

Minitutorial on the Use Case Maps (UCM) Notation

Daniel Amyot, Gunter Mussbacher
 Mitel Networks
 Daniel_Amyot@mitel.com

Objectives

- ◆ Introduce basic Use Case Maps (UCMs) Concepts and the UCM Notation
- ◆ Understand the benefits of Use Case Maps
- ◆ Learn with the help of an exercise
- ◆ Introduce the UCMNAV tool - the Use Case Maps Navigator

Table of Contents

◆ Introduction to Use Case Maps (UCMs) and the UCM Notation

- ◆ Use Case Maps Exercise
- ◆ Enhancing UCMs with Formal Scenario Definitions
- ◆ Tool Demonstration: UCMNAV - The UCM Navigator

Use Case Maps (UCMs)

- ◆ UCMs stand for **Use Case Maps** – a graphical notation that 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 satisfy initial URN requirements
- ◆ UCMs show related use cases in a map-like diagram
- ◆ A map shows the progression of scenarios along use cases

Use Case Maps (UCMs)

- ◆ The intent of UCMs is to facilitate reusability of scenarios across a wide range of architectures and to guide the design of high level architecture
- ◆ UCMs have a history of application to the description of object-oriented systems and reactive systems in various domains
 - Telecommunication, wireless, airline reservation, elevators, railway, agents, network management applications, web applications, graphical user interfaces, drawing packages, multimedia applications, banking applications, object-oriented frameworks, "work patterns" of software engineers, etc.

History of Use Case Maps

- ◆ End of 1980's at Carleton University, Canada
 - Buhr (design) and Woodside (performance)
- ◆ Slices: Vigder and Buhr, 1992
- ◆ Timethreads: Buhr and Casselman, 1993
- ◆ Use Case Maps: Buhr and Casselman, 1995
- ◆ Still evolving...
 - Scenario definition
 - Model generation (e.g. UCM to MSC)
 - Standardization
 - UCM patterns and styles
 - ...

Use Case Maps Web Page

- ◆ <http://www.UseCaseMaps.org/>
- ◆ Prime source of information about UCMs
- ◆ Supports the UCM User Group
 - Nearly 200 members. Mailing list and newsletter.
- ◆ UCM Virtual Library
 - Over 50 publications and 25 presentations
- ◆ Tools (**UCM Navigator** and others)
- ◆ XML Document Type Definition (DTD)
- ◆ UCMs and UML

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

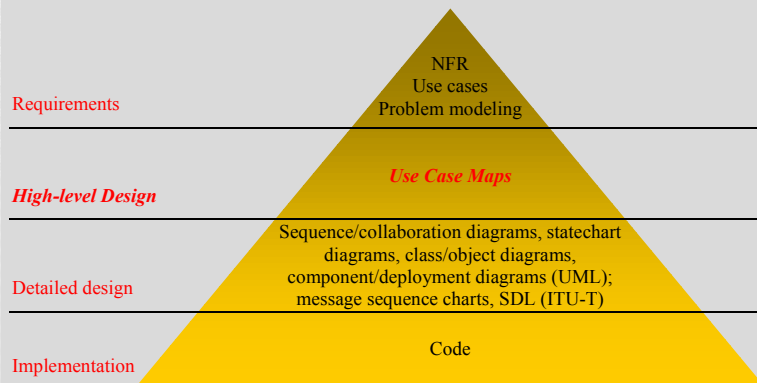
Why Use Case Maps?

- ◆ 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, very low learning curve
- ◆ Excellent documentation tool – document while you design
- ◆ Effective learning tool for people unfamiliar with domain
- ◆ May be transformed (e.g. into MSC/sequence diagrams, performance models, test cases)

