Object-oriented Requirements

Software Engineering

2004-2005

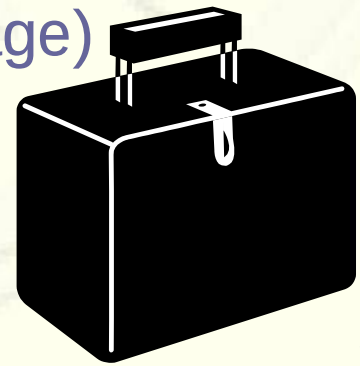
Marco Scotto (Marco.Scotto@unibz.it)

Content

- **Introduction**
- Review of UML in requirement elicitation
- Scenario-based requirements elicitation
- Use case elements, diagram, flow of events
- Use case relationships
- Misuse cases
- Formal SRS vs. use case SRS

Introduction

- Use cases
  - An approach for documenting stakeholder functional requirements
  - Helpful for project development and writing user manual
  - Textual and graphical
  - Use case diagrams
    - Part of UML (Universal Modeling Language)



Content

- *Introduction*
- **Review of UML in requirement elicitation**
- Scenario-based requirements elicitation
- Use case elements, diagram, flow of events
- Use case relationships
- Misuse cases
- Formal SRS vs. use case SRS

Introduction to UML

- Modeling language or diagram notation for OOP
- UML diagrams
  - Use case diagrams
  - State diagrams
  - Class diagrams
  - Understandable to non-technical stakeholders
- Allows more efficient communication among programmers

Content

- *Introduction*
- *Review of UML in requirement elicitation*
- **Scenario-based requirements elicitation**
- Use case elements, diagram, flow of events
- Use case relationships
- Misuse cases
- Formal SRS vs. use case SRS

Scenario-based requirements elicitation (1/3)

- Scenario
  - A subset of use case
  - Sequence of steps or behaviours describing an interaction between user and system
  - Eliciting, validating and documenting requirements
  - A technique of asking questions related to a descriptive story in order to ascertain the design requirements

Scenario-based requirements elicitation (2/3)

- Scenario for subway control system:
  - The train waits at the station until the signal turns green. Once the signal has turned green, the train makes sure no doors are blocked and then closes the door. Once the doors are closed, the train leaves the station at an acceleration rate of 10 km/min². After 6 seconds, the train reaches a cruising speed of 60 km/hr. The train maintains that speed until it reaches the next station. The train stops at the next station.

Scenario-based requirements elicitation (3/3)

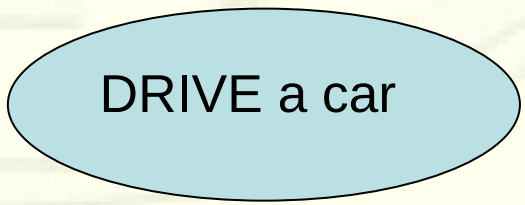
- System problem statements mapped into system specification
- Specification represented as
  - Set of actors
  - Use cases
- Written in natural language
- Scenario-based approach helps to bridge the gap between user/stakeholder view and functional view

Content

- *Introduction*
- *Review of UML in requirement elicitation*
- *Scenario-based requirements elicitation*
- **Use case elements, diagram, flow of events**
- Use case relationships
- Misuse cases
- Formal SRS vs. use case SRS

Elements of a use case (1/2)

- Use case
  - Set of scenarios tied together by a common user goal
  - Major piece of functionality that is complete from beginning to end
  - Sequence of transactions performed by a system
    - Which yields an outwardly visible, measurable result of value for a certain actor
- Notation of a use case
  - Represented as an oval
  - Title/label in a few words, with a present tense verb phrase in active voice
  - Example:



Elements of a use case (2/2)

- Actor
  - Represents whoever or whatever interacts with system
  - Is NOT part of the system
  - Input and/or output information to or from the system
  - Considered as role, not as individual
  - Identifying actors facilitates requirements elicitation
  - Represented by stickman
  - Is always a noun in the scenario

