Modelling System Interactions

Department of Software Engineering, CIT, CTU

Contents

- **Interactions with the new system**
  - How will people interact with the system?
  - When/Why will they interact with the system?
- **Use Cases**
  - Introduction to use cases
  - Identifying actors
  - Identifying cases
  - Advanced features
- **Sequence Diagrams**

Moving towards specification

- What functions will the new system provide?
  - How will people interact with it?
  - Describing functions from a user's perspective
- UML Use Cases
  - Used to show:
    - The functions to be provided by the system which actors will use which functions

Moving towards specification

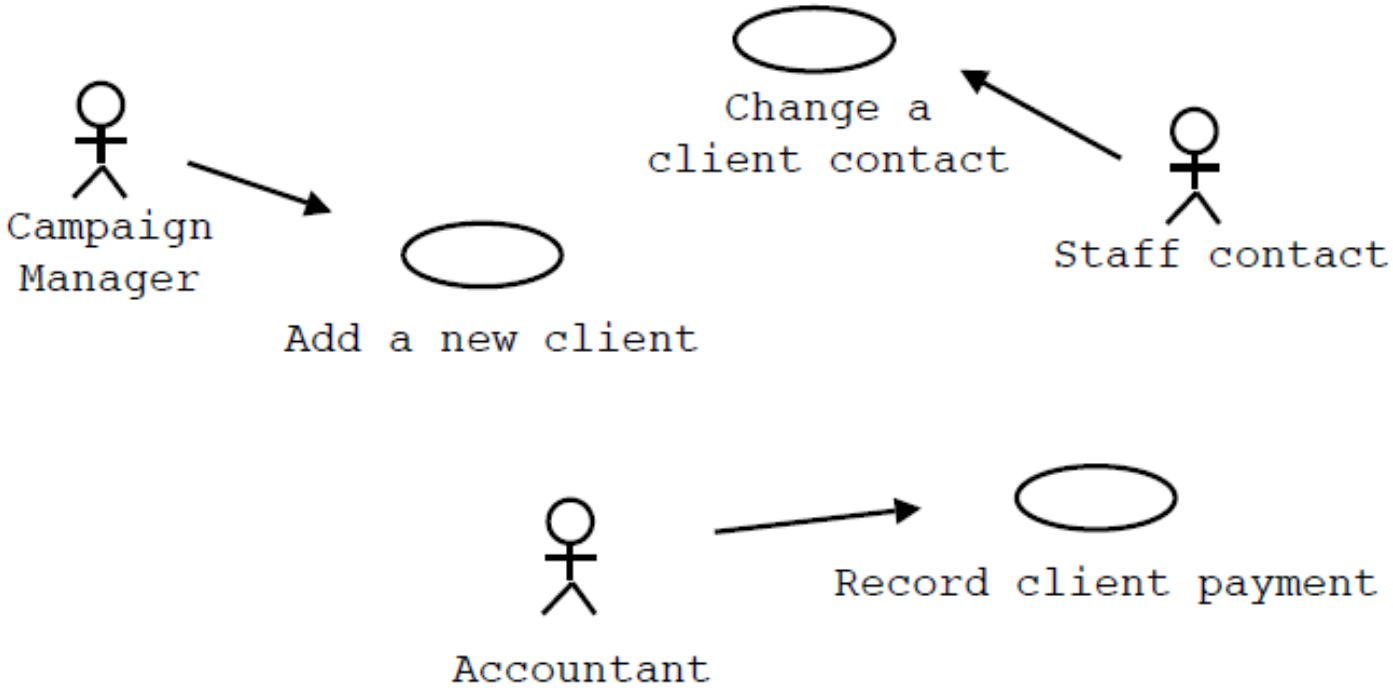
- **Use Case**
  - Requirements are often based on use cases.
  - A use case can be used to describe a systems functional requirements.
  - A use case treats the system which is to be built as a black box. A user (Actor) does something and the system responds.

