
Software Design

Models, Tools & Processes

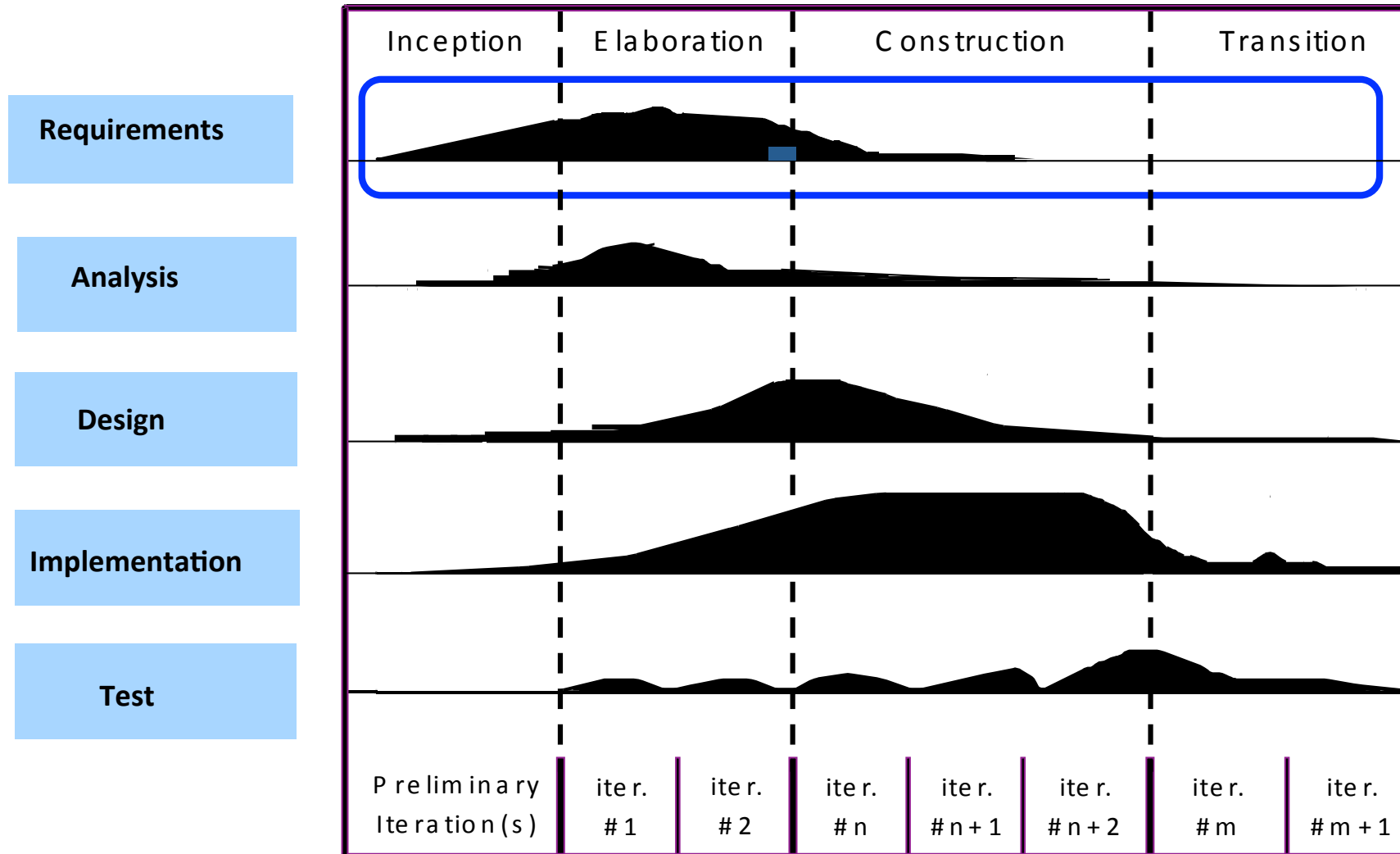
Lecture 2: Inception Phase

Cecilia Mascolo

Inception Phase

- This is the phase when most of the “system requirements” are identified.
 - Discover and reach agreement on what the system has to do with the customer.
 - Create a high level specification of what the system should do.

Development Process



What are requirements?

1. “A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
3. A documented representation of a condition or capability as in 1 or 2.”

[IEEE Glossary]

Why are requirements important?

- “The hardest single part of building a software system is **deciding precisely what to build**. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, & to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.”

