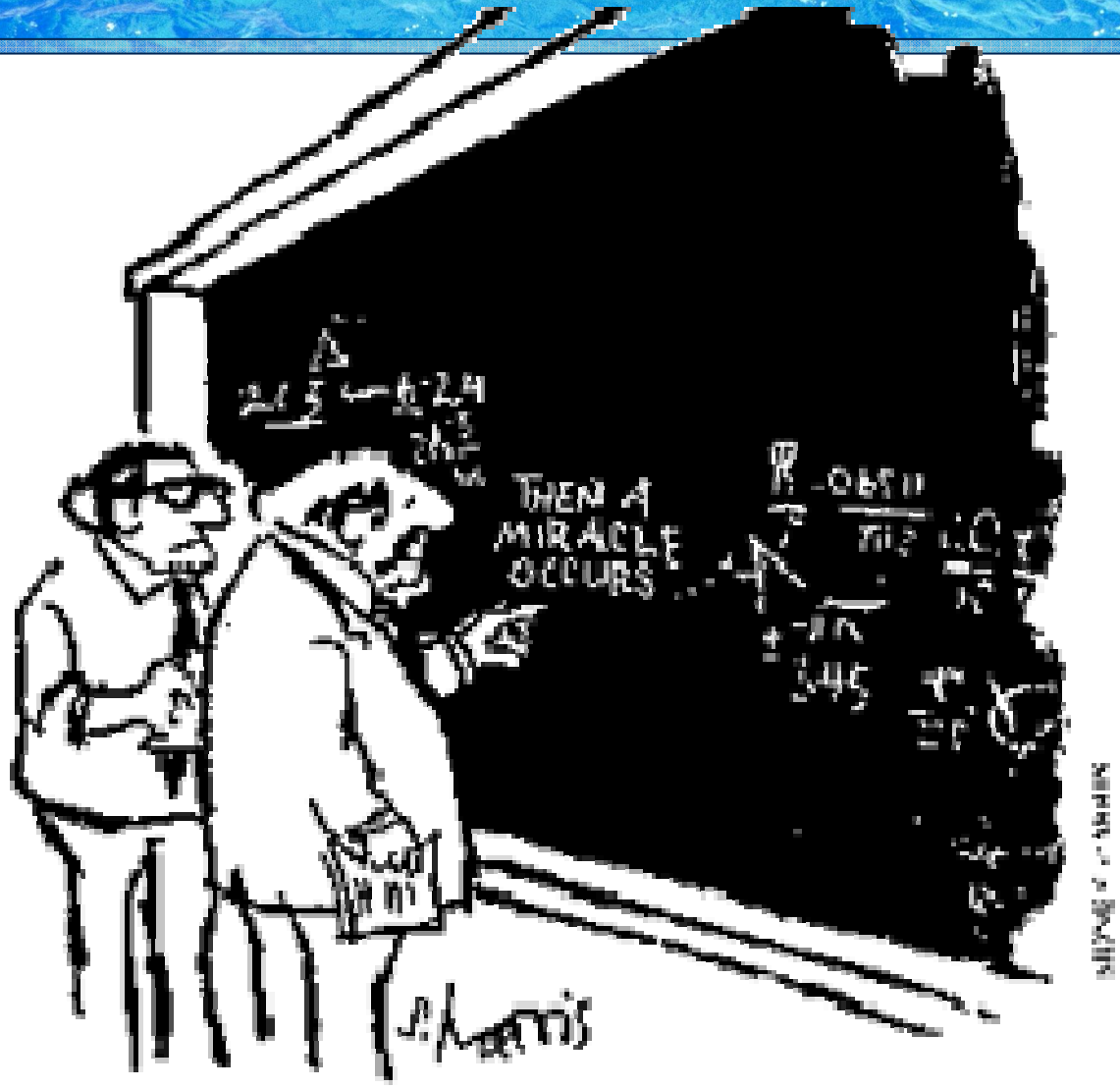
User Requirements Notation (Part II)

Gunter Mussbacher, University of Ottawa

Based on material from:
Mussbacher and Amyot 2009

Table of Contents



"I THINK YOU SHOULD BE
MORE EXPLICIT HERE IN STEP TWO"

