Ingegneria del Software

Corso di Laurea in Informatica per il Management

UML: Use case

Davide Rossi
Dipartimento di Informatica
Università di Bologna

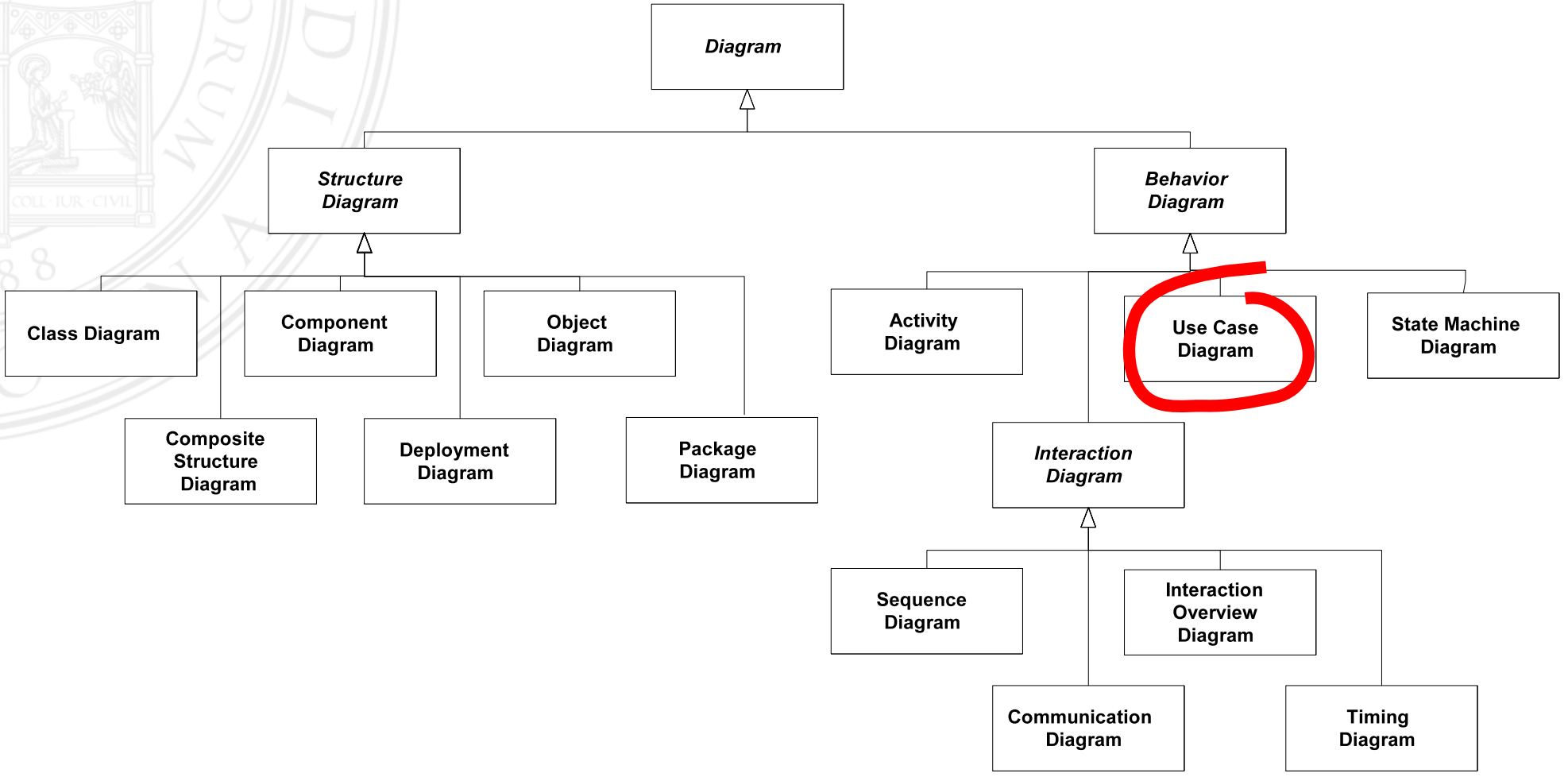


Diagram
Structure Diagram
Behavior Diagram
Class Diagram
Component Diagram
Object Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Interaction Diagram
Sequence Diagram
Interaction Overview Diagram
Communication Diagram
Timing Diagram