[Fred Brooks in “The Mythical Man Month”]

The economics of requirements

Relative cost to fix an error [Boehm 1980]

Phase in which found	Cost ratio
requirements	1
design	3-6
coding	10
development testing	15-40
acceptance testing	30-70
operation	40-1000

... & these figures are considered conservative!

Example

- Problem statement

“You have been contracted to develop a computer system for a university library. The library currently uses a 1960s program, written in an obsolete language, for some simple bookkeeping tasks, & a card index, for user browsing. You are asked to build an interactive system which handles both of these aspects online.”

- Your task is to determine a baseline set of information you can use to start designing your system

Writing a requirements document

- Requirements document, requirements specification document, requirements specification etc.
- Statement of organised user/system requirements - generally written in natural language
- We need this to start software development with USDP

<unique id> The <system name> **shall** <function to perform>

“23. The Library System **shall** validate the library card”

Note that this is **not** a formal deliverable in USDP

Functional & non-functional requirements

- **Functional** - what the system should do
 - e.g. the library system *shall* provide a facility for identifying the identity of a library user
 - e.g. the library system *shall* provide a reminder when a book is overdue
- **Non-functional** - a constraint on how the functional requirements can be implemented
 - e.g. the library system *shall* authenticate a library customer in five seconds or less
 - e.g. the library system *shall* communicate with borrowers using Email

Requirements prioritisation

MoSCoW criteria

- **M:** Must have - mandatory requirements that are fundamental to the system
- **S:** Should have - important requirements that could be omitted
- **C:** Could have - optional requirements
- **W:** Want to have - these requirements really can wait (i.e. bells & whistles)

Exercise



-
- Determine a set of functional requirements for the library system
 - Consider the problem statement
 - Interview each other in your group about how you use the library & consider what a system would need to do to support this
 - What you have to do
 - Express the functional requirements as *shall* statements with a unique ID for traceability
 - Group your requirements into sensible sets (e.g. user interface, borrowing, browsing)
 - Prioritise your requirements using MoSCoW

Example format

ID	Functional requirements	Priority
<i>Collection</i>		
1	the system shall ...	M
2	the ...	
<i>Borrowing</i>		
3		
<i>Browsing</i>		
4		
<i>Membership</i>		
5		
<i>User interface</i>		
6		

Exercise



-
- Now consider the non-functional requirements for the library system
 - Add a section to your document for non-functional requirements, & consider aspects like
 - Capacity
 - Availability
 - Reliability
 - Performance
 - Security
 - Compliance to standards
 - Again, give each a unique ID & a priority

Maintaining a project glossary

- This is another document to start at the beginning of a project & maintain throughout its duration
- The glossary captures the language of the business domain & project - it helps to provide a shared vocabulary & resolve some misunderstandings

Key term	Definition
Term 1	Definition 1

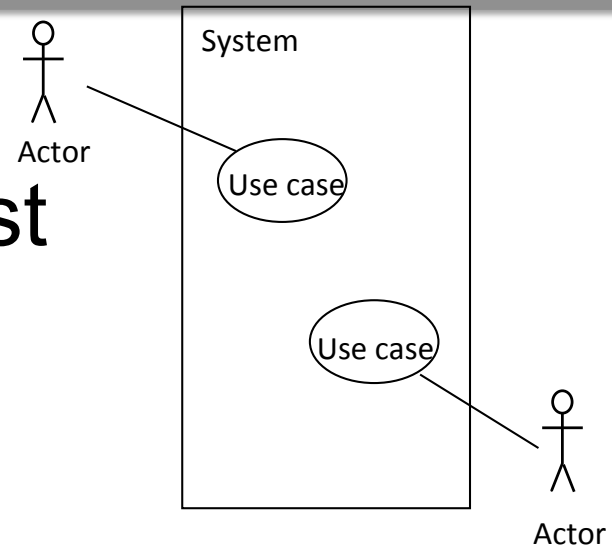
Exercise



-
- Try to build a list of words that are important in the domain of the library and briefly describe them
 - Nouns
 - Verbs

Use Case Modelling

- The use cases are the first step of the UML development process
- From the requirement document
- Use cases help in specifying what the system does with respect to the user



What is use case modelling?