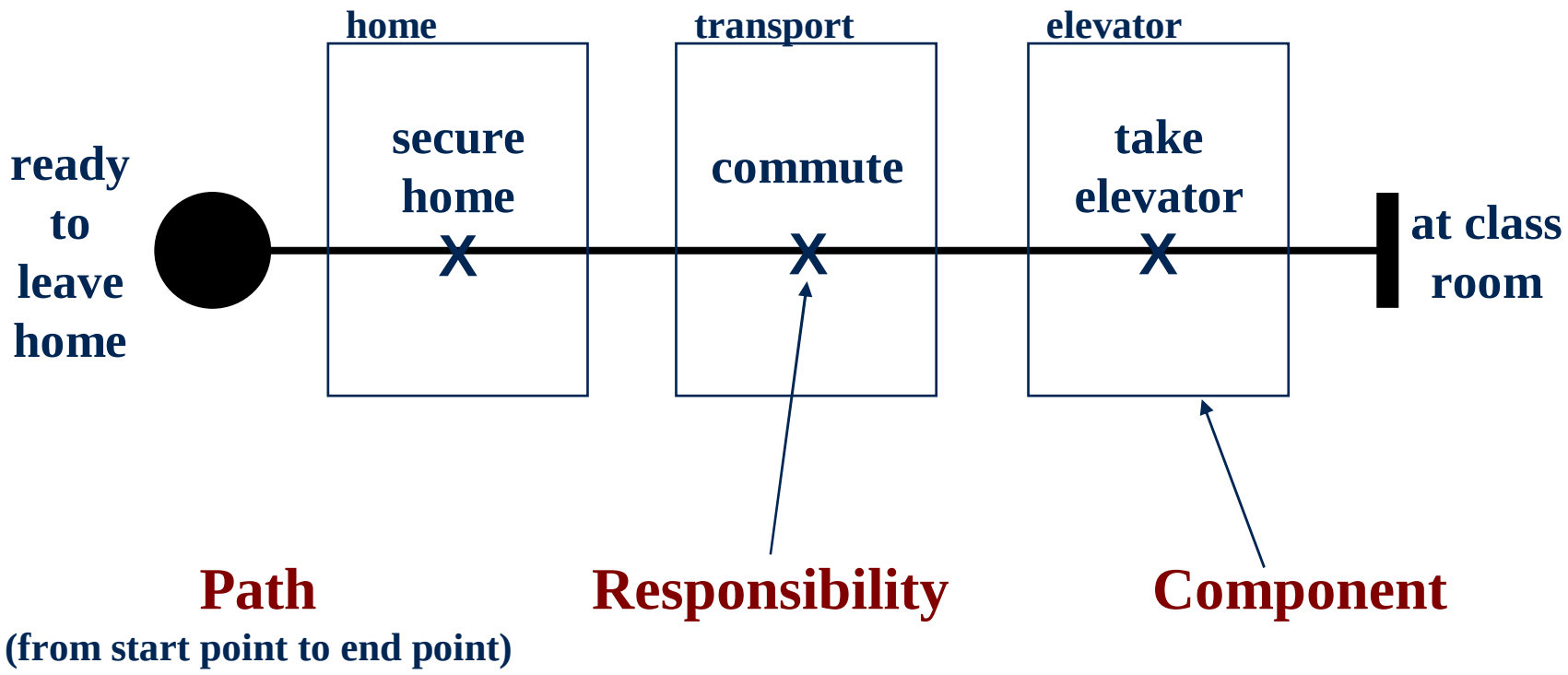
Use Case Maps (UCM)

Use Case Maps Overview

- Use Case Maps
  - Graphical scenario notation
  - Causal relationships between responsibilities
  - Scenario elements may (optionally) be allocated to components
- UCMs model the "what" aspects
  - Functional requirements as scenarios
  - Integration and reusability of scenarios
  - Guidance for architecture and detailed behavior
- Conflict detection
- Transformations
- Performance analysis

