Introduction to Use Case Maps

Daniel Amyot, Gunter Mussbacher

damyot@site.uottawa.ca
http://www.UseCaseMaps.org

Table of Contents

- Requirements & Software Engineering Issues
- Introduction to Use Case Maps
- UCM Usage
  - Requirements Capture
  - Architectural Evaluation
  - Transformations to Designs and Tests

Requirements Engineering Issues

- ◆ Early focus on low-level abstractions
- ◆ Requirements and high-level decisions buried in the details
- ◆ Evolution of functionalities difficult to handle (feature interactions, V&V, adaptability to legacy architectures...)
- ◆ Delay introduction of new services

Software Engineering Issues

- Requirements/analysis models need to support new types of dynamic systems
  - Run-time modification of system structure
  - Run-time modification of behaviour
- Need to go from a requirements/analysis model to design models in a seemless way

- We propose **Use Case Maps (UCMs)**!

Table of Contents

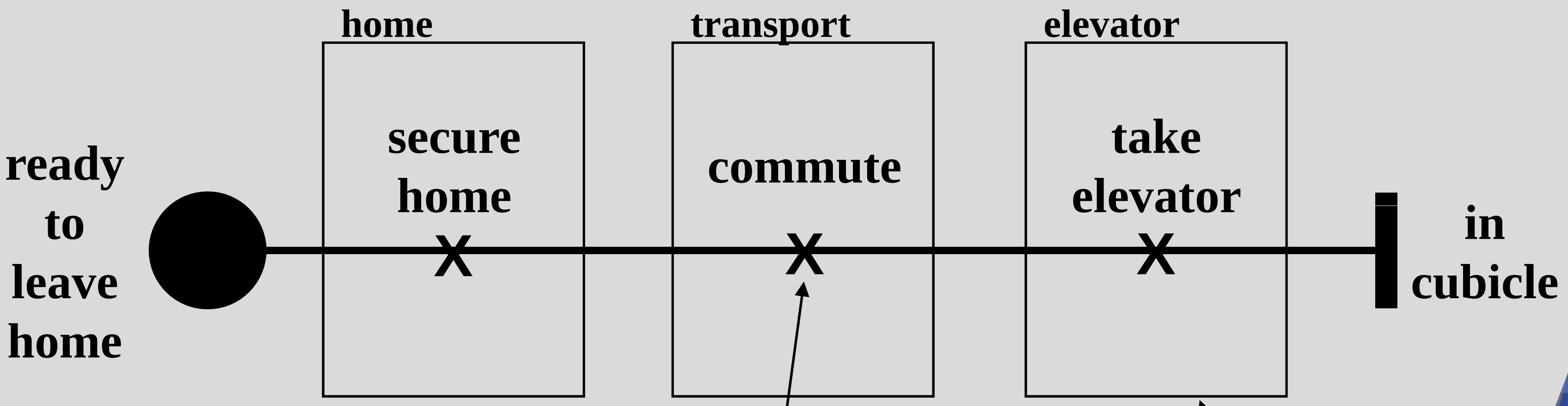
- Requirements & Software Engineering Issues
- Introduction to Use Case Maps
- UCM Usage
  - Requirements Capture
  - Architectural Evaluation
  - Validation and Feature Interaction Detection

Use Case Maps (UCMs)

- The **Use Case Maps** notation allows illustrating a scenario path relative to **optional** components involved in the scenario (gray box view of system)
- UCMs are a scenario-based software engineering technique for describing **causal** relationships between responsibilities of one or more use cases
- UCMs show related use cases in a map-like diagram

UCM Notation - Basic

UCM Example: Commuting



Basic Path
(from circle to bar)

Responsibility Point

Component
(generic)