- Basis of a user-oriented approach to system development
 - Identify the users of the system (actors)
 - Identify the tasks they must undertake with the system (use cases) & prioritise
 - Relate users & tasks (relationships)
- Use case models therefore contain actors, use cases & relationships
- Use case modelling is considered a form of requirements engineering, as it complements more traditional approaches

Helps identify the system boundary

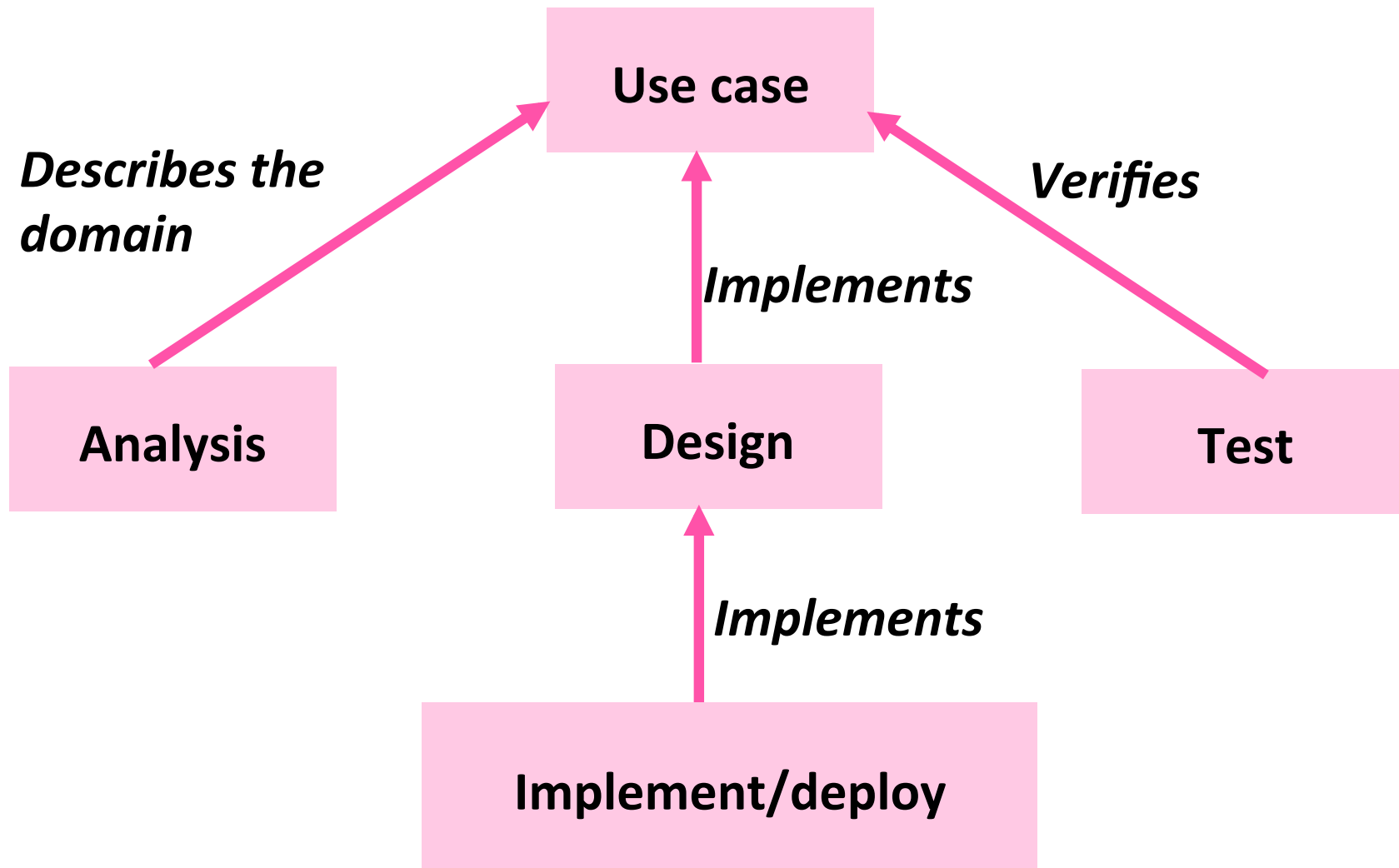
Establishing traceability

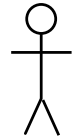
- Use case models document requirements in a complementary way to traditional requirements documents
- Items in these requirements documents should be linked to items in the use case models to ensure coverage, to help assess completeness of the requirements & to ensure consistency
- Establishing this link enables *requirements traceability*
- This is essential for change management & is typically tool supported

Requirement Tracing

	UC1	UC2	UC3
R1	x	x	
R2		x	
R3	x		x

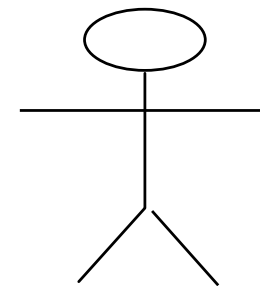
Use case driven



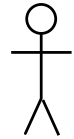


What are actors?