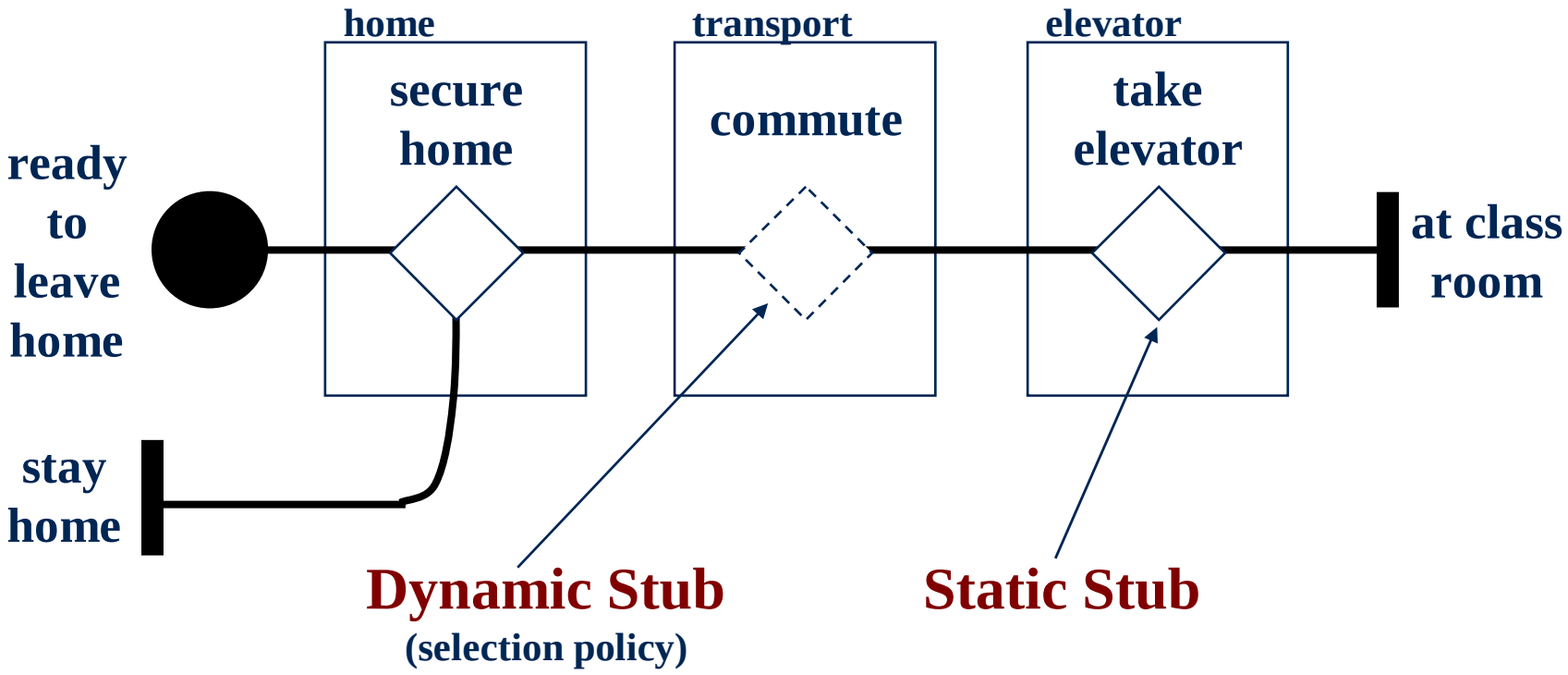
Use Case Maps Notation – Basic

UCM Example: Commuting



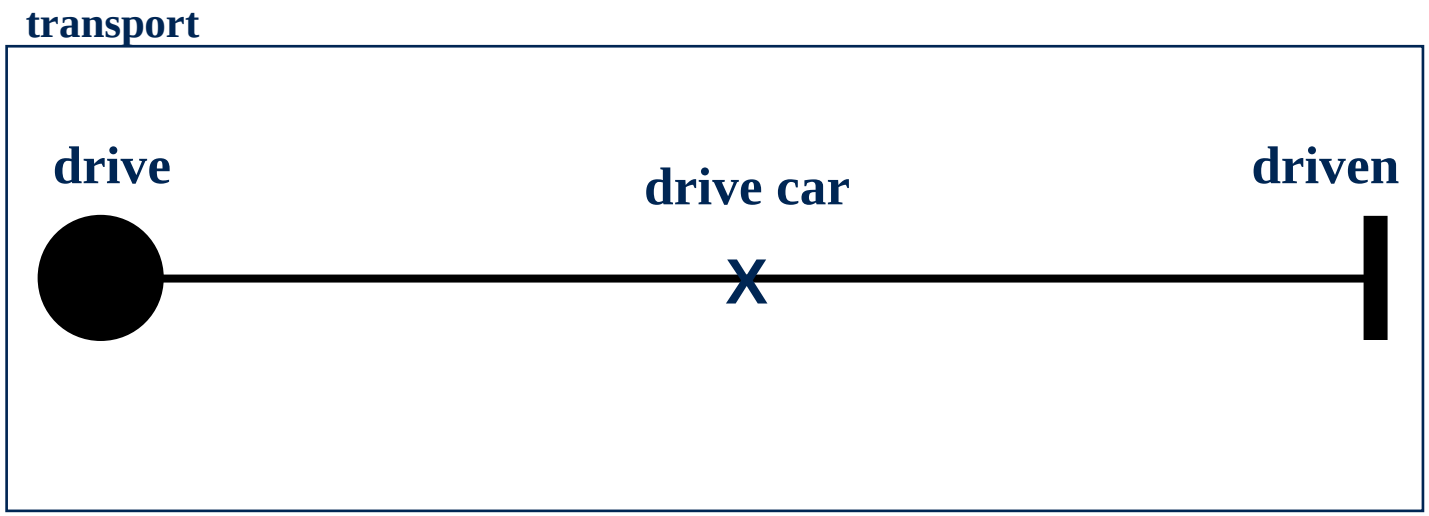
Use Case Maps Notation – Hierarchy

UCM Example: Commuting



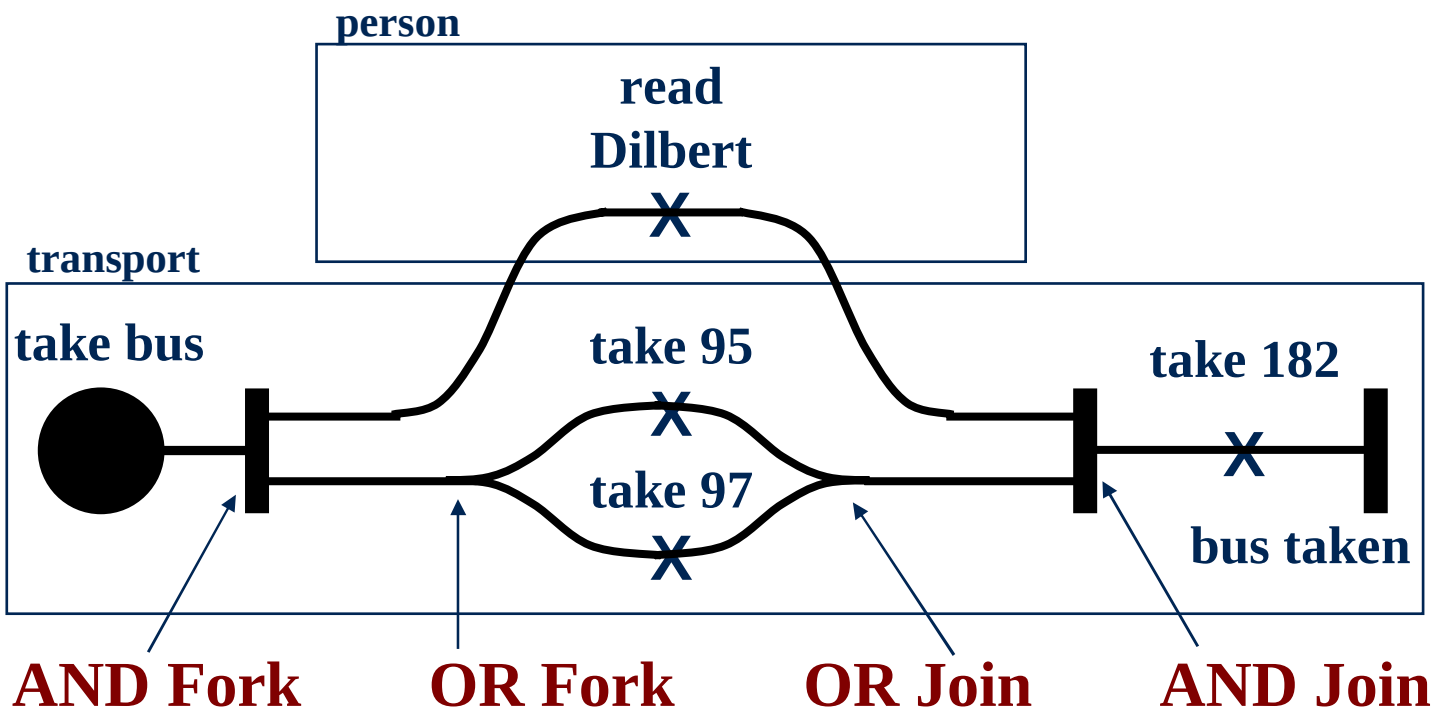
Use Case Maps Notation – Simple Plug-in

UCM Example: Commute - Car (Plug-in)