Information Needed to Construct Use Case Maps

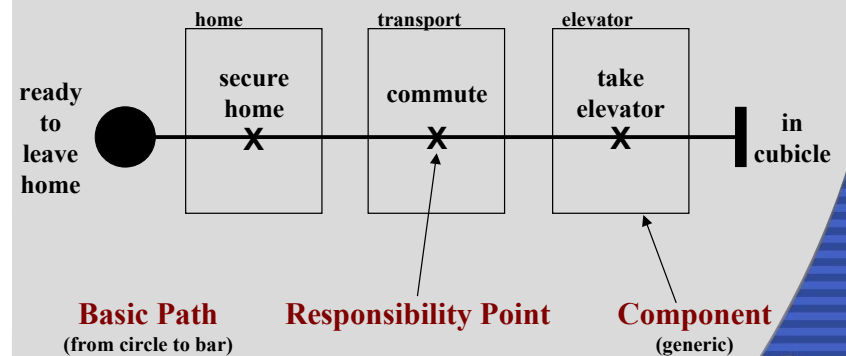
- ◆ Informal requirements or use cases or extensive domain knowledge
- ◆ Responsibilities either stated or inferred from requirements/use cases/domain knowledge
- ◆ Architectural components (optional)
 - High level description of these components (their nature, the relationships between them)
 - Vision of architectural structure
- ◆ Clearly defined interface between the environment and the system not required initially but eventually
 - Alternatives may be considered

The Design Pyramid



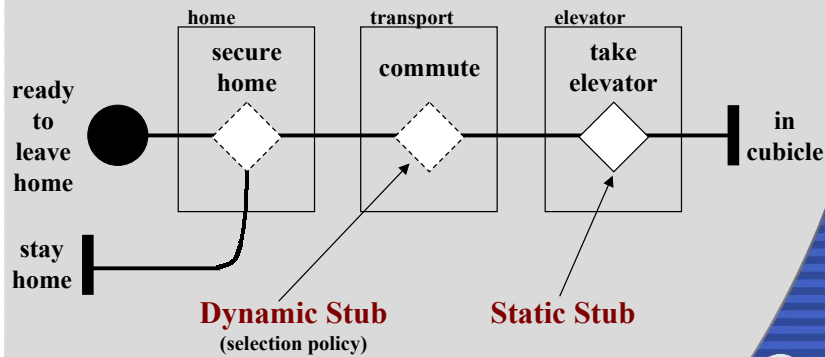
UCM Notation - Basic

UCM Example: Commuting



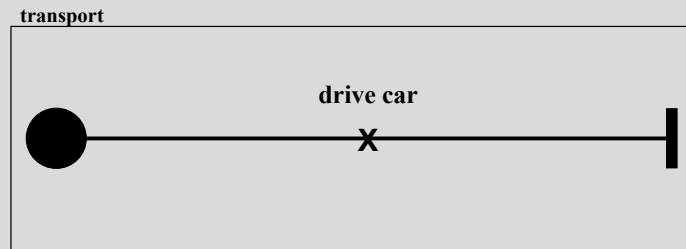
UCM Notation - Hierarchy

UCM Example: Commuting



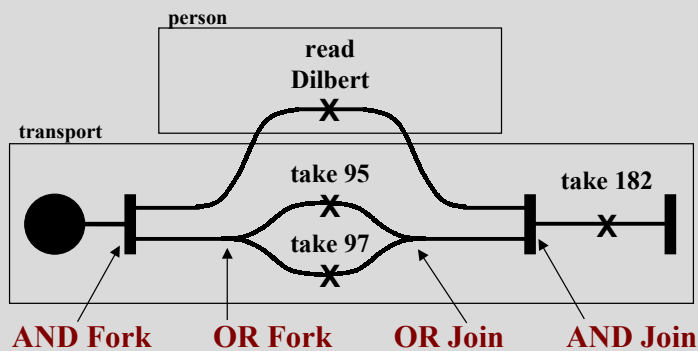
UCM Notation - Simple Plug-in

UCM Example: Commute - Car (Plug-in)