- Who or what uses the system
 - An *actor* is anything that interacts directly with the system (e.g. a person, a system)
 - An actor is a user of the system in a particular role
- An actor is generally external to a system, though the system may hold an internal representation of the actor (e.g. BookBorrower)
- Depicted as a stick person
- Actors trigger use cases

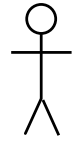


BookBorrower



How to find actors

- Observe the direct users of the system - those people or systems responsible for its installation, use or maintenance (both who & what)
 - What roles do these users play in the interaction?
 - Who provides information to the system?
 - Who receives information from the system?
- Same physical person may play the role of a number of actors; many people may play the same role so act as the same actor
- Becomes clearer as use cases are developed



Describing actors

- Describe each actor clearly & precisely in a few lines of English
 - Short name
 - Short description (semantics)
- Actors may have attributes (not so used)

BookBorrower

this actor represents someone that makes use of the library for borrowing books

Exercise



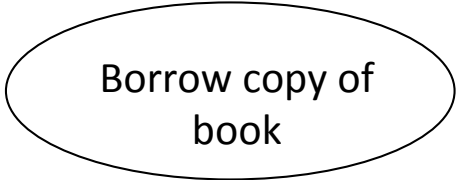
- Take the existing requirements document for the library system & identify all the actors that interact with the system
- Remember to specify the actors as roles
- For each actor, write down the actor name & provide a brief textual description describing the semantics of the actor

Actor	Semantics
Name 1	Description 1

What are use cases?



- Things actors do with the system
 - A task which an actor needs to perform with the help of the system (e.g. Borrow copy of book)
 - A specific kind of system use - a “case of use”
- Describes the behaviour of a system from a user’s standpoint by using actions & reactions (design independent)
- Represented as ellipses, internal to the system
- Triggered by an actor



Borrow copy of
book

How to find use cases



- Start with the list of actors & consider
 - What they need from the system (i.e. what use cases there are which have value for them)
 - Any other interactions they expect to have with the system (i.e. which use cases they might take part in for someone else's benefit)
- How do you know what **is** a use case?
 - Estimate *frequency* of use, examine *difference* between cases, distinguish between '*basic*' & '*alternative*' courses of events & create new use cases where necessary
- Leads to the identification of new actors...

Describing use cases

- Shown as a named ellipse representing the kind of task that has to be done with support from the system under development
- Semantics detailed in text - third-person, active-voice

Borrow copy of book

A BookBorrower presents a book. The system checks that the potential borrower is a member of the library, & that s/he does not already have the maximum number of books on loan. This maximum is six unless the member is a staff member, in which case it is twelve. If both checks succeed, the system records that this library member has this copy of the book on loan. Otherwise it refuses the loan.