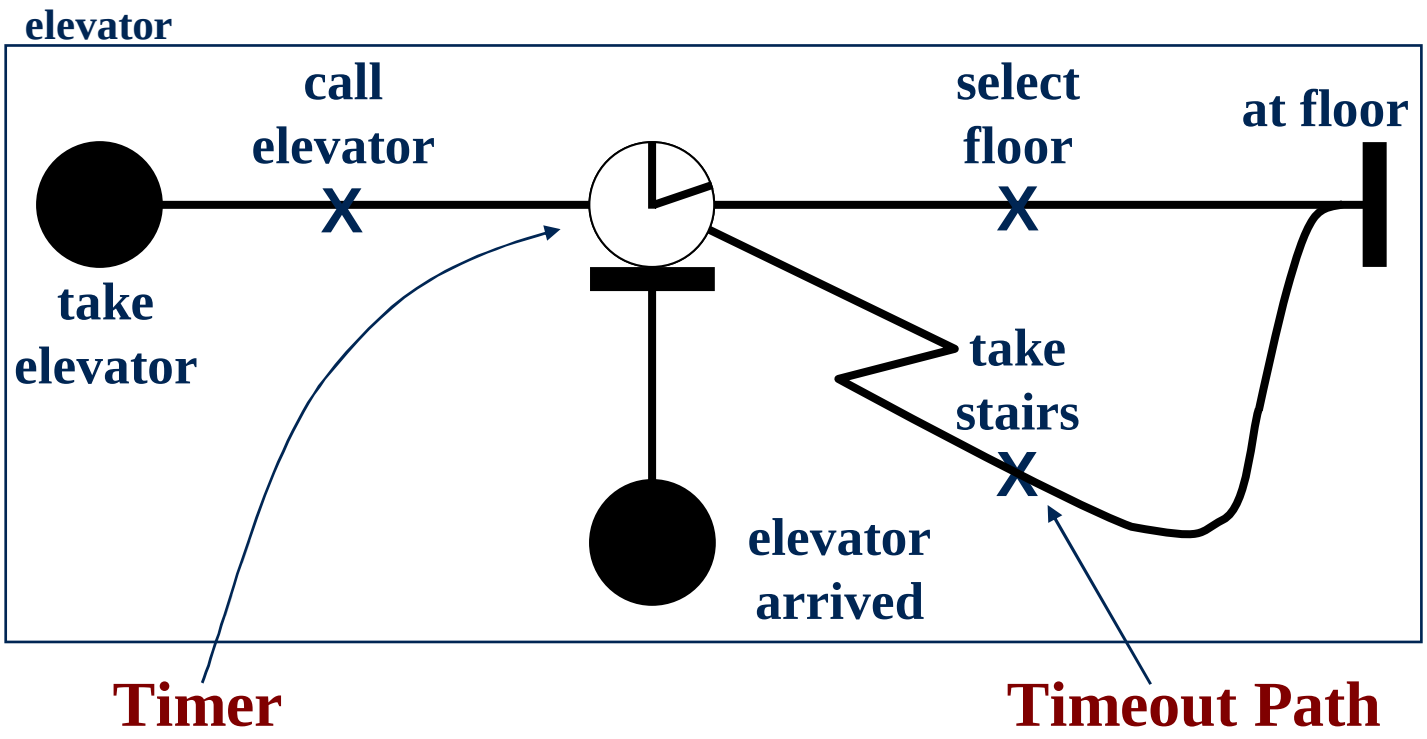
Use Case Maps Notation – Parallel / Alternatives

UCM Example: Commute - Bus (Plug-in)



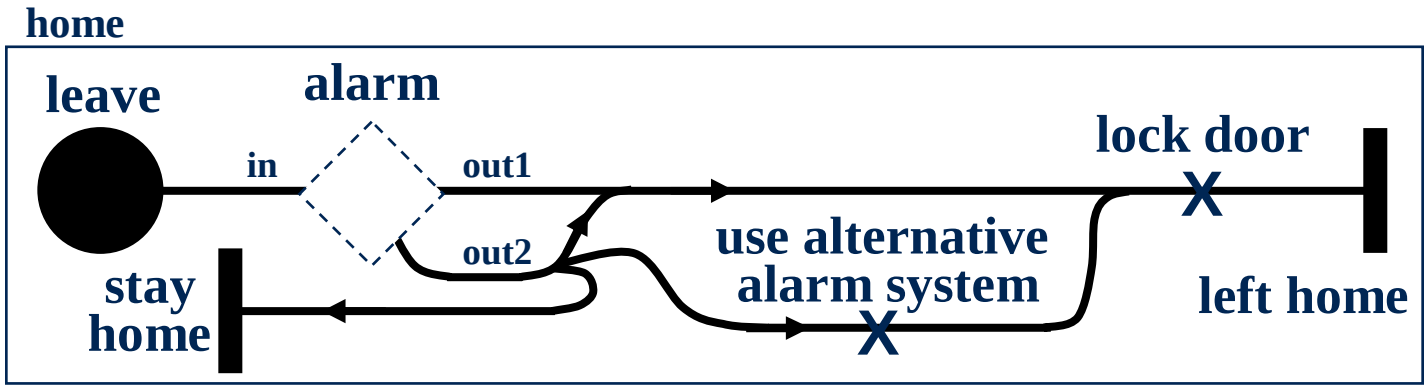
Use Case Maps Notation – Waiting Place / Timer