Use case diagrams

Use case diagrams are behavior diagrams used to describe a set of actions (use cases) that a system (*subject*) should or can perform in collaboration with one or more **external** users of the system (*actors*). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

A use case is not a diagram element

A use case is a list of actions or event steps typically defining the **interactions** between a role (or ***actor***) and a **system** to achieve a **goal**

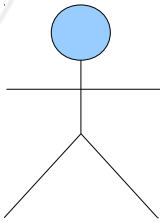
Use of use case diagrams

Use case diagrams are used to specify:

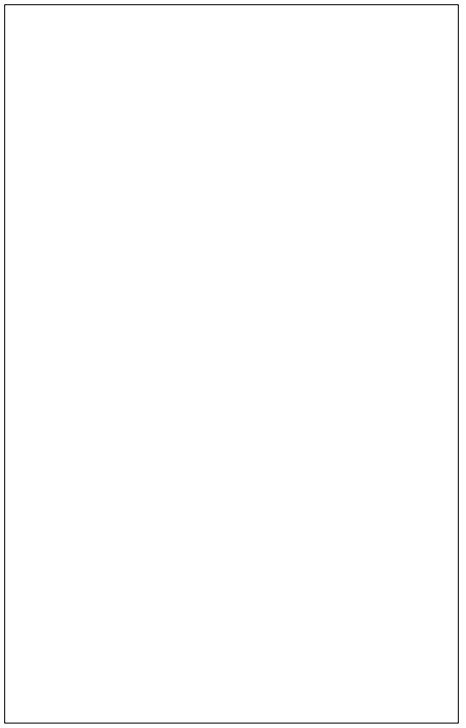
- (external) requirements on a subject, required usages of a system - to capture what a system under construction is supposed to do;
- the functionality offered by a subject – what system can do;
- requirements the specified subject poses on its environment - by defining how environment should interact with the subject so that it will be able to perform its services.

UC elements

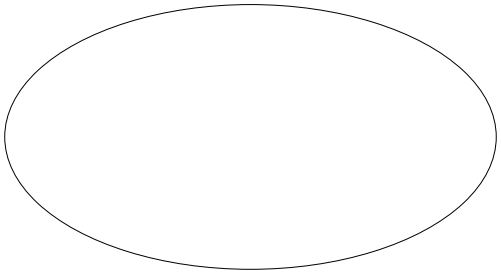
Actor



Actor



System, subject



Use case

UC: actor

In UML an ***actor*** is behaviored classifier which specifies a role played by an **external entity** that interacts with the **subject** (e.g., by exchanging signals and data), a human user of the designed system, some other system or hardware using services of the subject.

The standard UML notation for an actor is the "stick man" icon with the name of the actor above or below of the icon. Custom icons that convey the kind of actor may also be used to denote an actor, such as using a separate icon(s) for non-human actors.

UC: subject

The ***subject*** is the **system under analysis** or design to which a set of use cases apply. The subject could be a business, software system, physical system, or a smaller subsystem having some behavior. In UML terms, subject is a use case classifier playing the "subject" role.

Subject is presented by a rectangle with subject name in upper corner with the applicable use cases inside the rectangle and actors - outside of the system boundaries.