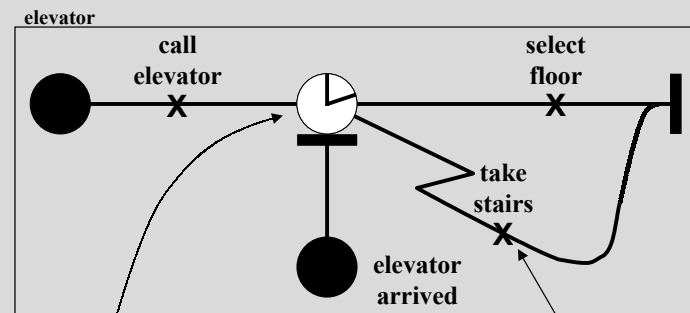
UCM Notation - AND/OR

UCM Example: Commute - Bus (Plug-in)



UCM Notation - Waiting Place / Timer

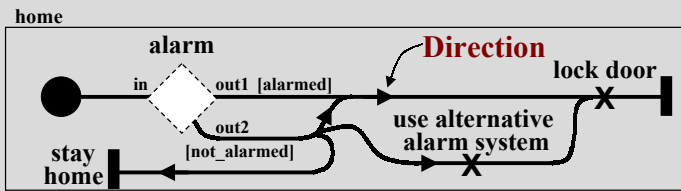
UCM Example: Take Elevator - Default (Plug-in)



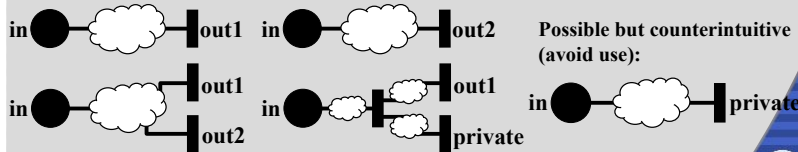
Timer (special waiting place) **Timeout Path**
 Note: Waiting places may be regular, memory, signal, or delay.

UCM Notation - Simple Plug-in with Stub

UCM Example: Secure Home - Default (Plug-in)

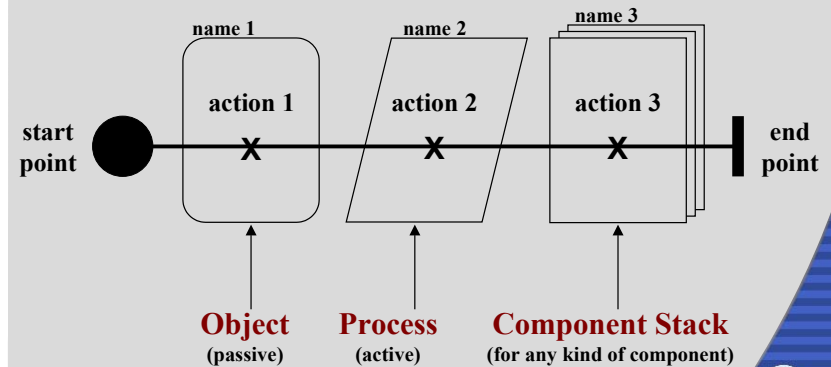


Possible plug-ins for a stub with one in path and two out paths:



UCM Notation - Component - Object, Process & Stack

Generic UCM Example



Key Points - UCM Introduction & Notation

- ◆ Modeling behavioral aspects of a system
 - UCM path including all path elements
 - More abstract than message exchanges (causal)
- ◆ Modeling structural aspects of a system
 - UCM components
 - ◆ Quite similar to the concept of roles in UML
 - ◆ Represent a slice of an architectural entity as required in a specific scenario
- ◆ Allocating behavior to structure
 - Place responsibility in component

Key Points - UCM Introduction & Notation

- ◆ UCMs (Use Case Maps) provide an integrated view of behavior and structure and bridge the conceptual gap between requirements and design
- ◆ UCMs are intuitive and easy to learn
- ◆ UCMs provide a gray-box view of the system
- ◆ The basic elements of the UCM notation are paths, start points, end points, responsibilities, static and dynamic stubs, AND/OR forks/joins, waiting places & timers, and components
- ◆ UCMs may provide reusable behavioral patterns even when the architecture evolves

Table of Contents

- ◆ Introduction to Use Case Maps (UCMs) and the UCM Notation
- ◆ **Use Case Maps Exercise**
- ◆ Enhancing UCMs with Formal Scenario Definitions
- ◆ Tool Demonstration: UCMNAV - The UCM Navigator