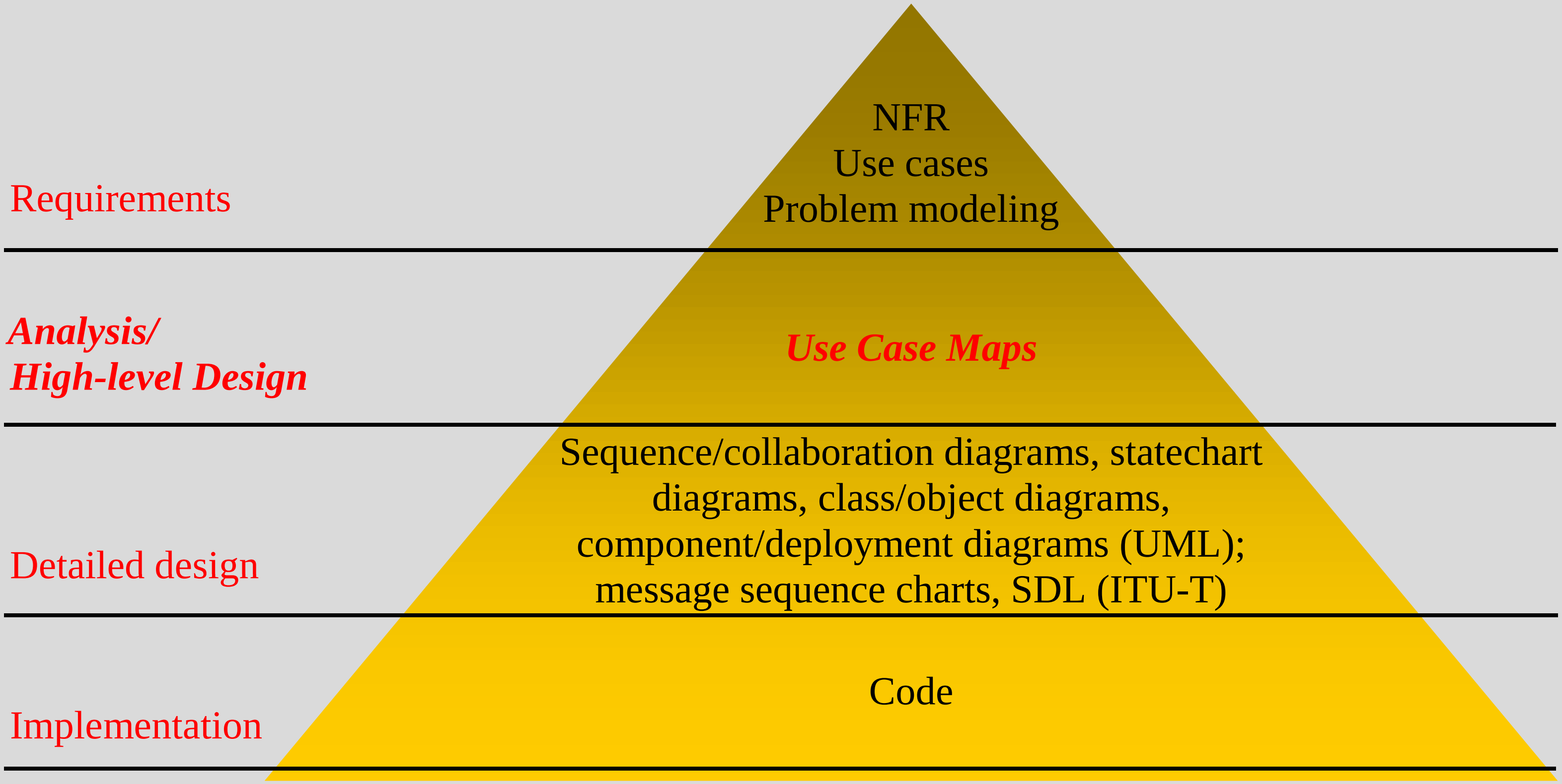
Why Use Case Maps?

- **Bridge** the **modeling gap** between requirements (use cases) and design
  - Link behavior and structure in an explicit and visual way
  - Provide a behavioral framework for making (evaluating) architectural decisions at a high level of design
  - Characterize the behavior at the architecture level once the architecture is decided
- Convey a lot of information in a compact form
- Use case maps **integrate many scenarios** - enables reasoning about potential undesirable interactions of scenarios

Why Use Case Maps?

- Provide ability to **model dynamic systems** where scenarios and structures may change at run-time
  - E-commerce applications
  - Telecommunication systems based on agents
- Simple, intuitive, low learning curve
- Document while you design
- Effective learning tool for people unfamiliar with the domain
- May be transformed (e.g. into MSC/sequence diagrams, performance models, test cases)