UC: use case

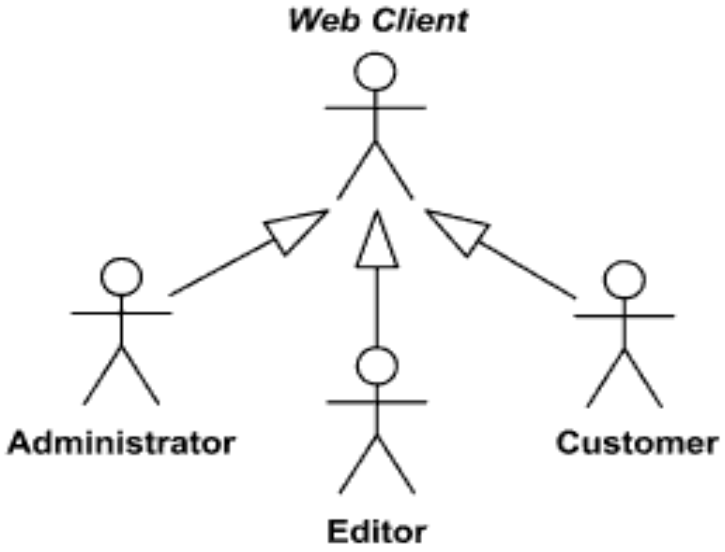
In UML a ***use case*** is a behaviored classifier which specifies the behavior of a subject by describing a set of **sequences of actions** performed by the system to yield an **observable result** of some value to one or more actors or other stakeholders of the system. In other words, each use case describes a unit of complete and useful functionality that the subject provides to its users.

Use case is usually shown as an ellipse containing the name of the use case.

Relationships Between Actors

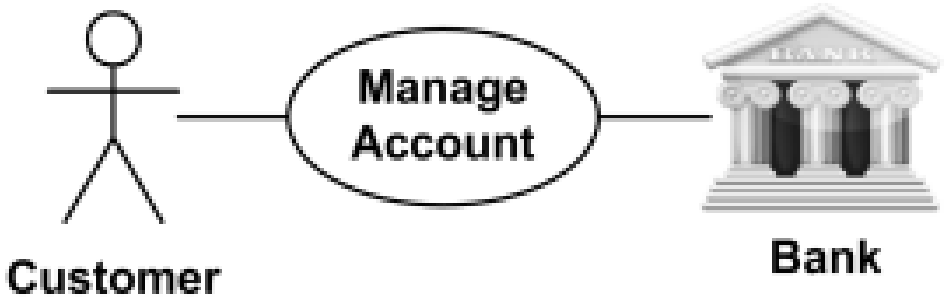
We can define abstract or concrete actors and specialize them using generalization relationship.

Generalization between actors is rendered as a solid directed line with a large arrowhead (same as for generalization between classes).



Associations Between Actors and Use Cases

Each use case specifies a unit of useful functionality that the subject provides to actors. This functionality should be initiated by an actor. Actors may be connected to use cases only by binary association relationship.



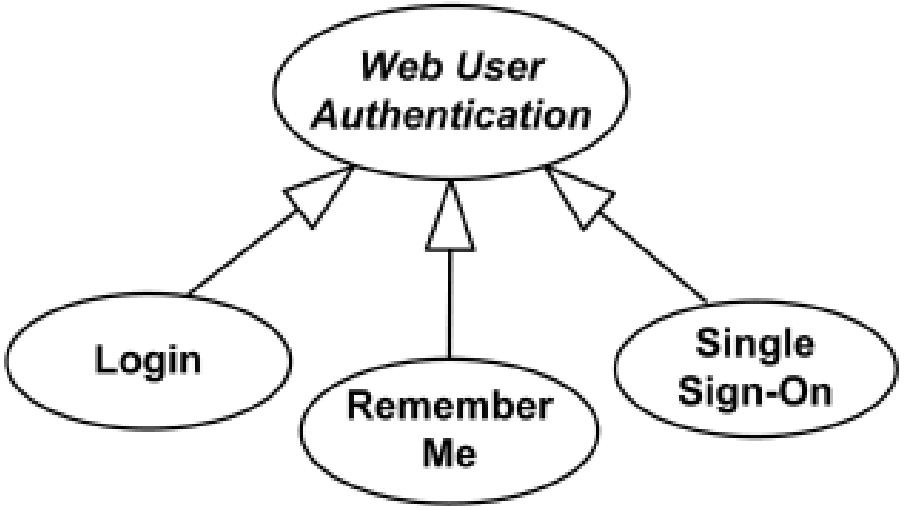
Relationships Between Use Cases

Use cases could be organized using the following relationships:

- generalization
- extend
- include
- (association)

Generalization Between Use Cases

Generalization is shown as a solid directed line with a large hollow triangle arrowhead, the same as for generalization between classifiers, directed from the more specific use case to the general use case.



