 1 year.
- A customer has one and only one sales contact.
- Database response time is less than 2 seconds.
- Web site uptime requirement is 99.8%.
- Number of simultaneous users will be 200 max.

Which of these requirements should be represented directly in a use case?

Use cases and requirements

- Special deals may not run longer than 6 months.
- Customers only become preferred after 1 year.
- A customer has one and only one sales contact.
- Database response time is less than 2 seconds.
- Web site uptime requirement is 99.8%.
- Number of simultaneous users will be 200 max.

Which of these requirements should be represented directly in a use case?

None! These are properties but not user-driven *behaviors* of the system, so the use cases wouldn't mention them.

Terminology

Terminology

- **Actor**: an entity that acts on the system
    - Person, external hardware (like a timer), or another system.

Terminology

- **Actor**: an entity that acts on the system
    - Person, external hardware (like a timer), or another system.
- **Primary actor:** initiates interaction to accomplish a goal

Terminology

- **Actor**: an entity that acts on the system
  - Person, external hardware (like a timer), or another system.
- **Primary actor:** initiates interaction to accomplish a goal
- **Goal**: desired outcome of the primary actor

Terminology

- **Actor**: an entity that acts on the system
    - Person, external hardware (like a timer), or another system.
- **Primary actor:** initiates interaction to accomplish a goal
- **Goal**: desired outcome of the primary actor
- **Level**
    - User goals (accomplished in one sitting)
    - Summary goals (accomplished in multiple sittings)
    - Subfunction goals (required to carry out user goals)

Terminology

- **Actor**: an entity that acts on the system
    - Person, external hardware (like a timer), or another system.
- **Primary actor:** initiates interaction to accomplish a goal
- **Goal**: desired outcome of the primary actor
- **Level**
    - User goals (accomplished in one sitting)
    - Summary goals (accomplished in multiple sittings)
    - Subfunction goals (required to carry out user goals)

Use cases are always initiated by actors and describe the **flow of events** that these actors are involved in.

Styles of use cases

Use case diagram

- in UML, the Unified Modeling Language

Informal use case

- a short paragraph

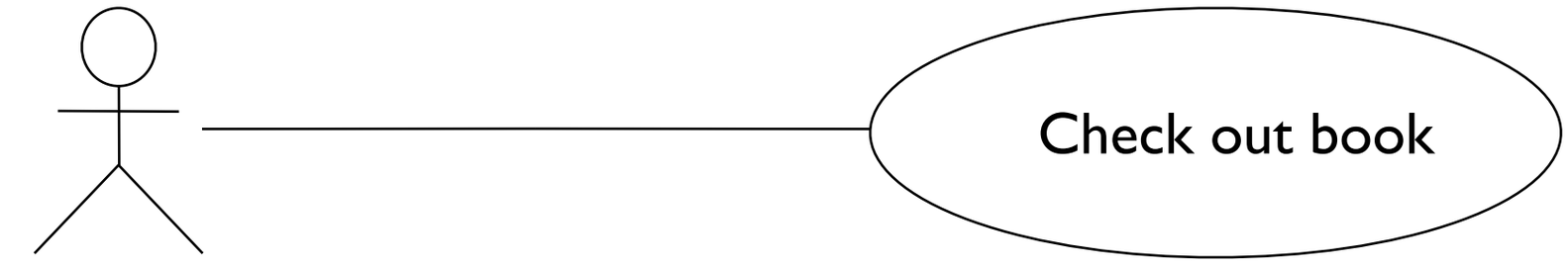
Formal use case

- a multi-part structured description

Use case diagram

The overall list of the system's use cases can be drawn as high-level diagrams, with:

- **actors as stick-men**, with their names (nouns)
- **use cases as ellipses**, with their names (verbs)
- **line associations**, connecting an actor to a use case in which that actor participates
- use cases can be connected to other cases that they use / rely on