The Development Pyramid



Requirements

NFR
Use cases
Problem modeling

Analysis/
High-level Design

Use Case Maps

Detailed design

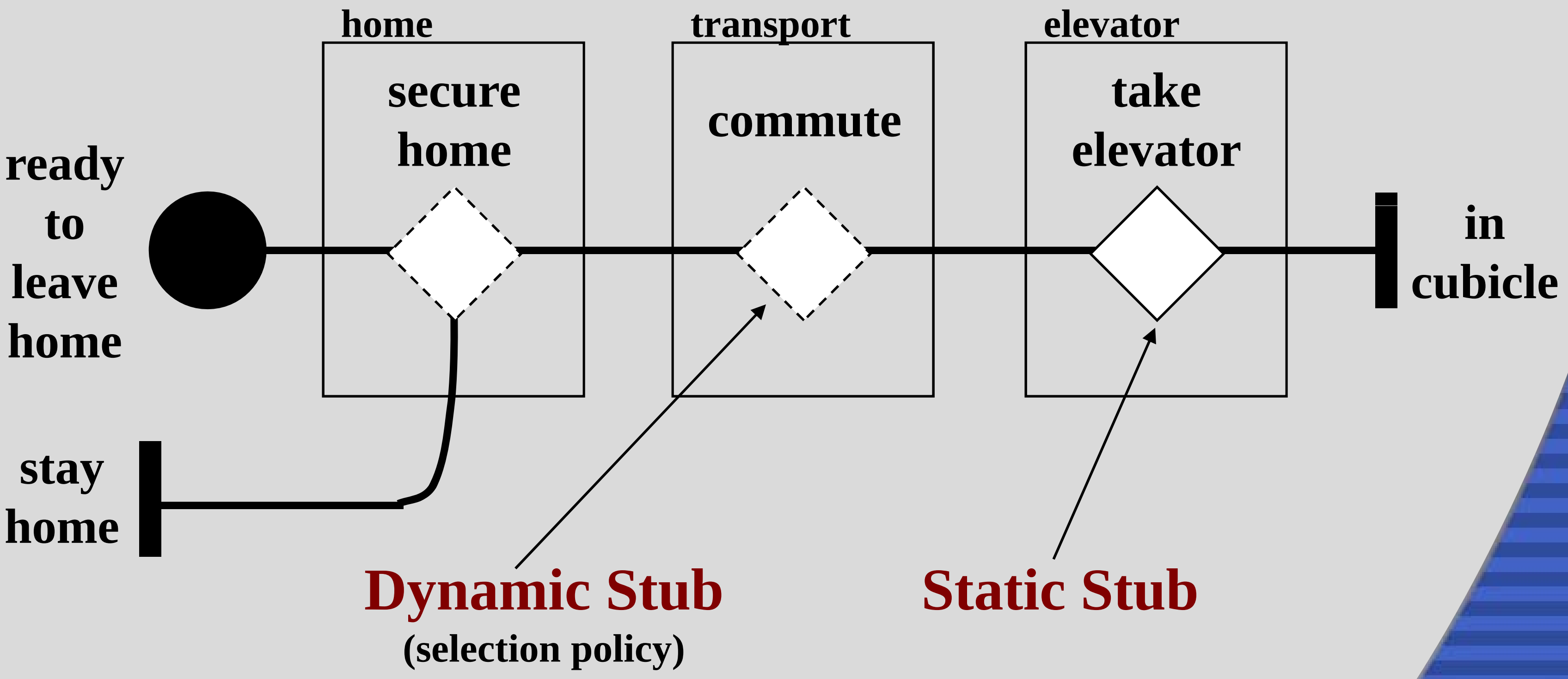
Sequence/collaboration diagrams, statechart diagrams, class/object diagrams, component/deployment diagrams (UML); message sequence charts, SDL (ITU-T)