UCM Example: Take Elevator (Plug-in)



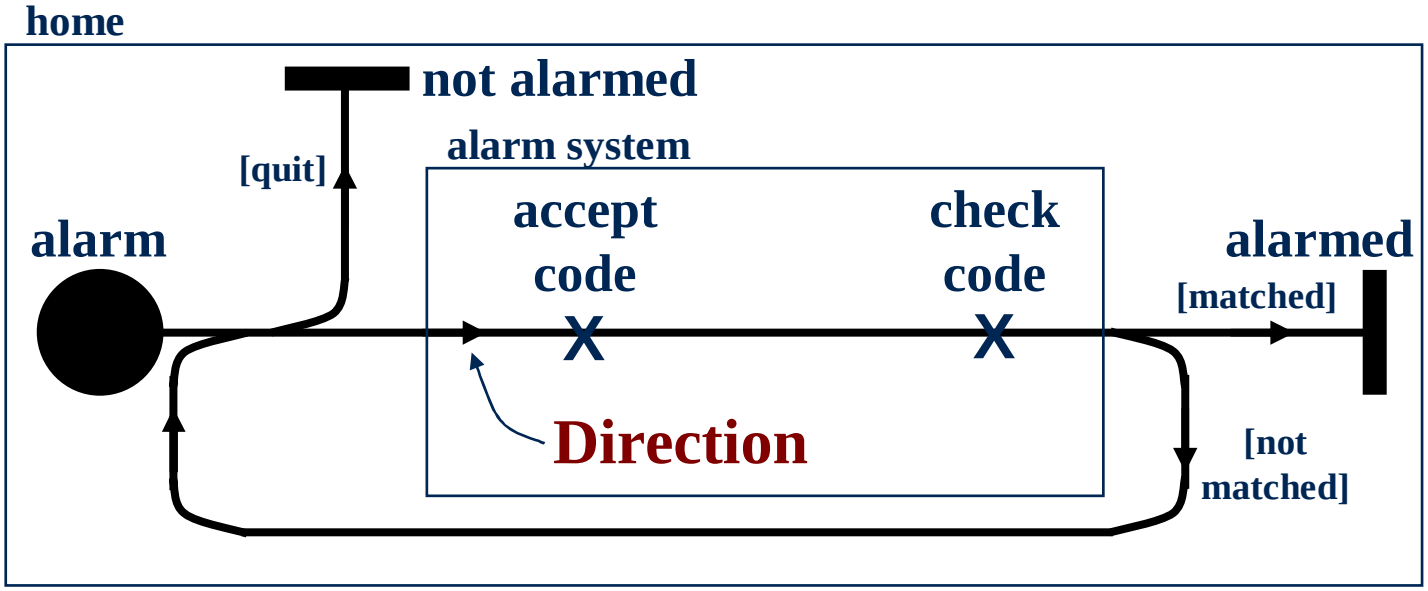
UCM Notation – Simple Plug-in with Stub

UCM Example: Secure Home (Plug-in)

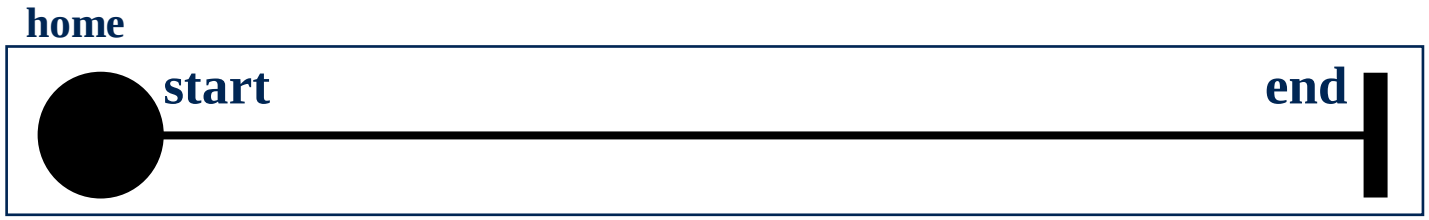


Use Case Maps Notation – Plug-ins at 2nd Level

UCM Example: Alarm - Installed (Plug-in)



UCM Example: Alarm – Not Installed (Plug-in)