Moving towards specification

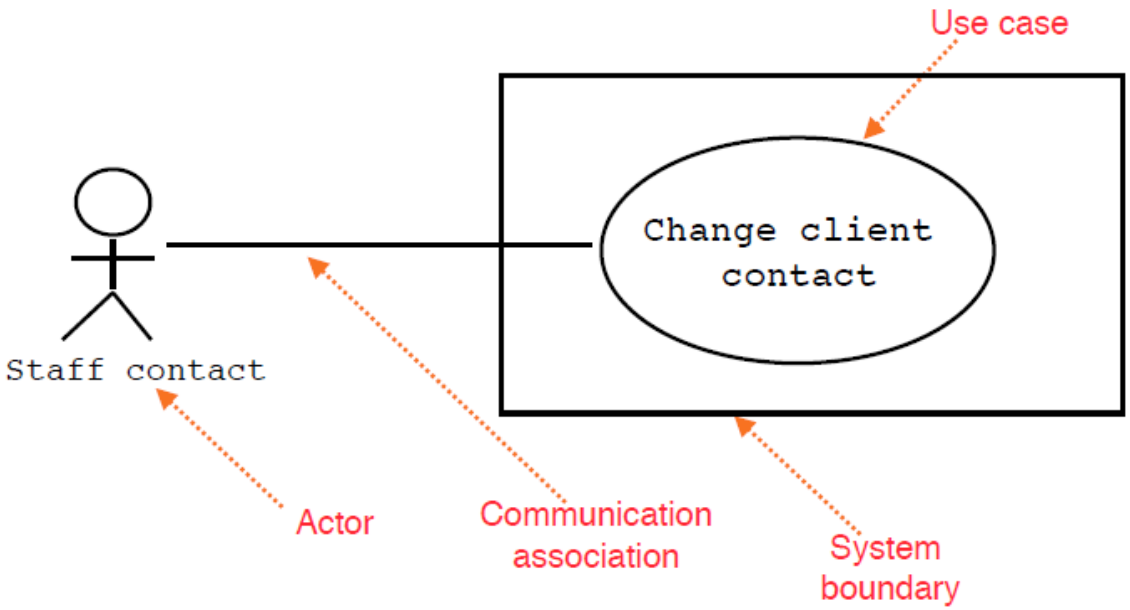
- UML Use Cases
  - Actor
    - Anything that needs to interact with the system:
      - a person.
      - a role that different people may play.
      - another (external) system.
      - All actors must have names according to the assumed role. Examples of actor names (user roles): Customer, Web Client, Student…