Implementation

Code

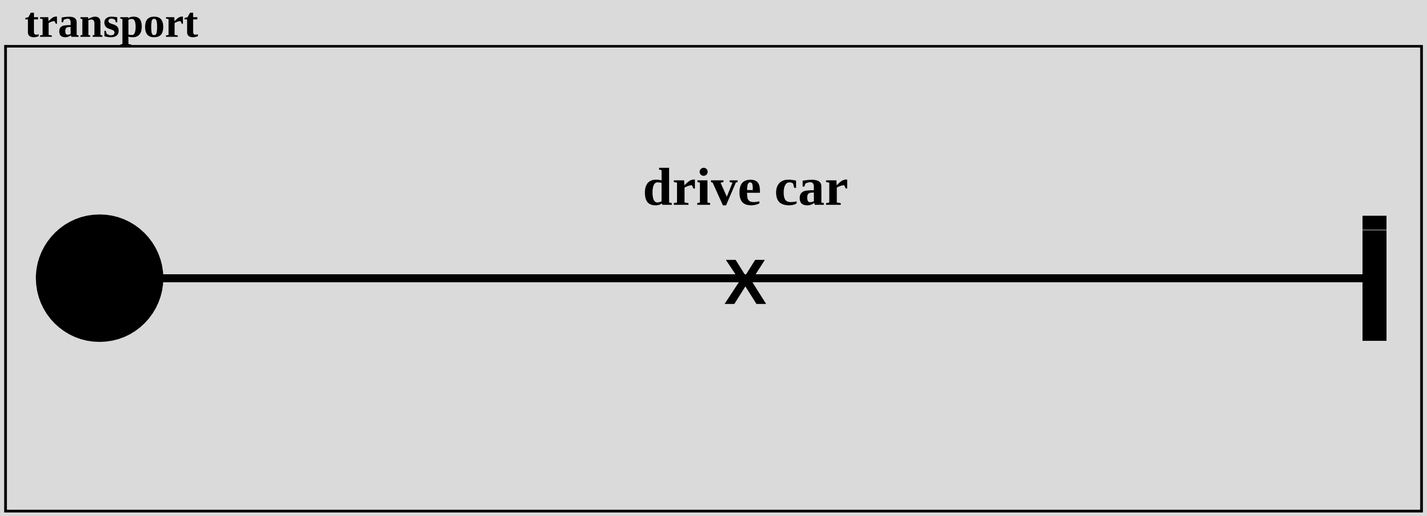
UCM Notation - Hierarchy

UCM Example: Commuting



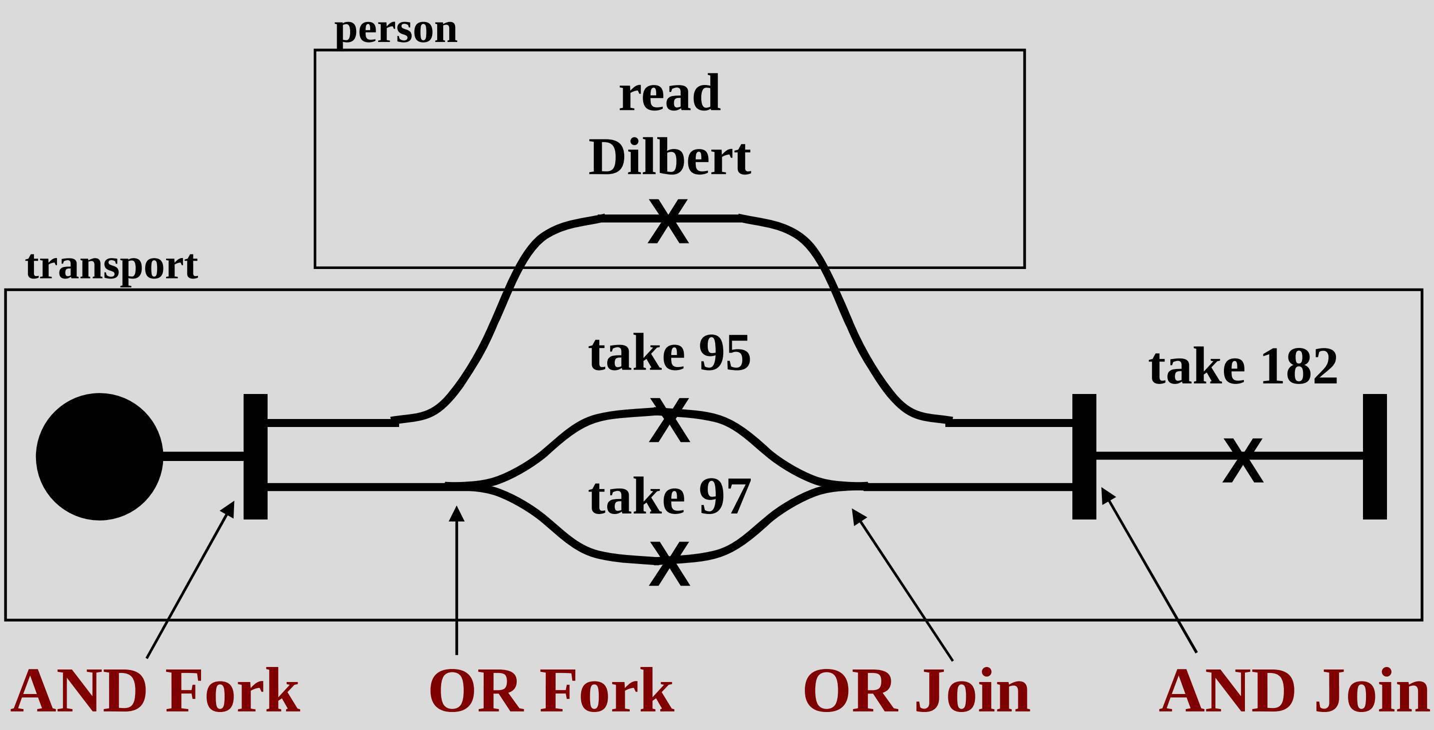
UCM Notation - Simple Plug-in

UCM Example: Commute - Car (Plug-in)