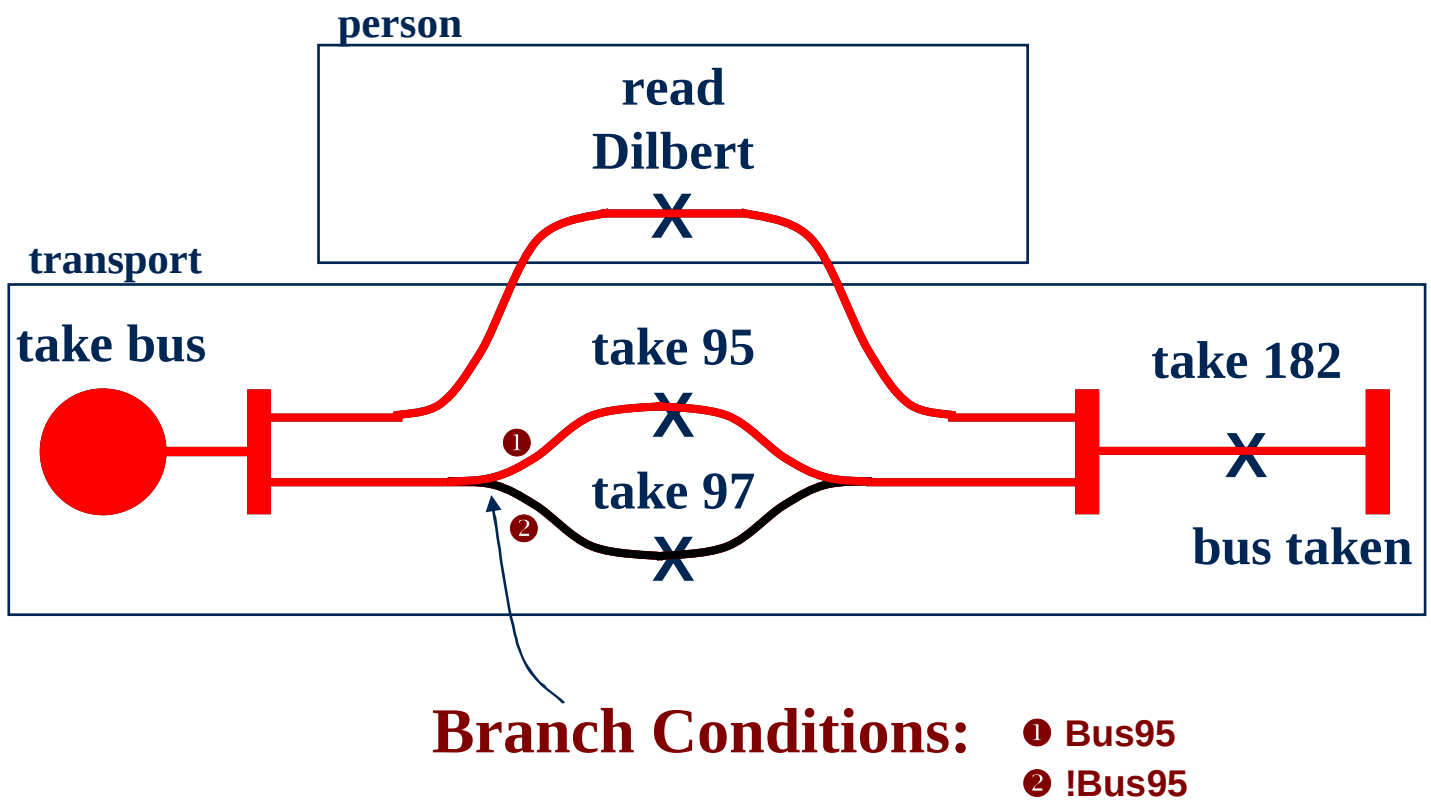


Use Case Maps – Scenario Execution (1)

- Scenarios start at start point “take bus”, end at end point “bus taken”

1st Scenario Definition:
Bus95 = false;

2nd Scenario Definition:
Bus95 = true;



Use Case Maps – Scenario Execution (2)

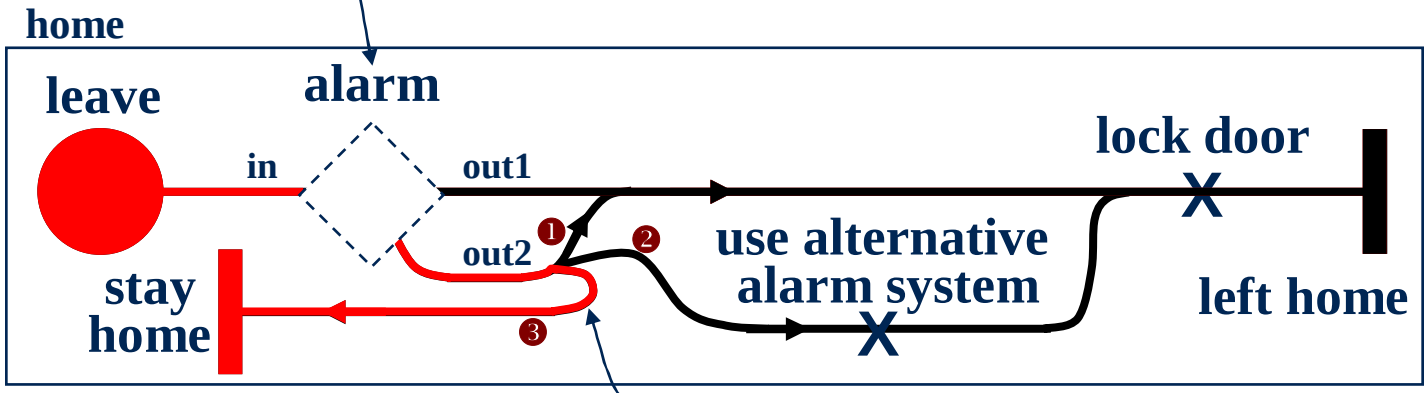
- 3rd Scenario Definition: starts at start point “leave”, ends at end point “left home”

3rd Scenario Definition:
Installed = true;
LeaveAnyway = false;
StayHome = false;
UseAlternativeAlarm = false;

Selection Policy:

Plug-in «Alarm – Installed»: Installed
Plug-in «Alarm – Not Installed»: !Installed

UCM Example: Secure Home (Plug-in)



Branch Conditions:

❶ LeaveAnyway
❷ UseAlternativeAlarm
❸ StayHome

4th Scenario Definition:
Installed = true;
LeaveAnyway = false;
StayHome = true;
UseAlternativeAlarm = false;