Extend Relationship

Extend is a directed relationship that specifies how and when the behavior defined in a supplementary *extending use case* **can be** inserted into the behavior defined in the extended use case.

The extension takes place at one or more extension points defined in the extended use case.

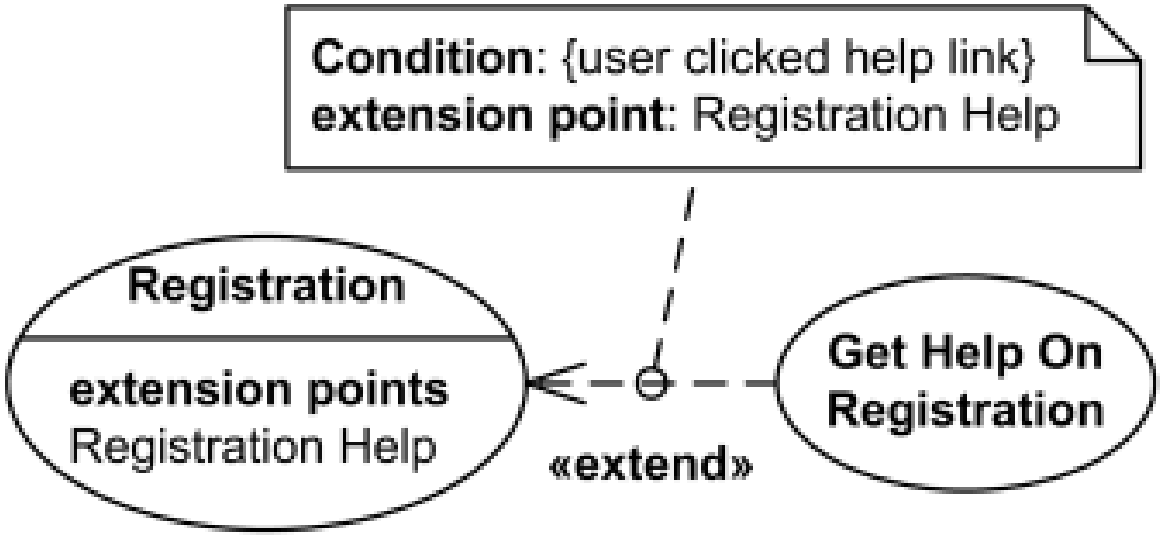
Extend relationship is shown as a dashed line with an open arrowhead directed from the extending use case to the extended (base) use case. The arrow is labeled with the keyword «extend».

Extension Point

An extension point is a feature of a use case which identifies (references) a point in the behavior of the use case where that behavior can be extended by some other (extending) use case, as specified by extend relationship.

Extension points may be shown in a compartment of the use case oval symbol under the heading extension points. Each extension point must have a name, unique within a use case.

Extension example



Include Relationship

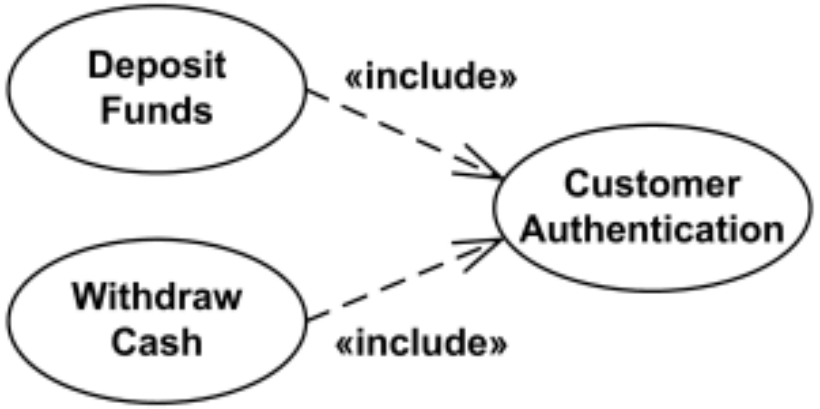
An include relationship is a directed relationship between two use cases when a required, **not optional** behavior of the included use case is inserted into the behavior of the including use case.