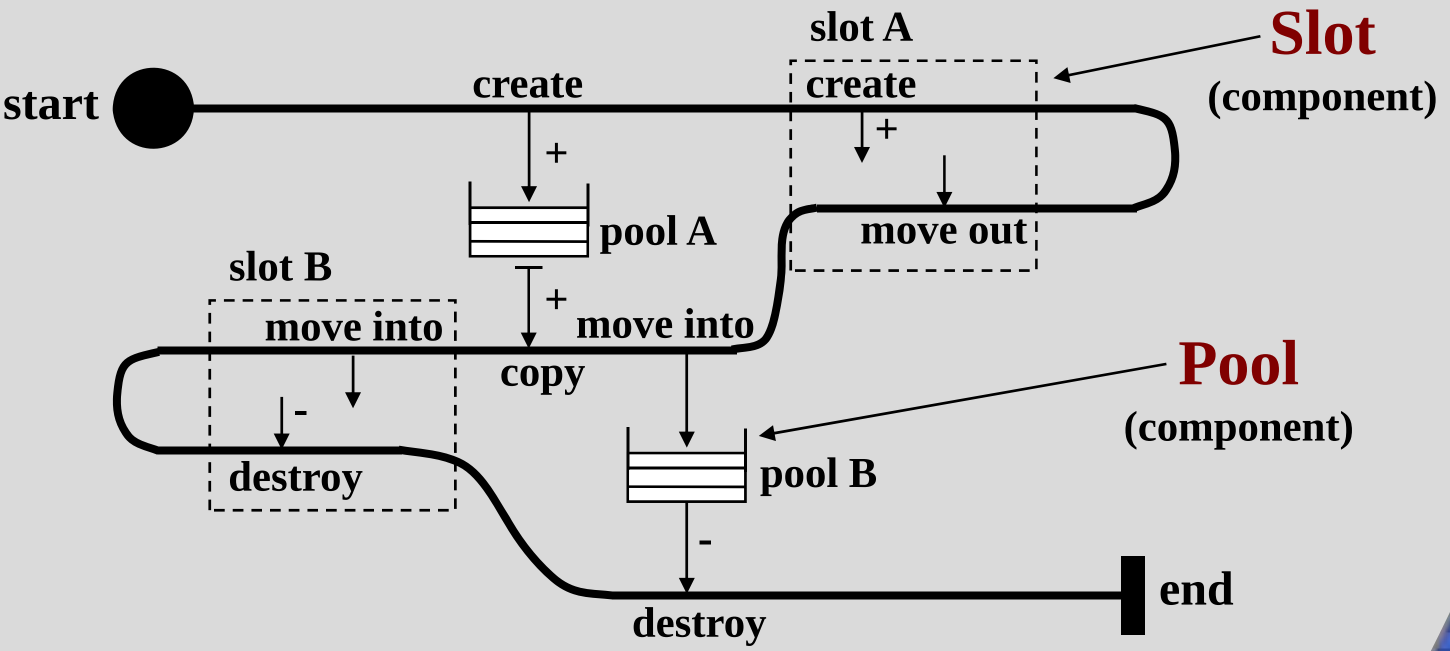
UCM Notation - AND/OR

UCM Example: Commute - Bus (Plug-in)