Exercise

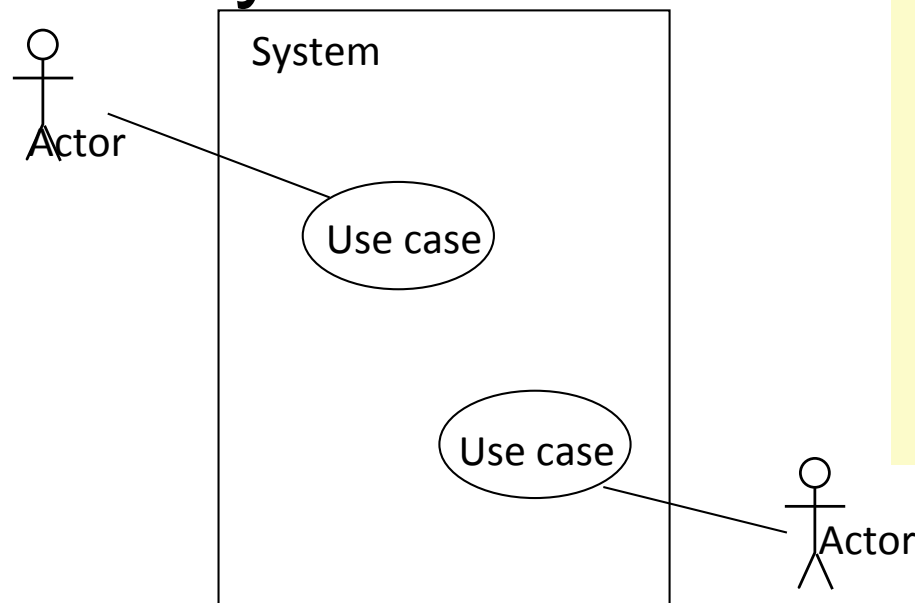


- Take the existing requirements document for the library system & your list of actors, then identify all the use cases for the system
- Remember to specify the use cases as active tasks
- For each use case, write down the use case name & provide a brief textual description describing the semantics of the use case

Use case	Semantics
Name 1	Description 1

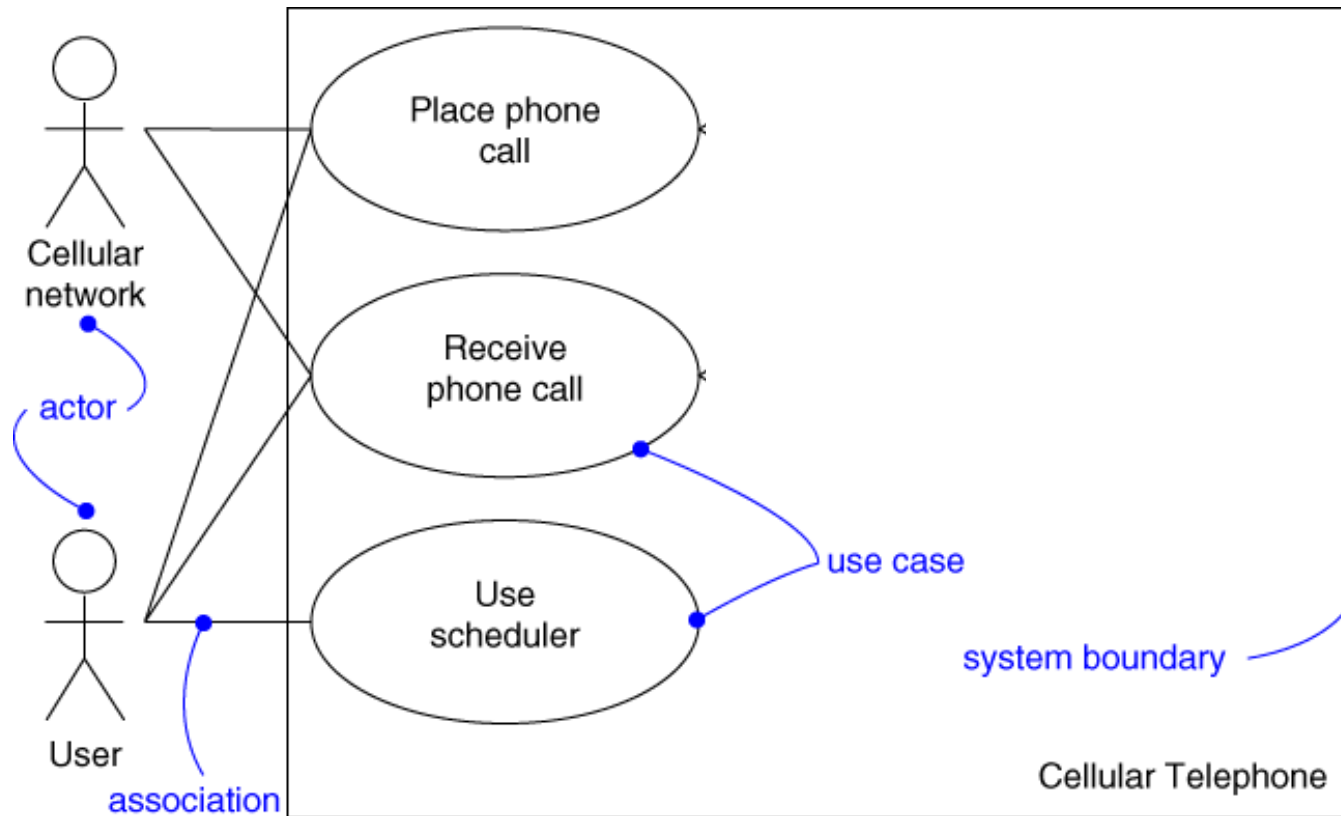
How to construct a use case diagram

- Characterises the *behaviour of whole system* (i.e. shows all the use cases at the top level & their actors)
- Helps to visualise context & boundary of the system