Elevator Control System - Exercise

- ◆ Part 1: Create UCM for “Select Destination” use case
- ◆ Part 2: Create UCM for “Request Elevator” use case
- ◆ Part 3: Integrate both use cases into one UCM
- ◆ Use the following components for part 1, 2, and 3
 - User, Arrival Sensor (i.e. the actors)
 - Elevator Control System
- ◆ Part 4: Create one UCM with an alternative architecture

Solution

Elevator Control System - Select Destination Use Case

- ◆ **Actors:** Elevator User (primary), Arrival Sensor
- ◆ **Precondition:** User is in the elevator.
- ◆ **Description:**
 - User presses an elevator button for a floor above the current position. The elevator button sensor sends the user request to the system, identifying the destination floor the user wishes to visit.
 - The new request is added to the list of floors to visit. If the elevator is stationary, the system determines in which direction the system should move in order to service the next request. The system commands the elevator door to close. When the door has closed, the system commands the motor to start moving the elevator, either up or down.
 - As the elevator moves between floors, the arrival sensor detects that the elevator is approaching a floor and notifies the system. The system checks whether the elevator should stop at this floor. If so, the system commands the motor to stop. When the elevator has stopped, the system commands the elevator door to open.
 - If there are other outstanding requests, the elevator visits these floors on the way to the floor requested by the user. Eventually, the elevator arrives at the destination floor selected by the user.
- ◆ **Alternatives:**
 - User presses an elevator button for a floor below the current position to move down. System response is the same as for the main sequence.
 - If the elevator is at a floor and there is no new floor to move to, the elevator stays at the current floor, with the door open.
- ◆ **Postcondition:** Elevator has arrived at the destination floor selected by the user.

Elevator Control System - Request Elevator Use Case

- ◆ **Actors:** Elevator User (primary), Arrival Sensor
- ◆ **Precondition:** User is at a floor and wants an elevator.
- ◆ **Description:**
 - User presses an up floor button. The floor button sensor sends the user request to the system, identifying the floor number.
 - The system selects an elevator to visit this floor. The new request is added to the list of floors to visit. If the elevator is stationary, the system determines in which direction the system should move in order to service the next request. The system commands the elevator door to close. After the door has closed, the system commands the motor to start moving the elevator, either up or down.
 - As the elevator moves between floors, the arrival sensor detects that the elevator is approaching a floor and notifies the system. The system checks whether the elevator should stop at this floor. If so, the system commands the motor to stop. When the elevator has stopped, the system commands the elevator door to open.
 - If there are other outstanding requests, the elevator visits these floors on the way to the floor requested by the user. Eventually, the elevator arrives at the floor in response to the user request.
- ◆ **Alternatives:**
 - User presses down floor button to move down. System response is the same as for the main sequence.
 - If the elevator is at a floor and there is no new floor to move to, the elevator stays at the current floor, with the door open.
- ◆ **Postcondition:** Elevator has arrived at the floor in response to user request.

Table of Contents

- ◆ Introduction to Use Case Maps (UCMs) and the UCM Notation
- ◆ Use Case Maps Exercise
- ◆ **Enhancing UCMs with Formal Scenario Definitions**
- ◆ Tool Demonstration: UCMNAV - The UCM Navigator

Scenario Definitions

- ◆ Enhances the behavioral modeling capability of UCM paths and path elements
- ◆ Requires a **path data model** (for conditions at various points along the path)
 - Currently, global and modifiable Boolean variables
 - ◆ Values may be assigned to variables along a path
 - In future, ...
 - ◆ Variables may possibly have different types
 - ◆ Variables may be scoped to paths or components
 - ◆ Scenarios may be structured into sub-scenarios

Scenario Definitions

- ◆ Requires a more formal definition of some notational elements
 - Currently, logical expressions with global variables
 - Currently, OR forks, selection policies, start points, waiting places, & timers covered (in future: loops)
- ◆ Scenario definitions consist of ...
 - Name of scenario (scenarios may be grouped for convenience)
 - Set of concurrent start points
 - Set of initial values assigned to global variables