Use Case Diagrams

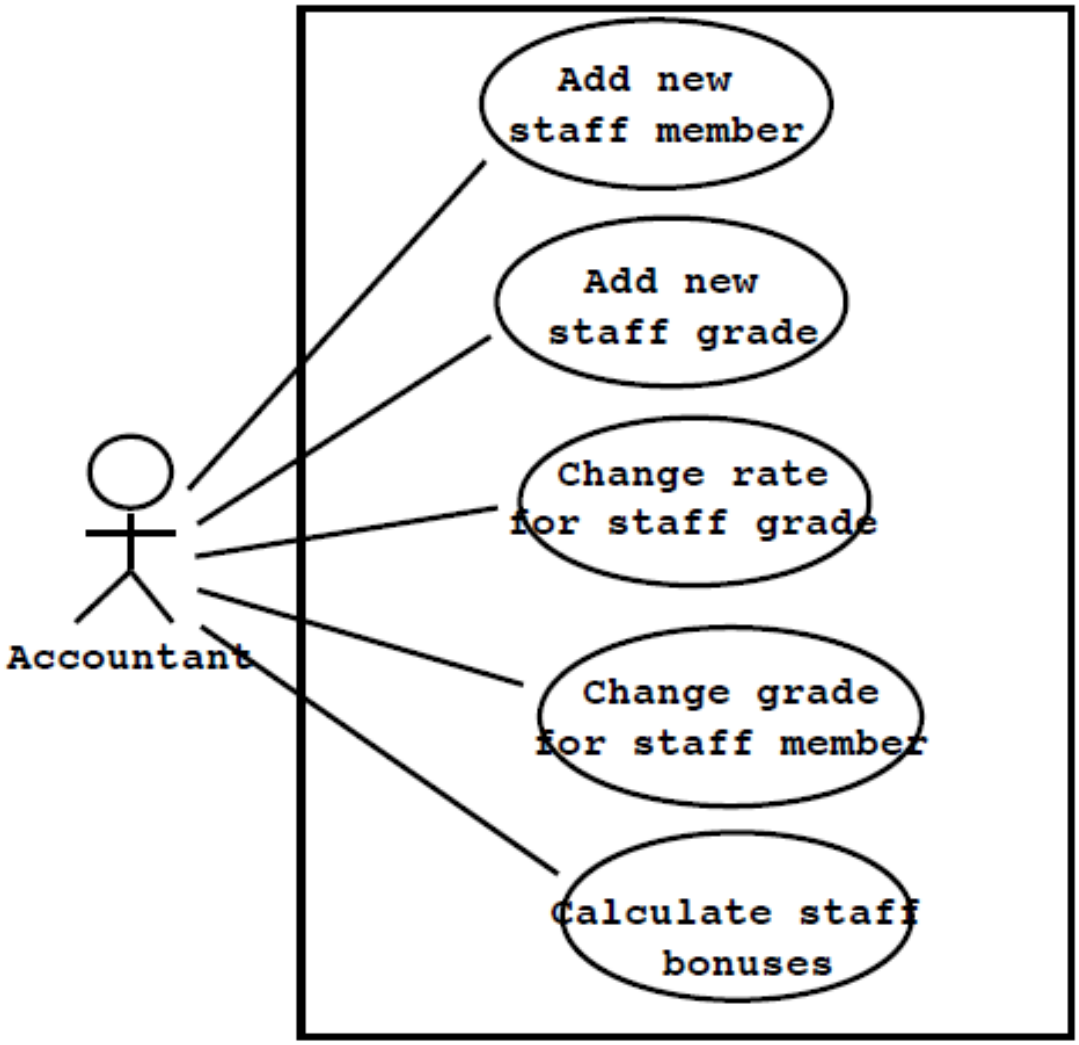
- Capturing the relationships between actors and Use Cases



Notation for Use Case Diagrams



Example

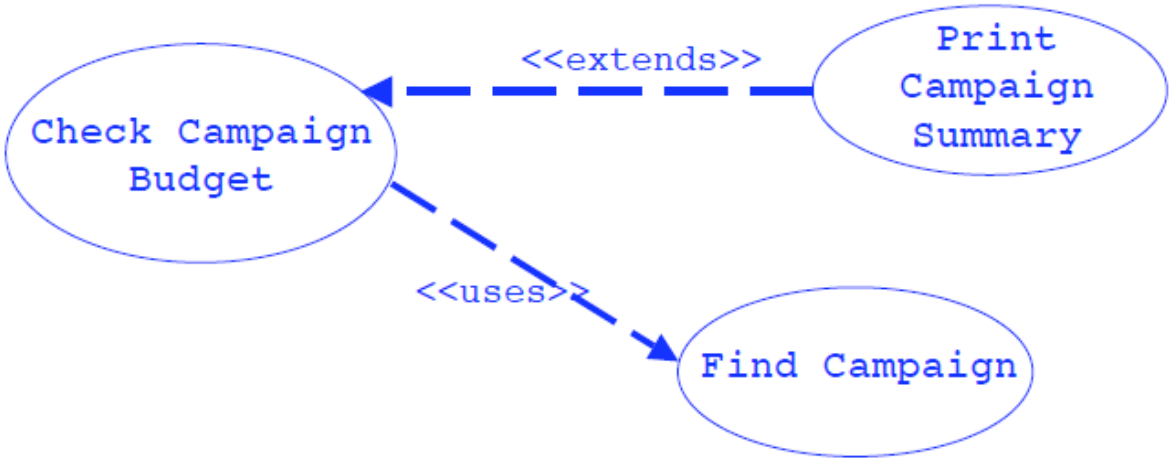

Add new staff member
Add new staff grade
Change rate for staff grade
Change grade for staff member
Calculate staff bonuses
Accountant