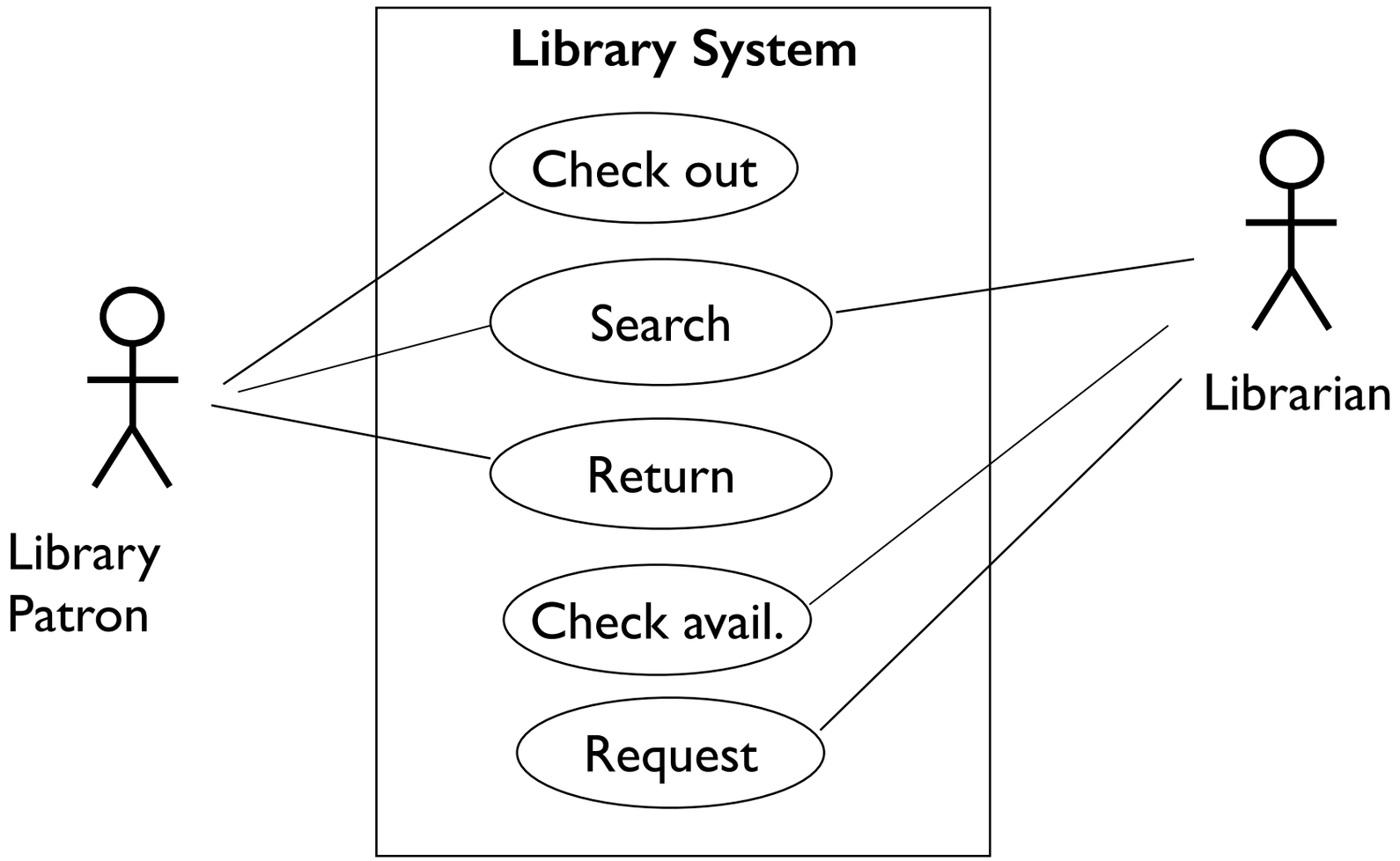


Actor-goal lists: function content of the system

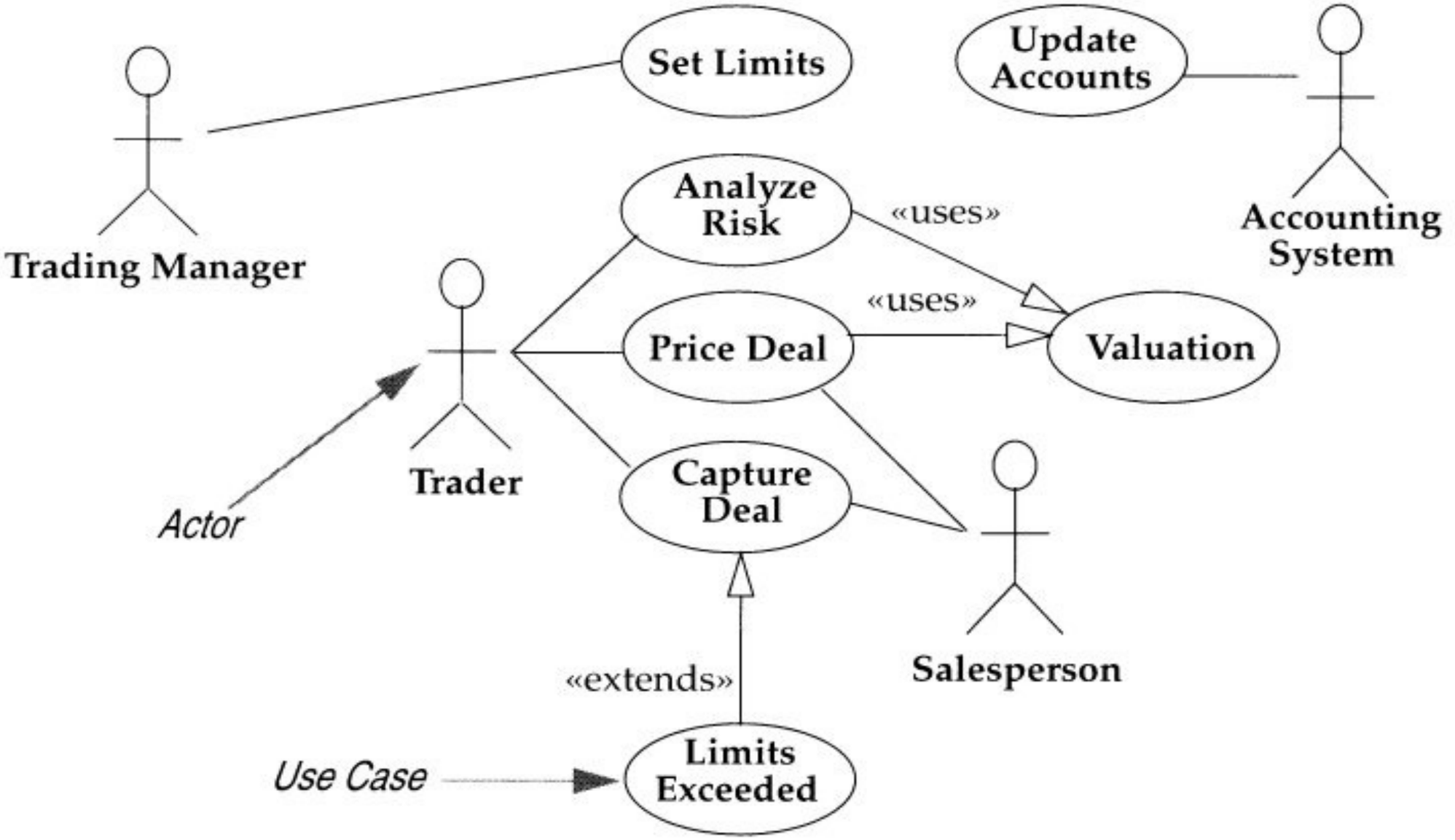
Actor	Goal
Library Patron	Search for a book
	Check out a book
	Return a book
Librarian	Search for a book
	Check availability
	Request a book from another library

It can be useful to create a list or table of primary actors and their "goals" (use cases they start). The diagram will then capture this material.

Use case summary diagrams



Use case summary diagrams



What is an extension?

What is an extension?

- A possible branch in a use case scenario, often triggered by an error or failure in the process.
    - Useful for finding edge cases that need to be handled and tested.

What is an extension?

- A possible branch in a use case scenario, often triggered by an error or failure in the process.
    - Useful for finding edge cases that need to be handled and tested.
- Do
    - Think about how every step of the use case could fail.
    - Give a plausible response to each extension from the system.
    - Response should either jump to another step of the case, or end it.

What is an extension?

- A possible branch in a use case scenario, often triggered by an error or failure in the process.
    - Useful for finding edge cases that need to be handled and tested.
- Do
    - Think about how every step of the use case could fail.
    - Give a plausible response to each extension from the system.
    - Response should either jump to another step of the case, or end it.
- Don’t
    - List things outside the use case ("User's power goes out").
    - Make unreasonable assumptions ("DB will never fail").
    - List a remedy that your system can't actually implement.