UCM Notation - Dynamic Structures

Generic UCM Example

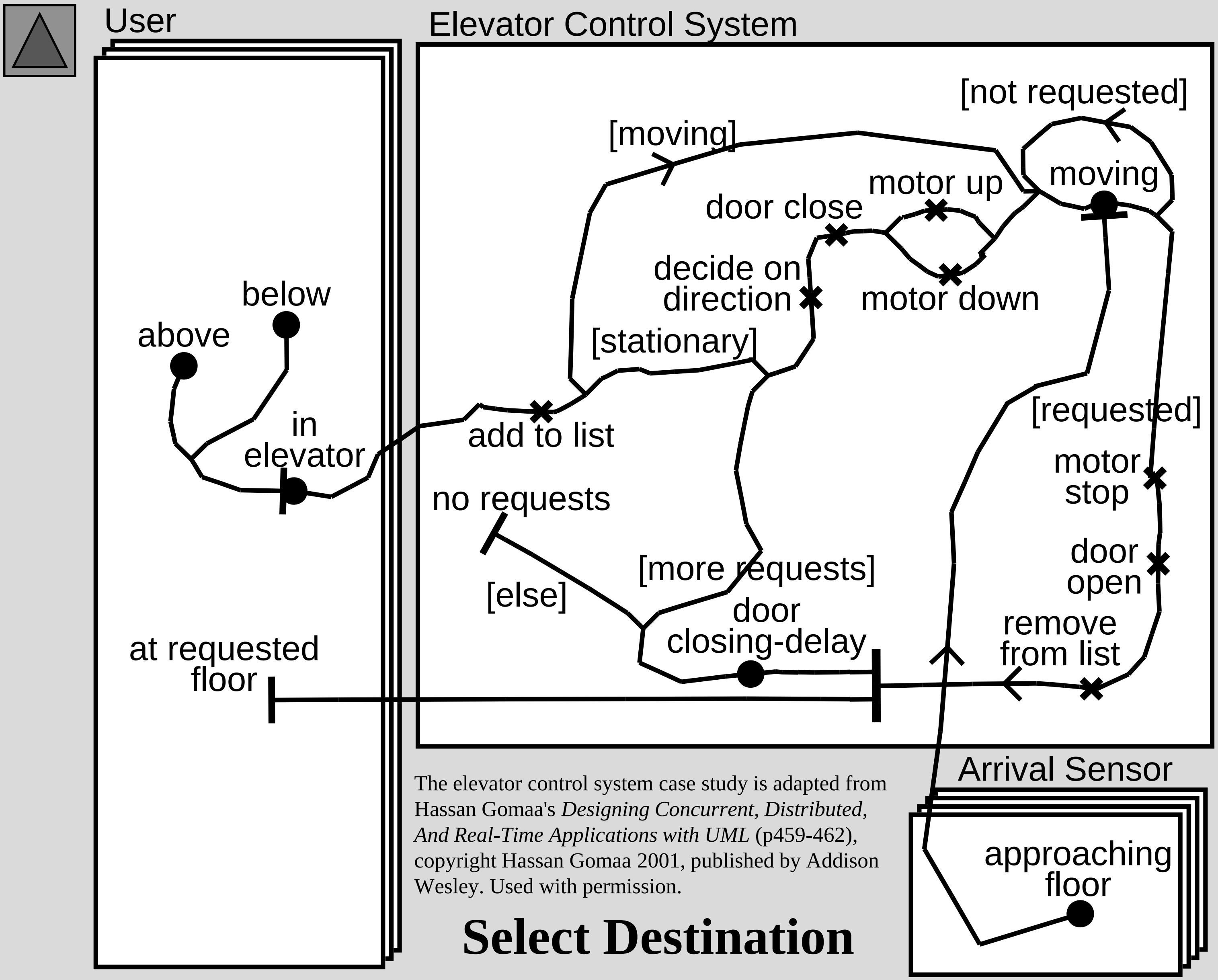


Dynamic Responsibilities and Dynamic Components

Table of Contents

- ◆ Requirements & Software Engineering Issues
- ◆ Introduction to Use Case Maps
- ◆ UCM Usage
  - – Requirements Capture
  - – Architectural Evaluation
  - – Validation and Feature Interaction Detection
  - – Transformations to Designs and Tests
- ◆ Standardization
- ◆ Research Issues