<<extends>> and <<uses>>

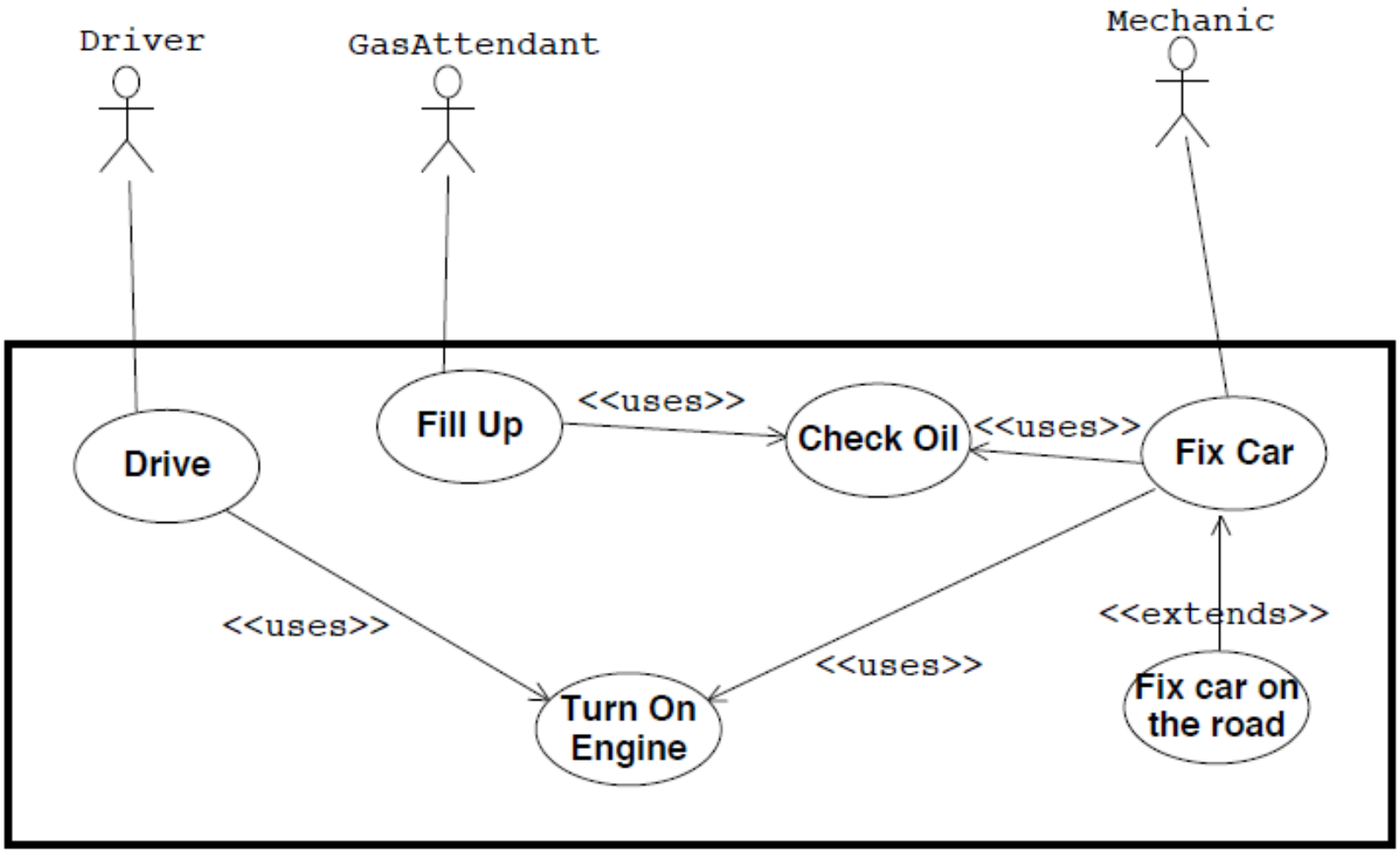
- **<<extends>>:** when one **use case** adds behavior to a **base case.**
  - used to model **a part of a use case** that the user may see as optional system behavior.