The include relationship could be used:

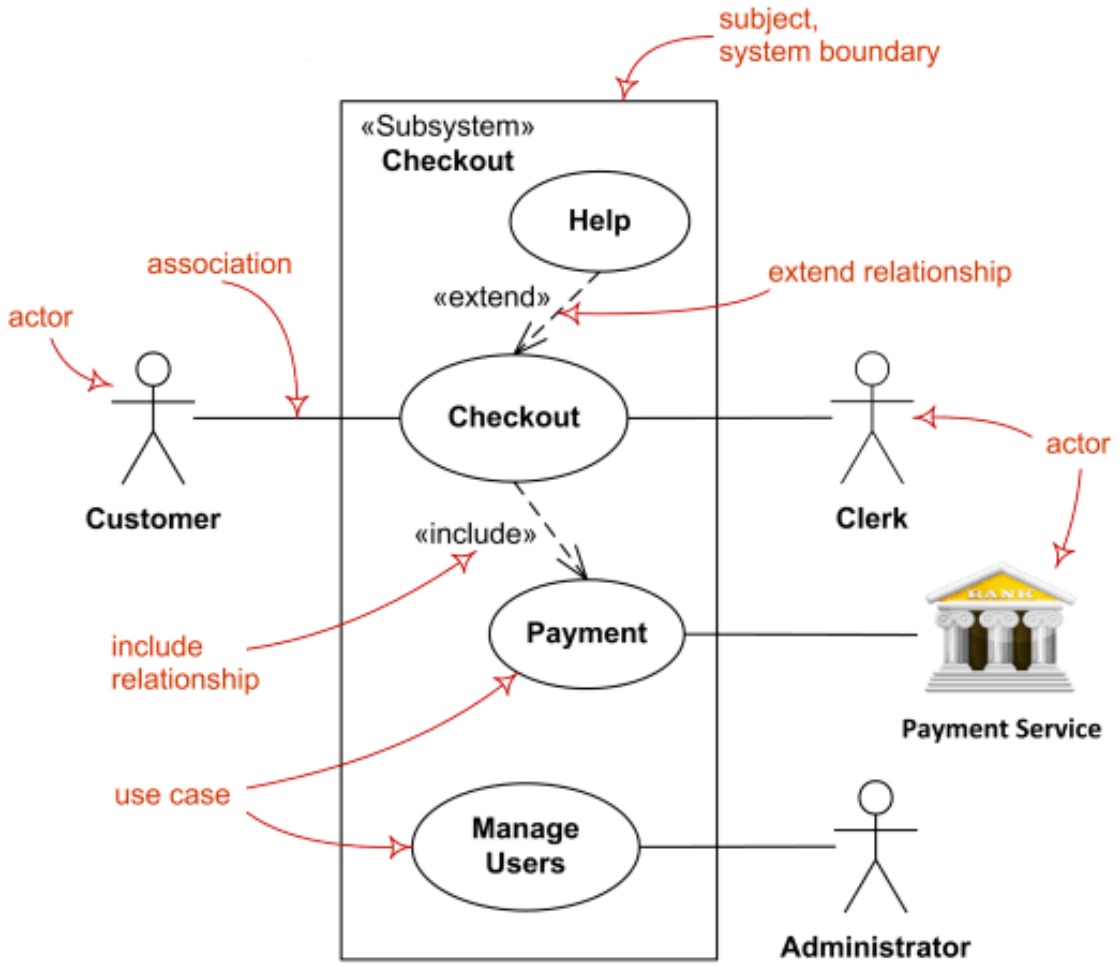
- when there are common parts of the behavior of two or more use cases,
- to simplify large use case by splitting it into several use cases.