Scenario Highlight (UCMNAV 2)

The screenshot displays the UCMNAV 2 interface. The main window shows a UCM diagram with a path highlighted in red. A 'Scenario Definitions' dialog box is open, showing a list of scenarios. The 'Scenarios' tab is selected, and the 'OCS_CNDdisplay' scenario is highlighted. The dialog includes fields for 'Description of Scenario Group' and 'Description of Scenario'. Below these are sections for 'Boolean Variables (unreferenced)', 'Variable Initializations', and 'Scenario Starting Point'. The 'Scenario Starting Point' section includes a 'Path Start: req Map: root' field and buttons for 'Add Selected Path Start', 'Delete Specified Scenario Start', 'Generate MSC', 'Highlight Path', and 'Duplicate Scenario'. The background shows a UCM diagram with a path highlighted in red.

Key Points - Scenario Definitions

- ◆ Path data model is not a problem domain data model
- ◆ Improves understanding of scenarios
- ◆ Scenario definitions are the foundation for more advanced functionality such as ...
 - Highlighting of a scenario ... available
 - Animation of a scenario ... just a small step
 - Generation of MSCs ... available
 - Generation of test cases ... just a small step
 - Detection of feature interactions ... early results
 - Execution of UCMs ... future research

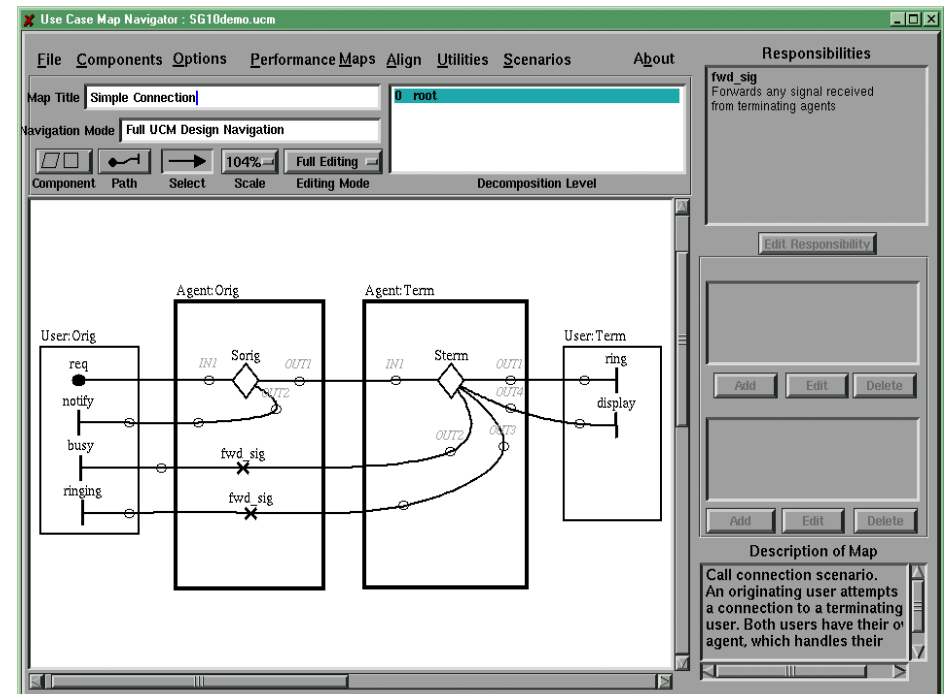
Table of Contents

- ◆ Introduction to Use Case Maps (UCMs) and the UCM Notation
- ◆ Use Case Maps Exercise
- ◆ Enhancing UCMs with Formal Scenario Definitions

◆ Tool Demonstration: UCMNAV - The UCM Navigator

UCMNAV

- ◆ Developed by Andrew Miga (Carleton U.) since 1997
- ◆ Editing and navigating of UCMs
- ◆ Supports UCM path and component notations
- ◆ Maintains bindings
 - Plug-ins to stubs, responsibilities to components, sub-components to components, etc.
- ◆ Editing is transformation-based
 - Operations maintain syntactic correctness and enforce some static semantics constraints