- **<<uses>>:** when one **use case** invokes another (like a procedure call).
  - used to avoid describing the same flow of events several times.
  - puts the common behavior in a use case of its own.

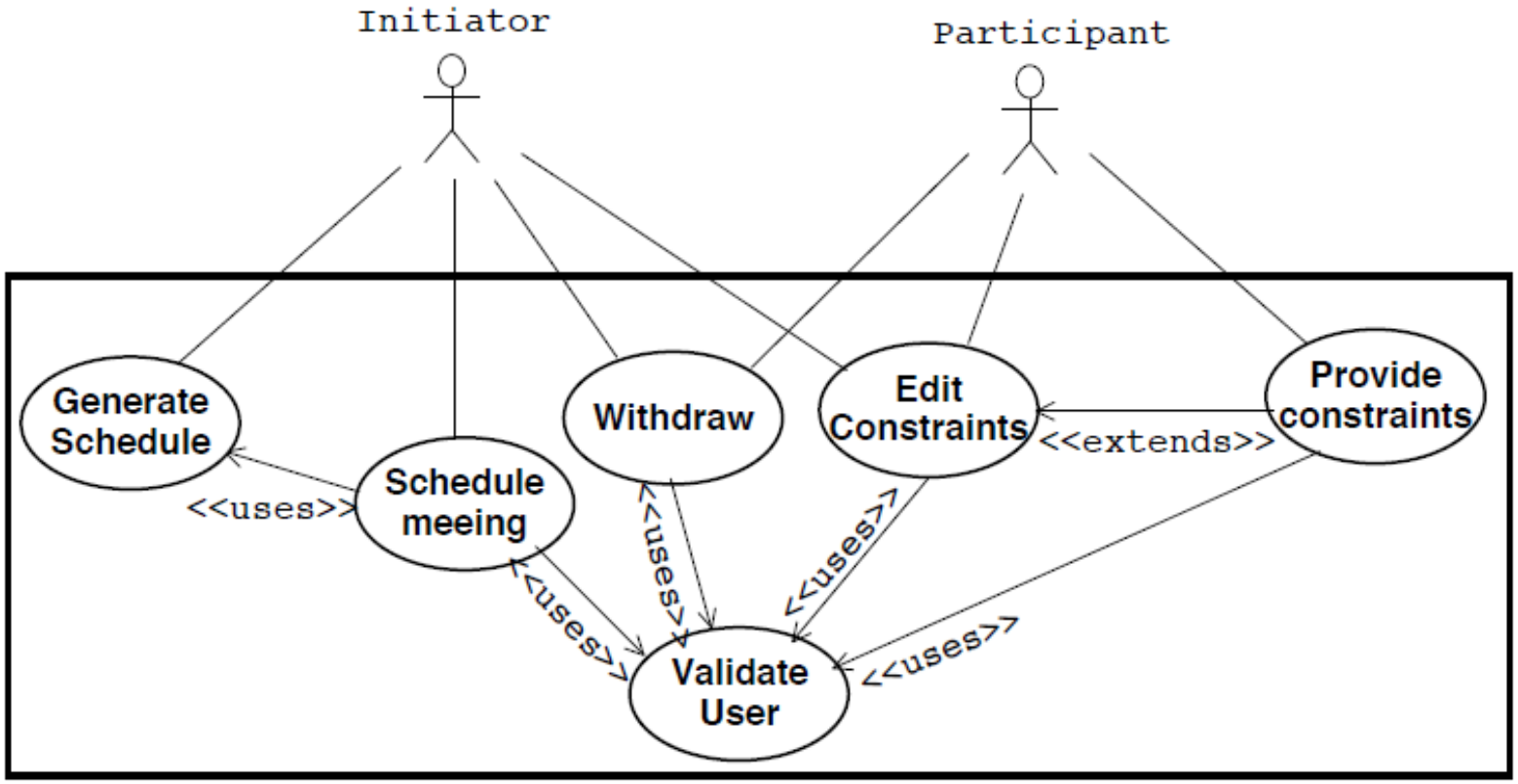
<<extends>> and <<uses>> (cont.)