Include Relationship

The include relationship between use cases is shown by a dashed arrow with an open arrowhead from the including (base) use case to the included (common part) use case. The arrow is labeled with the keyword «include».



UC diagram example



System use case vs business use case

- In a system use case the subject is a system
- In a business use case the subject is an organization
- Black box vs white box approach
  - It follows that in business use case actors can be internal to the organization

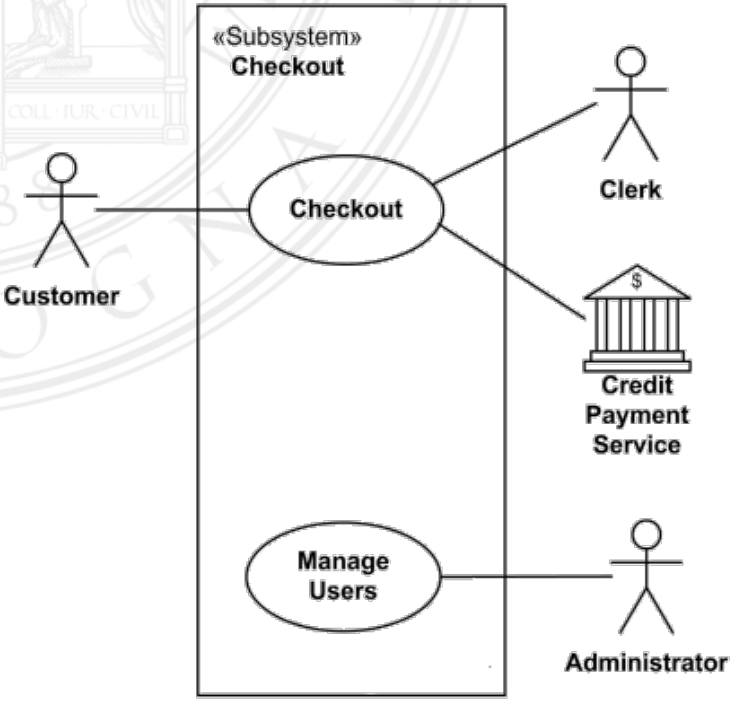
Extent of system use case

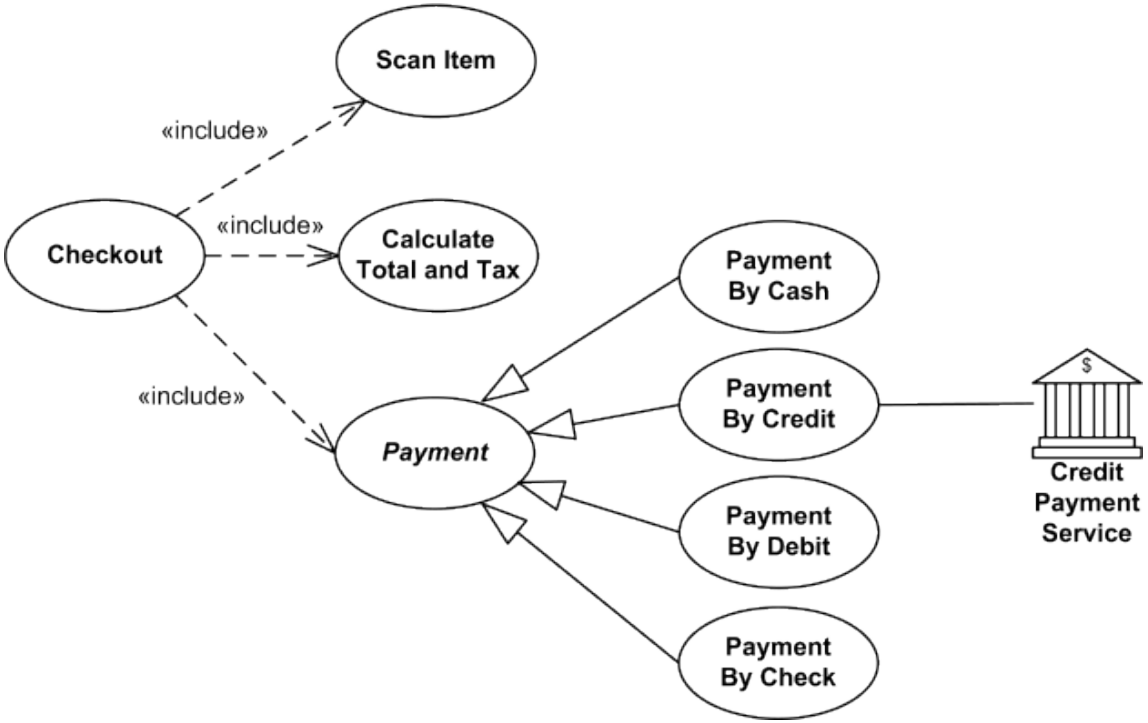
- The **smallest unit of activity** that provides a meaningful result to the user
- Defined by goals that can be accomplished in a ***session***
- A dozen steps at the most
- When a use case involves multiple actors with different goals this should be modeled as multiple use cases