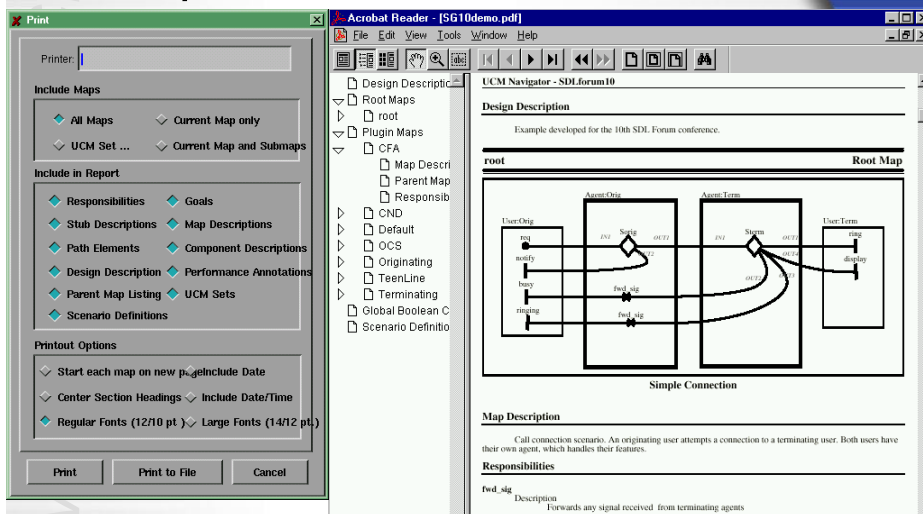
UCMNAV Facts

- ◆ Load/save/import/export in XML
- ◆ Developed in C++, GUI in Xforms
- ◆ Requires an X-server
 - E.g. Xfree86 freeware (<http://xfree86.cygwin.com/>)
- ◆ Multiple platforms are currently supported
 - Solaris, Linux (Intel and Sparc), HP/UX, and Windows (95, 98, 2000 and NT)
- ◆ Current stable version: 1.13.4
 - Freely available at <http://www.UseCaseMaps.org>
- ◆ Beta version: 2.0.0
 - Scenario definition, path highlighting, MSC generation

UCM Documents

- ◆ XML (conforms to UCM DTD)
- ◆ Export of UCM figures
 - Encapsulated PostScript (EPS)
 - Maker Interchange Format (MIF)
 - Computer Graphics Metafile (CGM)
 - ◆ Used extensively in this tutorial
 - Scalable Vector Graphics (SVG) - for 2.0.0
- ◆ Flexible report generation
 - Content options
 - PostScript, with PDF hyperlink information

Report Generation in PS/PDF



The screenshot shows the UCM Navigator interface. On the left, there are several panels for report generation: 'Include Maps' (All Maps, Current Map only, UCM Set, Current Map and Submaps), 'Include in Report' (Responsibilities, Goals, Stub Descriptions, Map Descriptions, Path Elements, Component Descriptions, Design Description, Performance Annotations, Parent Map Listing, UCM Sets, Scenario Definitions), and 'Printout Options' (Start each map on new page, include Date, Center Section Headings, Include Date/Time, Regular Fonts (12/10 pt), Large Fonts (14/12 pt)). The main window displays a diagram titled 'Simple Connection' showing a sequence of messages between 'User-Orig' and 'User-Term' through 'Agent-Orig' and 'Agent-Term'. Below the diagram, there is a 'Map Description' section with a call connection scenario and a 'Responsibilities' section for 'fwd_sig'.

Main References

- ◆ Web site: <http://www.UseCaseMaps.org/>
 - Amyot, D., *Use Case Maps Quick Tutorial Version 1.0*, 1999.
 - Buhr, R.J.A., *Use Case Maps as Architectural Entities for Complex Systems*, In: Transactions on Software Engineering, IEEE, Vol. 24, No. 12, December 1998, pp. 1131-1155.
 - Buhr, R.J.A. and Casselman, R.S., *Use CASE Maps for Object-Oriented Systems*, Prentice Hall, 1996.