Sample use cases for a car

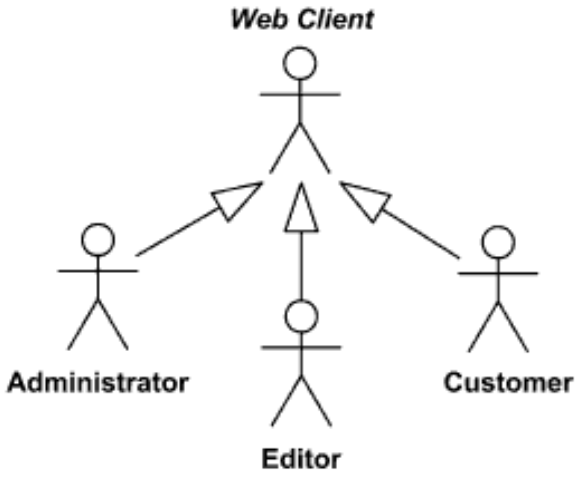


Meeting Scheduler Example



Generalizations

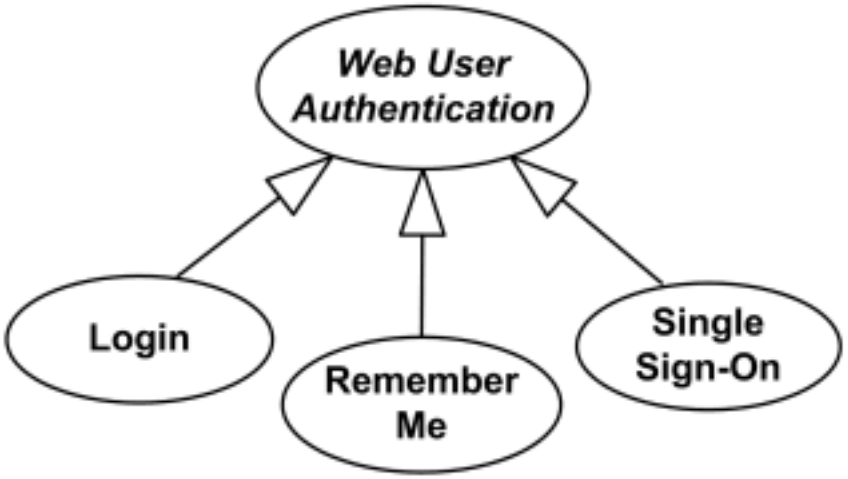
- Actor classes
  - **Generalization** between actors is rendered as a solid directed line with a large arrowhead (same as for generalization between classes).