Train administration

1. Identifying the Actors

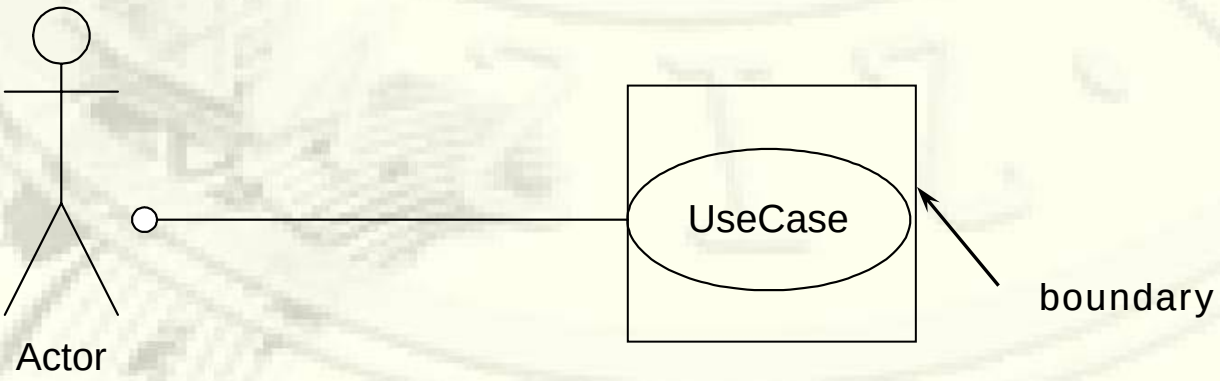
- Who uses the system?
- Who installs the system?
- Who starts up the system?
- Who maintains the system?
- Who shuts down the system?
- What other systems use this system?
- Who gets info from this system?
- Who provides info to this system?
- Does anything happen automatically at present time?

2. Identifying the Use Cases

- What functions will the actor want from the system?
- Does the system store info? What actor will manage it?
- Does the system need to notify an actor about changes in the internal state?
- Are there any external events the system must know about? What actor informs the system of those events?

3. Identifying the system boundary

- System boundary contains things of concerns
  - Clearly defining the boundary is important
  - Things you need to worry about creating
  - In UML use case diagram
    - rectangle



Use case diagram

- Visual representation of relationships between actors and use cases
- Illustrates the system's intended behaviour
- Arrows and lines are used to show relationships between actors and use cases and between use cases
- Default relationship between an actor and a use case is the communication relationship
  - Notation: line with a small circle
- Often developed incrementally

Use case flow-of-events

- Textual description of what really happens
- What to do, not how to do
- Example template:
  - Preconditions
    - What is required before the start of the use case?
  - Main flow
    - Series of declarative steps = sequence of events
    - Notations:
      - [UCx], [Sx]: indicate when and which use case and sub-flow must be run
      - [Ex]: indicates when and which exception may occur
  - Sub-flows
    - List individual sequences of the main flow
  - Alternative flows
    - Define exceptional behaviour and what to do under error conditions

Content

- *Introduction*
- *Review of UML in requirement elicitation*
- *Scenario-based requirements elicitation*
- *Use case elements, diagram, flow of events*
- **Use case relationships**
- Misuse cases
- Formal SRS vs. use case SRS

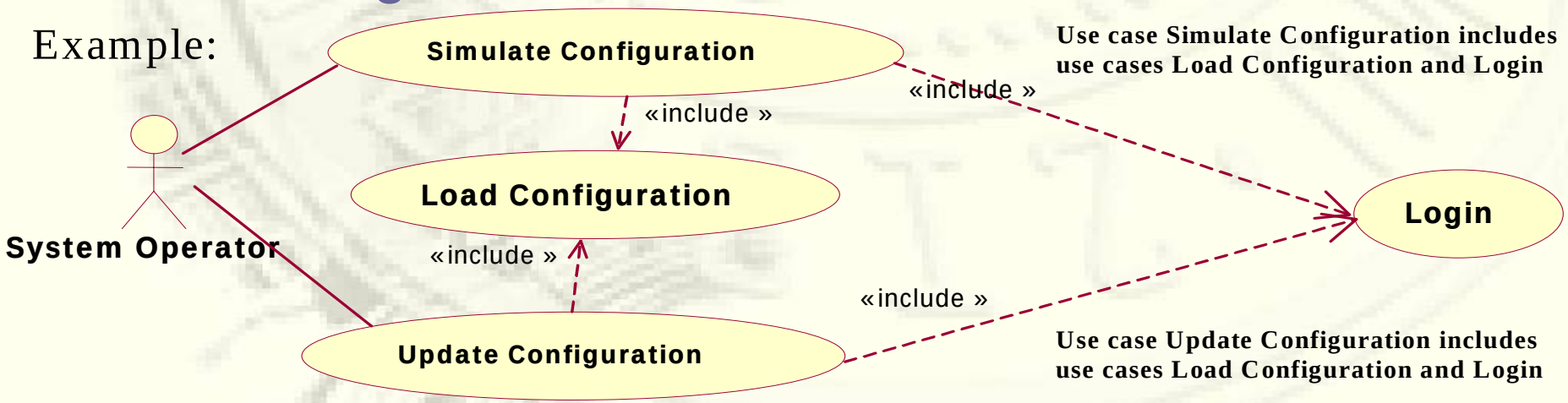
Use case relationships

- Communication relationship
  - Between actor and use case
    - An actor communicates with use cases because it wants measurable results
  - Between use cases
    - A case needs information from or initiates action of another
  - Notation
    - Line or arrow without label
- Include relationship
- Extend relationship