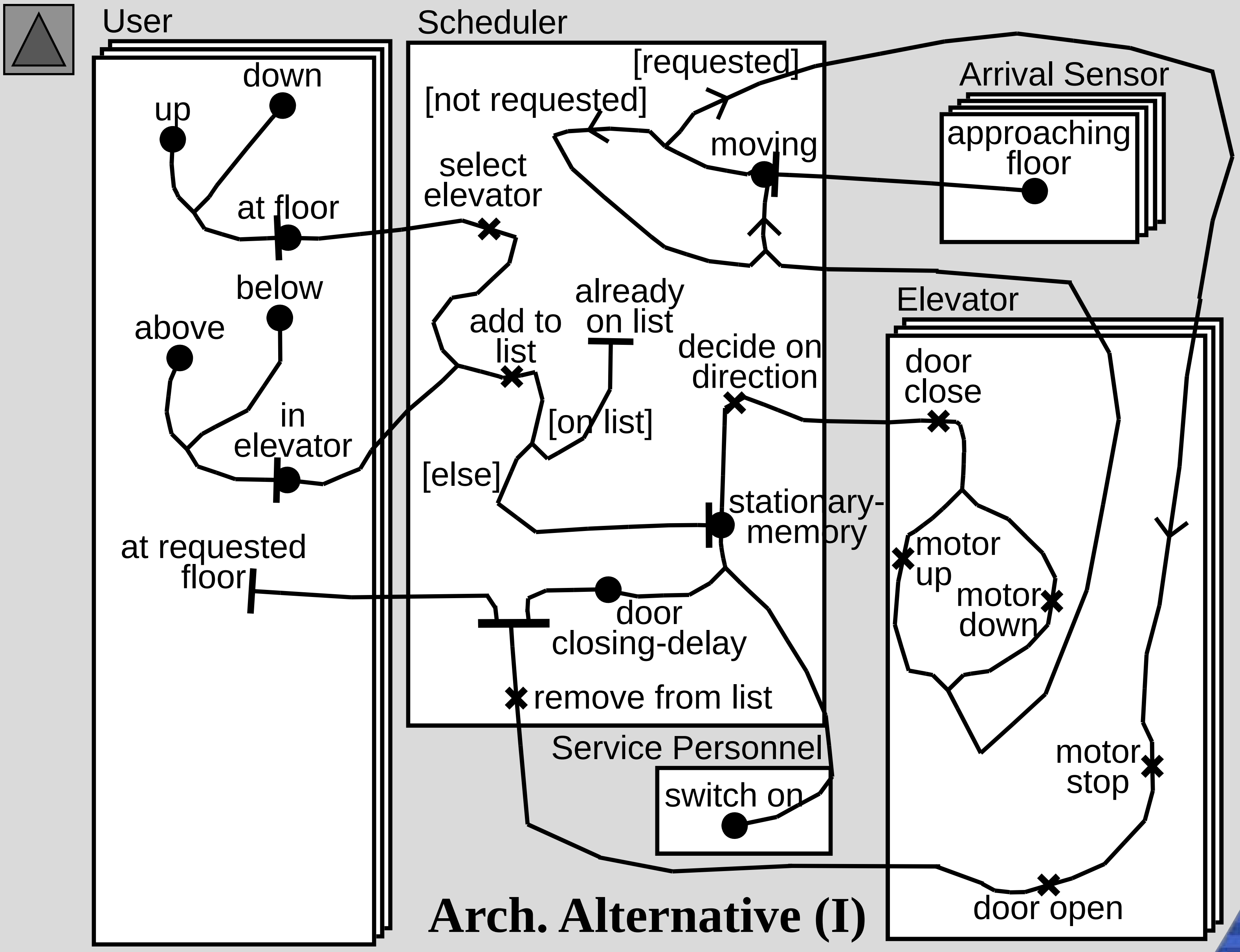
Select Destination



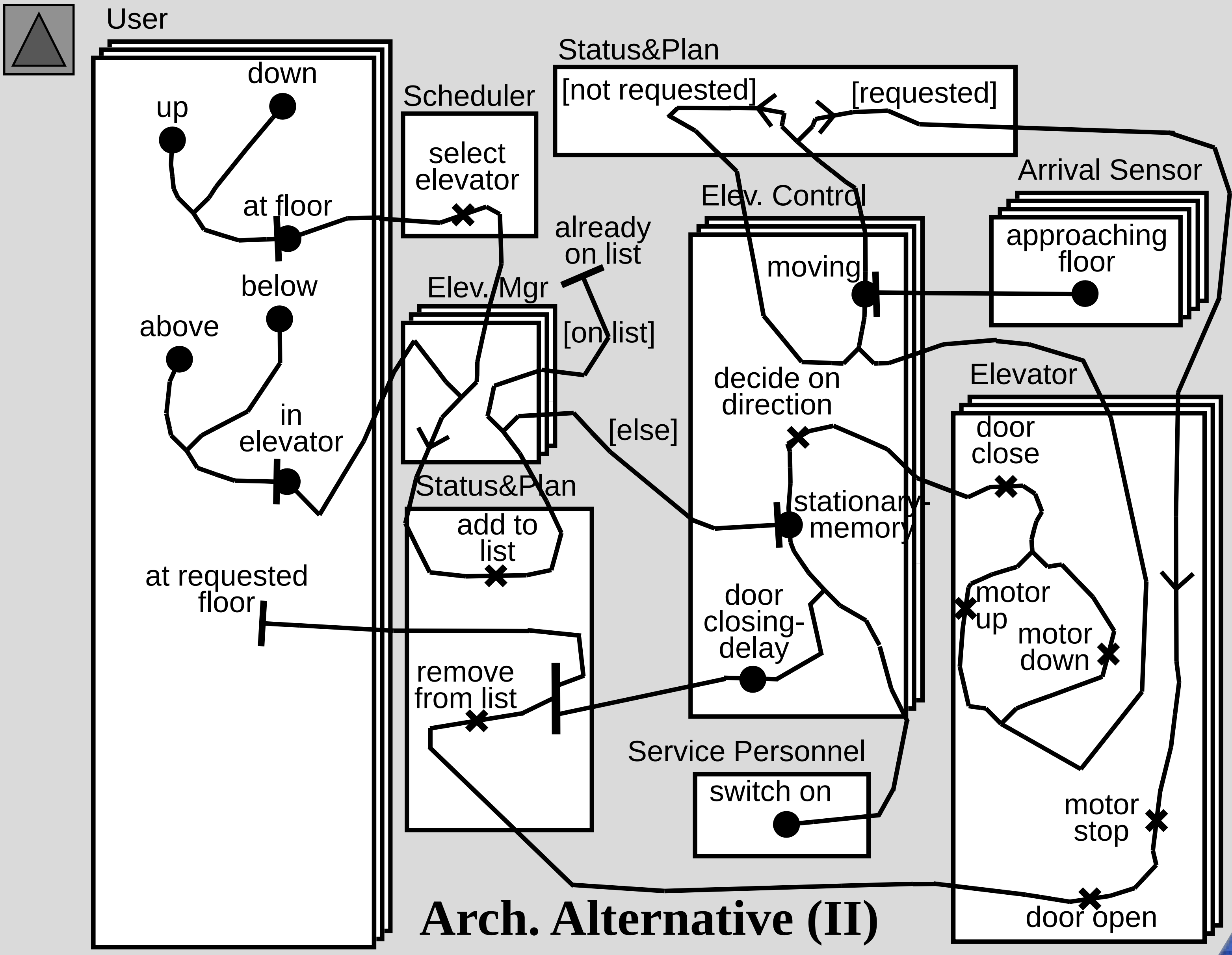
The elevator control system case study is adapted from Hassan Gomaa's *Designing Concurrent, Distributed, And Real-Time Applications with UML* (p459-462), copyright Hassan Gomaa 2001, published by Addison Wesley. Used with permission.