Web Client actor is abstract superclass for Administrator, Editor and Customer

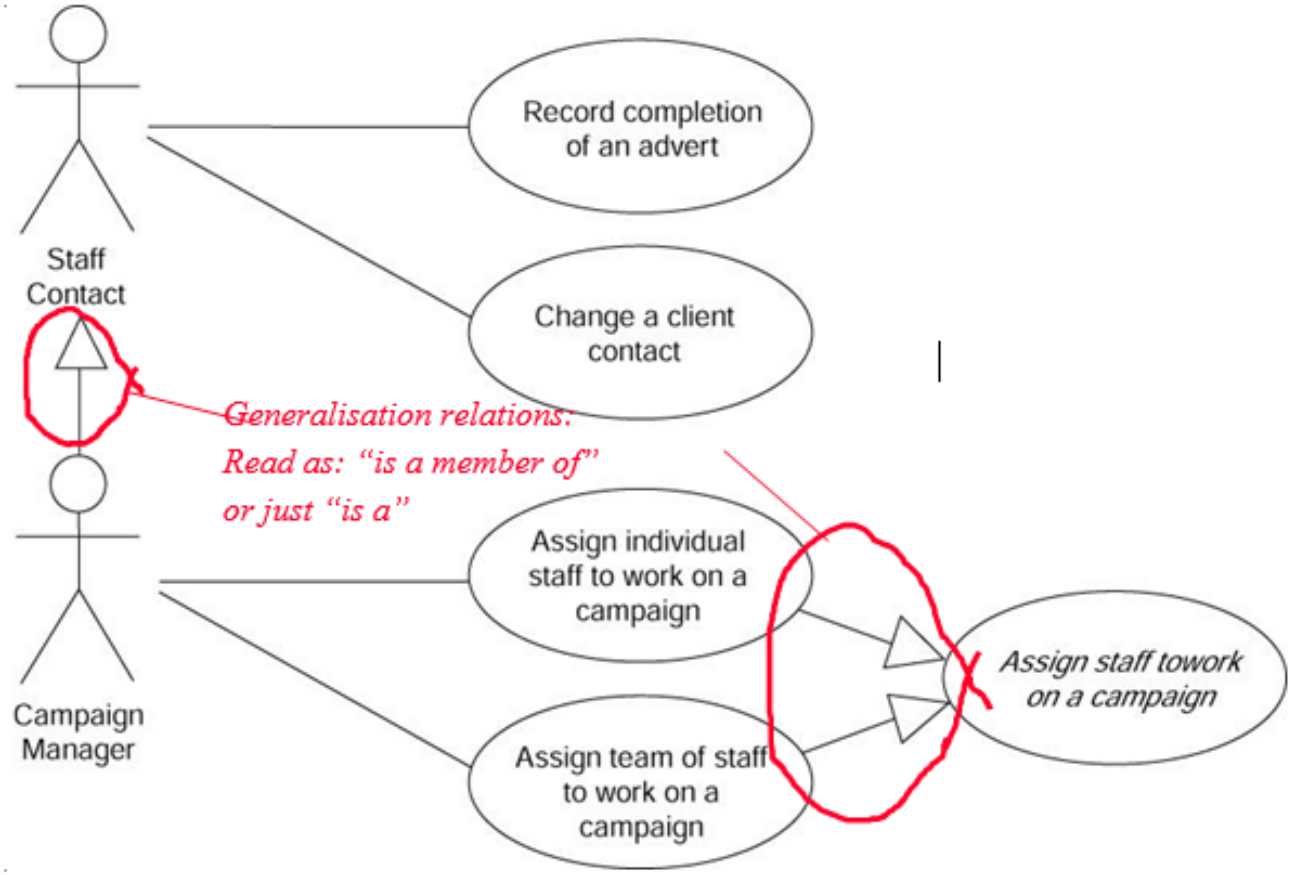
Generalizations

- Use Case classes
  - Sometimes it is useful to identify a generalization of several use cases.



Generalizations (cont.)

- An example



Describing Use Case Behaviors

- Use case **behaviors** may be described in a natural language text (opaque behavior), which is current common practice, or by using UML **behavior diagrams** for specific behaviors such as:
  - activity
  - state machine
  - interaction.

Identifying Actors

- **Ask the following questions**
  - Who will be a primary user of the system? **(primary actor)**
    - Who will need support from the system to do her daily tasks?
    - Who or what has an interest in the results that the system produces?
  - Who will maintain, administrate, keep the system working? **(secondary actor)**
  - Which hardware devices does the system need?
  - With which other systems does the system need to interact with?
- **Look for**
  - The users who **directly** use the system
  - Others who need services from the system

Finding Use Cases

- **For each actor, ask the following questions**
  - Which functions does the actor require from the system?
  - What does the actor need to do?
  - Does the actor need to read, create, destroy, modify, or store some kinds of information in the system?
  - Does the actor have to be notified about events in the system?
  - Does the actor need to notify the system about something?
  - Could the actor's daily work be simplified or made more efficient through new functions provided by the system?

Main references

- Prof Steve Easterbrook, lecture notes, University of Toronto, Canada.
- http://www.uml-diagrams.org

Q&A