The “include” relationship

- A set of behaviour similar across more than one use case
- Avoiding repetitive copying
- Not default; label <<include>> required
- Included use case is always called by the “including” use case’s flow-of-events

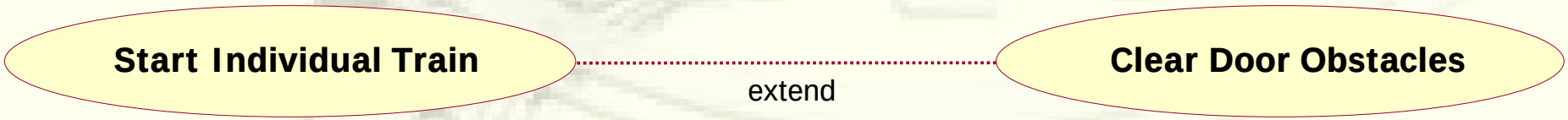
Example:



The “extend” relationship

- Variation on normal behaviour or behaviour executed only under certain stated conditions
- Similar to alternative flows, but used when these are:
  - Fairly complex and/or multi-stepped
  - Possibly with sub-flows and alternative flows
  - Best to segregate functionality into separate extension use case
- Not default, label <<extend>> required
- Use case flow-of-events must call its extending use case(s)

Clear Door Obstacles is a conditional sub-task of Start Individual Train



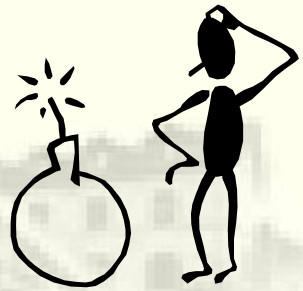
<<include>> vs. <<extend>>

- Consider the following comparison:
  - Use Case X includes Use Case Y
    - Y is a multi-step subtask of X
    - Y will always be completed
  - Use Case X extends Use Case Y
    - X is similar to a sub-task of Y, but more specialized
    - X only happens in an exception situation; Y can complete without X ever happening

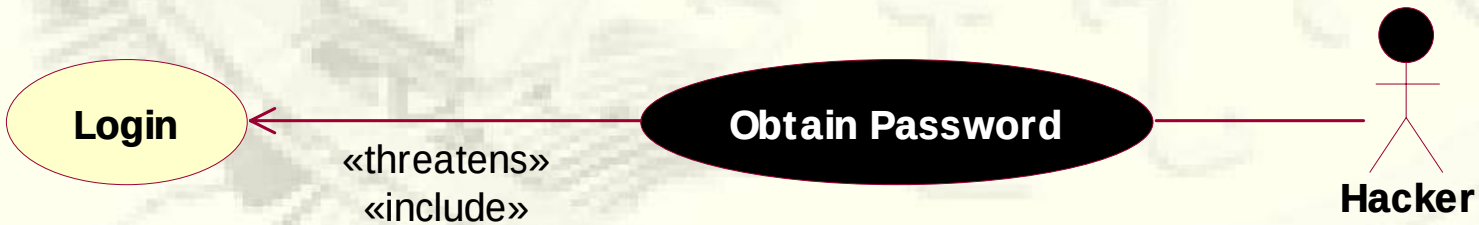
Content

- *Introduction*
- *Review of UML in requirement elicitation*
- *Scenario-based requirements elicitation*
- *Use case elements, diagram, flow of events*
- *Use case relationships*
- **Misuse cases**
- Formal SRS vs. use case SRS

Misuse cases



- A special use case
  - From point of view of an actor hostile to the system
    - E.g. hacker
  - Intentionally threatening the security of system and the privacy of system's users
- Should be considered in requirements stage
- Used to plan for mitigating possible threats



Content

- *Introduction*
- *Review of UML in requirement elicitation*
- *Scenario-based requirements elicitation*
- *Use case elements, diagram, flow of events*
- *Use case relationships*
- *Misuse cases*
- **Formal SRS vs. use case SRS**

Formal SRS vs. Use Case SRS

- SRS – software requirements specification
- A matter of personal preference
- Operational SRS (formal) vs. scenarios (use case)
- Use case diagrams provide overviews of system functionality