UC do's and don'ts

- Avoid interface creeping (no UI details in UCs)
- Don't decompose for other reasons than reuse
- Master the differences between generalization, include and extend
- Do not create new stereotyped relationships
- Remember: actors are external to the system (for system UCs)
- Time can be an actor

Detail and reuse





UC model and UC diagram

- A UML use case diagram is NOT a use case model
- The diagram can be seen as a *summary*
- Missing aspects;
  - When the UC applies, associated non-functional requirement, pre/post-conditions, …
  - Details about the interaction steps
- There is NO a standard notation to model use case details. Several textual templates have been proposed.

Cockburn's "fully dressed"

- Title: "an active-verb goal phrase that names the goal of the primary actor"
- Primary Actor
- Goal in Context
- Scope
- Level
- Stakeholders and Interests
- Precondition
- Minimal Guarantees
- Success Guarantees
- Trigger
- Main Success Scenario
- Extensions
- Technology & Data Variations List
- Related Information.

Example

ID and Name:	UC-4 Request a Chemical		
Created By:	Lori	Date Created:	8/22/13
Primary Actor:	Requester	Secondary Actors:	Buyer, Chemical Stockroom, Training Database
Description:	The Requester specifies the desired chemical to request by entering its name or chemical ID number or by importing its structure from a chemical drawing tool. The system either offers the Requester a container of the chemical from the chemical stockroom or lets the Requester order one from a vendor.		
Trigger:	Requester indicates that he wants to request a chemical.		
Preconditions:	PRE-1. User's identity has been authenticated. PRE-2. User is authorized to request chemicals. PRE-3. Chemical inventory database is online.		
Postconditions:	POST-1. Request is stored in the CTS. POST-2. Request was sent to the Chemical Stockroom or to a Buyer.		
Normal Flow:	**4.0 Request a Chemical from the Chemical Stockroom** 1. Requester specifies the desired chemical. 2. System lists containers of the desired chemical that are in the chemical stockroom, if any. 3. System gives Requester the option to View Container History for any container. 4. Requester selects a specific container or asks to place a vendor order (see 4.1). 5. Requester enters other information to complete the request. 6. System stores the request and notifies the Chemical Stockroom.		
Alternative Flows:	**4.1 Request a Chemical from a Vendor** 1. Requester searches vendor catalogs for the chemical (see 4.1.E1). 2. System displays a list of vendors for the chemical with available container sizes, grades, and prices. 3. Requester selects a vendor, container size, grade, and number of containers. 4. Requester enters other information to complete the request. 5. System stores the request and notifies the Buyer.		
Exceptions:	**4.1.E1 Chemical Is Not Commercially Available** 1. System displays message: No vendors for that chemical. 2. System asks Requester if he wants to request another chemical (3a) or to exit (4a). 3a. Requester asks to request another chemical. 3b. System starts normal flow over. 4a. Requester asks to exit. 4b. System terminates use case.		
Priority:	High		
Frequency of Use:	Approximately 5 times per week by each chemist, 200 times per week by chemical stockroom staff		
Business Rules:	BR-28, BR-31		
Other Information:	The system must be able to import a chemical structure in the standard encoded form from any of the supported chemical drawing packages.		
Assumptions:	Imported chemical structures are assumed to be valid.		