Table of Contents

- ◆ Requirements & Software Engineering Issues
- ◆ Introduction to Use Case Maps
- ◆ UCM Usage
  - – Requirements Capture
  - – Architectural Evaluation
  - – Validation and Feature Interaction Detection
  - – Transformations to Designs and Tests
- ◆ Standardization
- ◆ Research Issues



Arch. Alternative (I)



Arch. Alternative (II)

Table of Contents

- Requirements & Software Engineering Issues
- Introduction to Use Case Maps
- UCM Usage
  - Requirements Capture
  - Architectural Evaluation
  - Validation and Feature Interaction Detection
  - Transformations to Designs and Tests
- Standardization
- Research Issues

Generic Problem with Scenarios

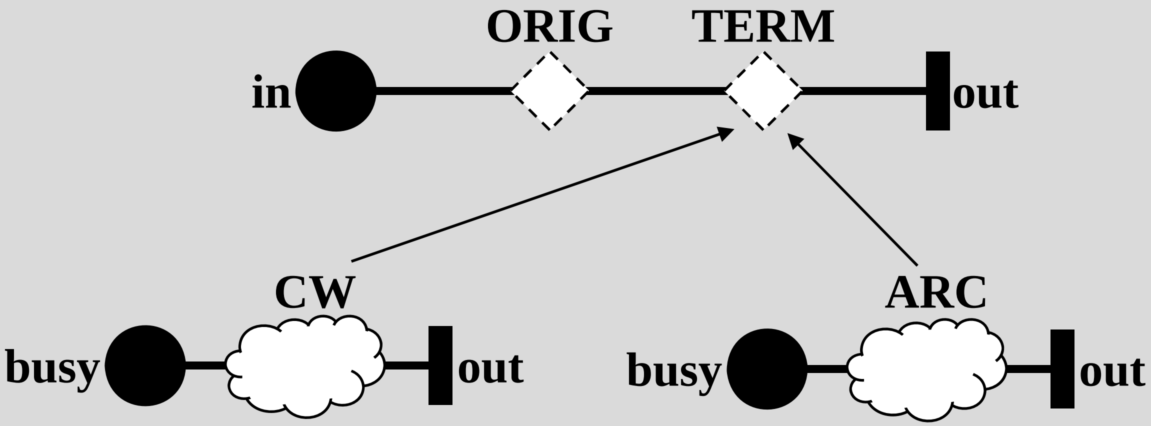
- ◆ Given a set of scenarios capturing informal (functional) requirements
- ◆ Specify (formally) the integration of scenarios
  - – Undesirable emergent behaviour may result…
- ◆ Validate, i.e. look for logical errors and check against informal requirements
- ◆ Numerous tools and techniques can be applied (e.g. functional testing)

UCM Validation by Inspection

- Several problems detectable by inspection
  - Non-determinism in selection policies and OR-forks
  - Erroneous UCMs
  - Ambiguous UCMs, lack of comments
- Many **feature interactions** (FI) solved while integrating feature scenarios together
- Remaining undesirable FI need to be detected!
  - Many are located in stubs and selection policies
  - Need more formal analysis

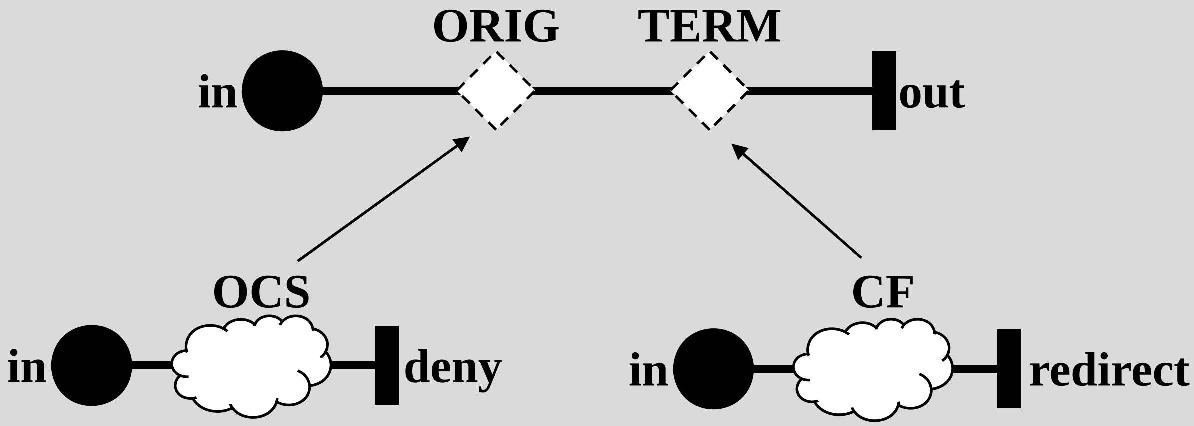
Feature Interaction

- Conflict between candidate plug-ins for the same stub (preconditions of plug-ins are the same)
  - Call waiting (CW) vs. automatic re-call (ARC)



Feature Interaction

- Unexpected behavior among different selected plug-ins for different stubs (postconditions of plug-ins are not the same)
  - Originating call screening (OCS) denies call whereas call forward (CF) redirects call to screened number



Analysis Model Construction

- Source scenario model ⇒ Target analysis model
- Q1. What should the target language be?
  - Use Case Maps Specification ⇒ ?
- Q2. What should the construction strategy be?
  - Analytic approach
    - build-and-test construction
  - Synthetic approach
    - scenarios "compiled" into new target model
    - interactive or automated
- Several approaches studied (UCM to LOTOS, UCM to SDL, …)