We are now adding communication relationships between the actors & the use cases (i.e. identifying beneficiaries)

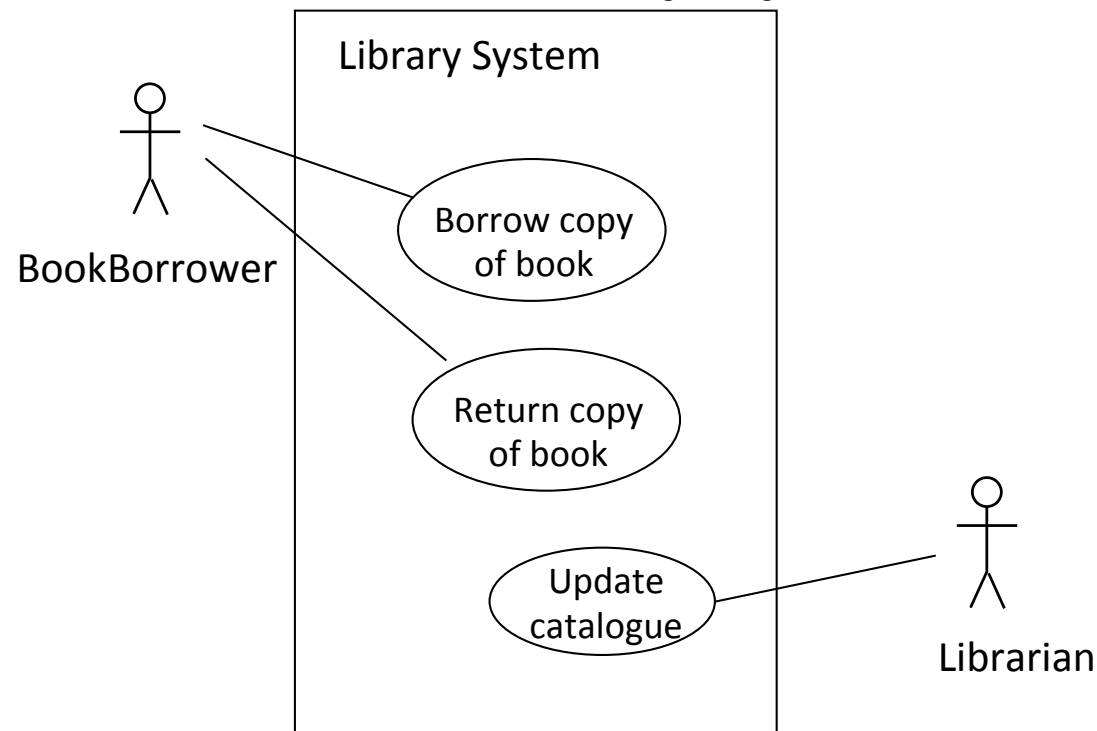
Example use case diagram



Exercise



- Use the UML notation for both actors & use cases to create a use case diagram that indicates the relationships between the actors & the use cases in our library system



Detailing a use case

- This involves writing a specification for the use case
- Requires an outline use case model and the additional requirements documents
- There are some good practice guides
 - **Preconditions:** the system state before the use case can begin (i.e. things that must be true)
 - **Flow of events:** the steps in the use case (i.e. something does some action)
 - **Postconditions:** the system state after the use case has completed (i.e. things that must be true)

Example specification

Borrow copy of book

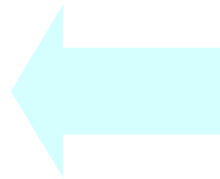
Preconditions

1. The BookBorrower is a member of the library
2. The BookBorrower had not got more than their permitted number of Books on loan

Flow of events

1. The use case starts when the BookBorrower attempts to borrow the copy of the book
2. The librarian checks it is ok to borrow the book
3. If

.....



Indicates an alternative path of action

Postconditions

1. The system has updated the number of Books the BookBorrower has on loan

Summary

- Use case modelling sets out all the actors, use cases & the relationships between them, setting the boundaries of the system to be built
- An actor is a role that interacts directly with the system by exchanging information
- A use case is a coherent unit of functionality that the system can perform by interacting with outside actors
- A use case diagram captures all the top-level functionality of the system as seen by its users (i.e. the key use cases)