K. Wiegers, J. Beatty - User requirements – Microsoft Press

A simpler UC specification template

- ID
- Actors
- Pre-conditions
- Main sequence
- Alternative sequences
- Post-conditions

Sequence example

MAIN SUCCESS SEQUENCE (a.k.a. *Happy path*)

1. User selects the option to enter a new transfer order
2. System shows the transfer order form (destination account details, amount, date)
3. User enters data in the transfer form and chooses to execute order
4. System displays order accepted message

EXTENSIONS

3a. Amount higher than availability:

3a1. System shows error message and returns to step 2

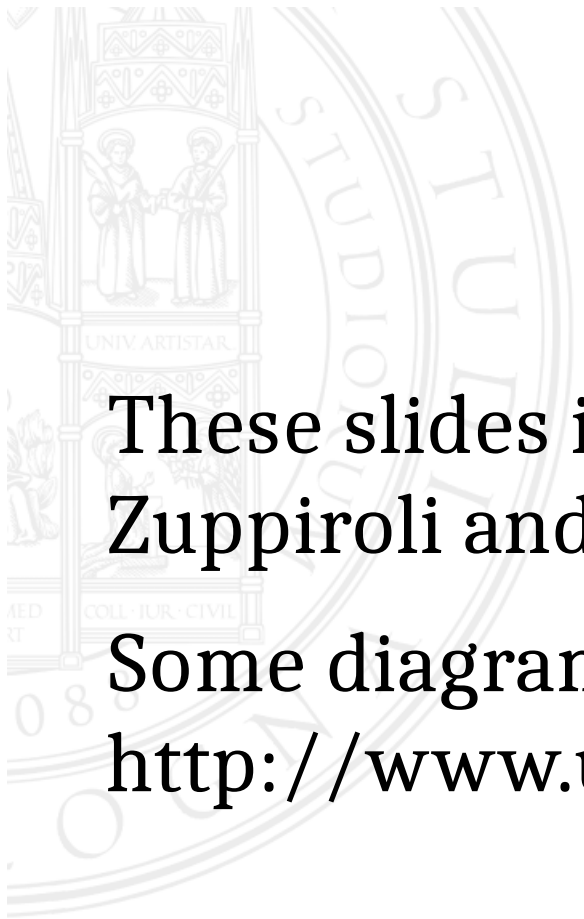
Scenarios

- A scenario can be seen as a UC instance
- It describes a possible interaction between actor(s) and the system
- Scenarios are useful for documentation purposes and to create test cases.

States of a UC model

A UC model is usually subject to iterative refinements:

- Goal Established
- Story Structure Understood
- Simplest Story Fulfilled
- Sufficient Stories Fulfilled
- All Stories Fulfilled



Acknowledgments

These slides include work by Paolo Ciancarini, Sara Zuppiroli and Gian Piero Favini.

Some diagrams and texts are taken from http://www.uml-diagrams.org.

Exercise

A *blog* is a web application presenting a collection of date-tagged messages (*post*s) on miscellaneous topics. Messages are posted by the blog owner who puts them online. The author can associate messages to one or more categories (expressed using keywords).

Blog's visitors can comment messages; the comments, if approved by a moderator (usually the blog's owner), appear in a specific section under the original message.