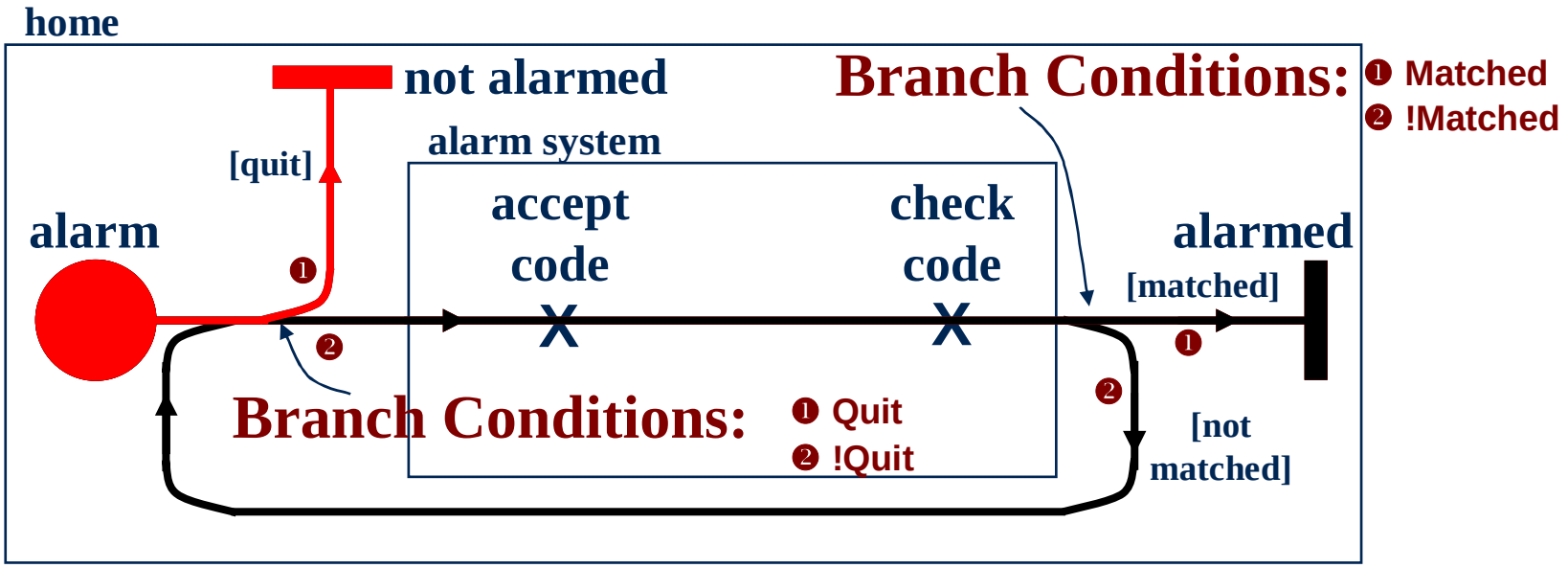
- 4th Scenario Definition: starts at start point “leave”, ends at end point “stay at home”

Use Case Maps – Scenario Execution (3)

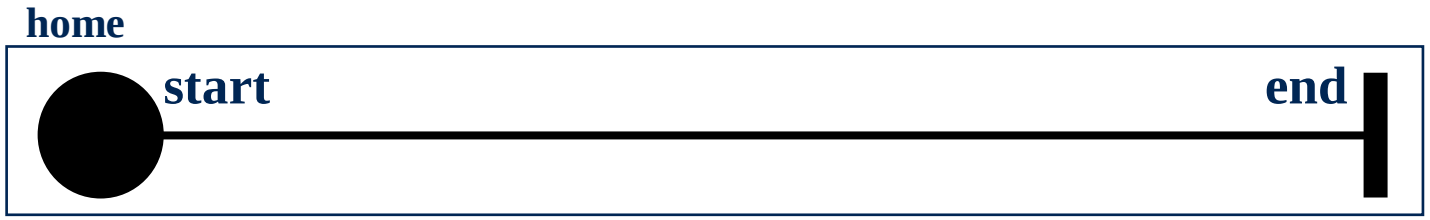
3rd Scenario
Definition:
Matched = true;
Quit = false;

4th Scenario
Definition:
Matched = true;
Quit = true;

UCM Example: Alarm - Installed (Plug-in)

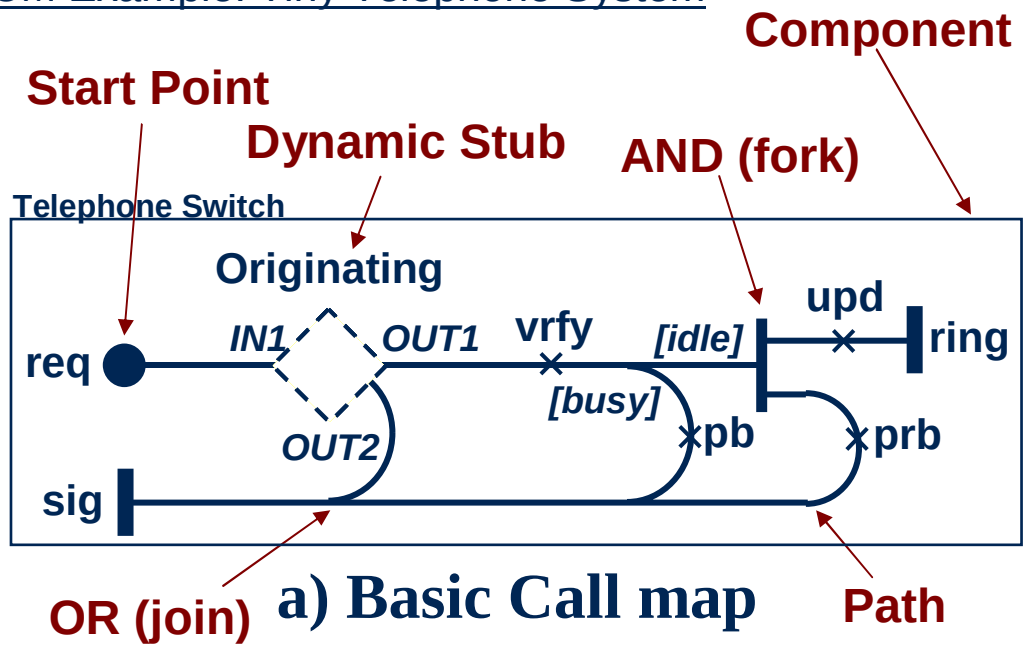


UCM Example: Alarm – Not Installed (Plug-in)