Informal use case

Patron loses a book

- The **library patron** reports to the librarian that she has lost a book. The **librarian** prints out the library record and asks patron to speak with the head librarian, who will arrange for the patron to pay a fee. The **system** will be updated to reflect lost book, and patron's record is updated as well. The **head librarian** may authorize purchase of a replacement book.

Informal use case is written as a paragraph describing the scenario / interaction.

Informal use case with structured text

I

- I.A
    - I.A.ii
        - I.A.ii.3
        - I.A.ii.3.q

You will probably use something in this general style.

Although not ideal, it is almost always better than unstructured natural language.

Formal use case

Goal	Patron wishes to reserve a book using the online catalog
Primary actor	Patron
Scope	Library system
Level	User
Precondition	Patron is at the login screen
Success end	Book is reserved
Failure end condition	Book is not reserved
Trigger	Patron logs into system

Parts that make up a formal use case (continued on the next slide).

Formal use case (continued)

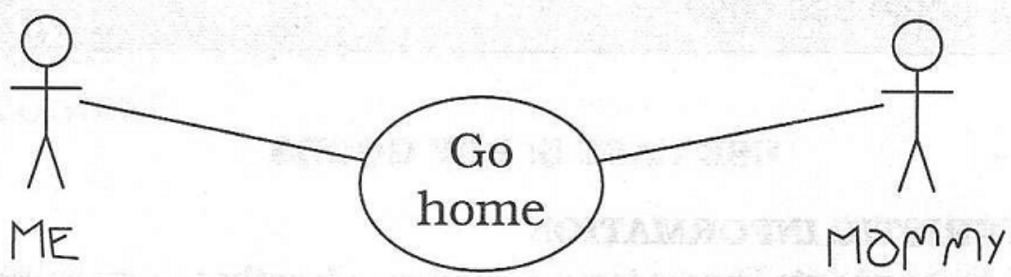
Main success scenario	1. Patron enters account and password 2. System verifies and logs patron in 3. System presents catalog with search screen 4. Patron enters book title 5. System finds match and presents location choices 6. Patron selects location and reserves book 7. System confirms reservation and re-presents catalog
Extensions (error scenarios)	2a. Password is incorrect 2a.1 System returns patron to login screen 2a.2 Patron backs out or tries again 5a. System cannot find book 5a.1 …
Variations (alternative scenarios)	4. Patron enters author or subject

What notation is good?

There are standard templates for requirements documents, diagrams, etc. with specific rules. Is this a good thing? Should we use these standards or make up our own?

- Standards are helpful as a template or starting point.
- But don't be a slave to formal rules or use a model/ scheme that doesn't fit your project's needs.

Figure 7.3. UML use cases—so simple a child could do it!



Steps for creating a use case

1. Identify actors and goals
2. Write the main success scenario
3. List the failure extensions
4. List the variations

Alistair Cockburn

1. Identify actors and goals

- What computers, subsystems and people will drive our system? (actors)
- What does each actor need our system to do? (goals)
- Exercise: actors/goals for your projects

Come up with 4 use case names for your software, draw a UML use case summary diagram for it, and write out one complete (formal) use case.

2. Write the main success scenario

- Main success scenario is the preferred "happy path”
    - easiest to read and understand
    - everything else is a complication on this

- Capture each actor's intent and responsibility, from trigger to goal delivery
    - say what information passes between them
    - number each line

3. List the failure extensions

- Usually, almost every step can fail (bad credit, out of stock)
    - Note the failure condition separately, after the main success scenario
- Describe failure-handling
    - recoverable: back to main course (low stock + reduce quantity)
    - non-recoverable: fails (out of stock, or not a valued customer)
    - each scenario goes from trigger to completion
- Label with step number and letter:
    - 5a failure condition
        - 5a.1 use case continued with failure scenario
        - 5a.2 continued

Exercise: describe one failure extension for your project's use case.

4. List the variations

- Many steps can have alternative behaviors or scenarios
- Label with step number and alternative
    - 5’. Alternative 1 for step 5
    - 5”. Alternative 2 for step 5

Pulling it all together: how much is enough?

You have to find a balance

- comprehensible vs. detailed
- graphics vs. explicit wording and tables
- short and timely vs. complete and late

Your balance may differ with each customer depending on your relationship and flexibility

Summary

- Uses case describe example system behaviors (contracts) from the user's point of view.
- Can be diagrams, informal paragraphs, formal use cases.
- 4 steps to create use cases.