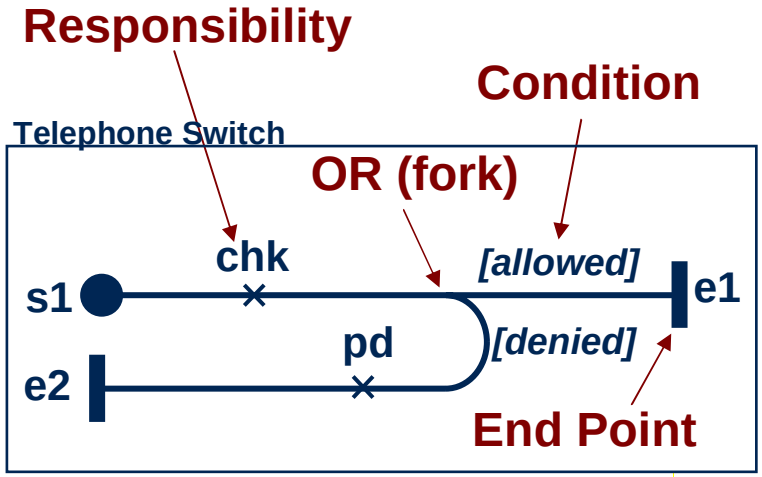


Use Case Maps Notation – Summary

UCM Example: Tiny Telephone System



a) Basic Call map



b) *OCS* plug-in map

c) *default* plug-in map

AND (join)

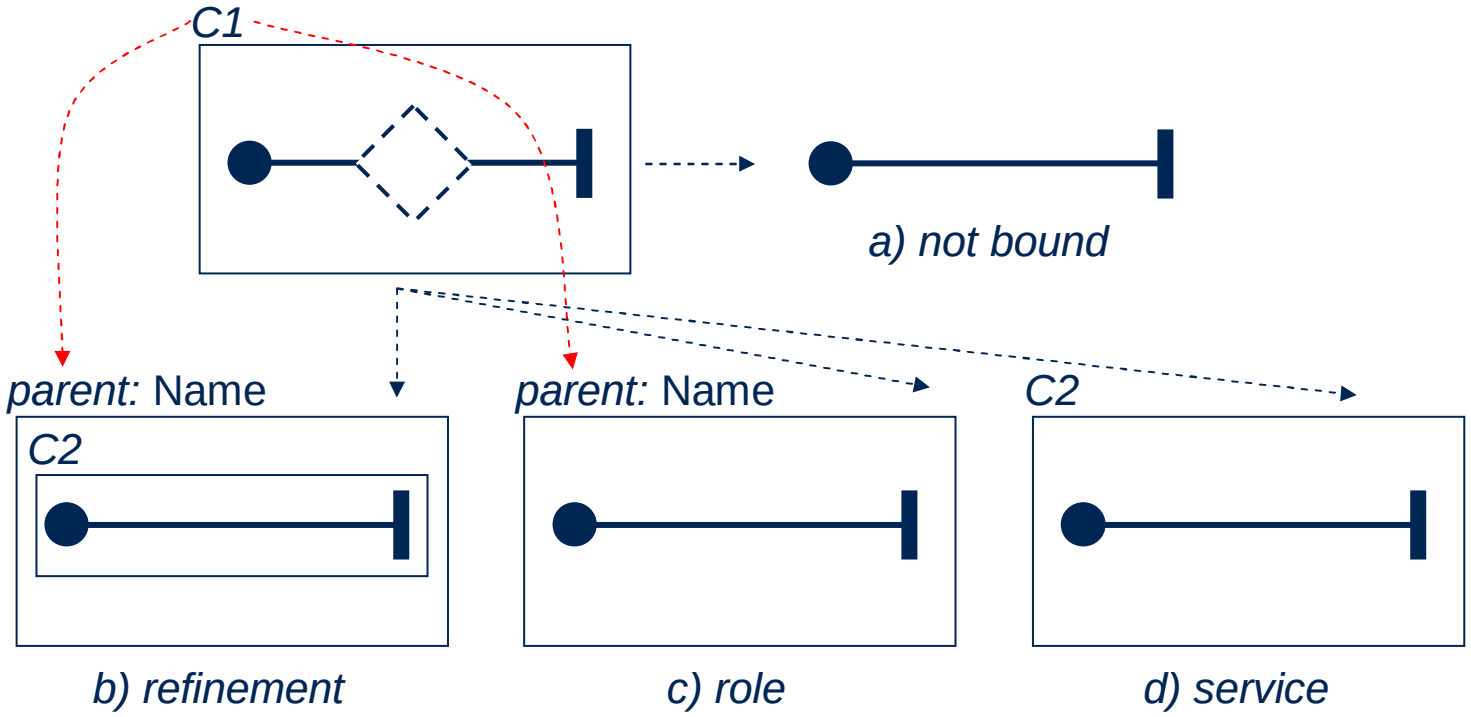
waiting place

timer

Static Stub (at most one plug-in map)

UCM Notation – Component Plug-in Bindings

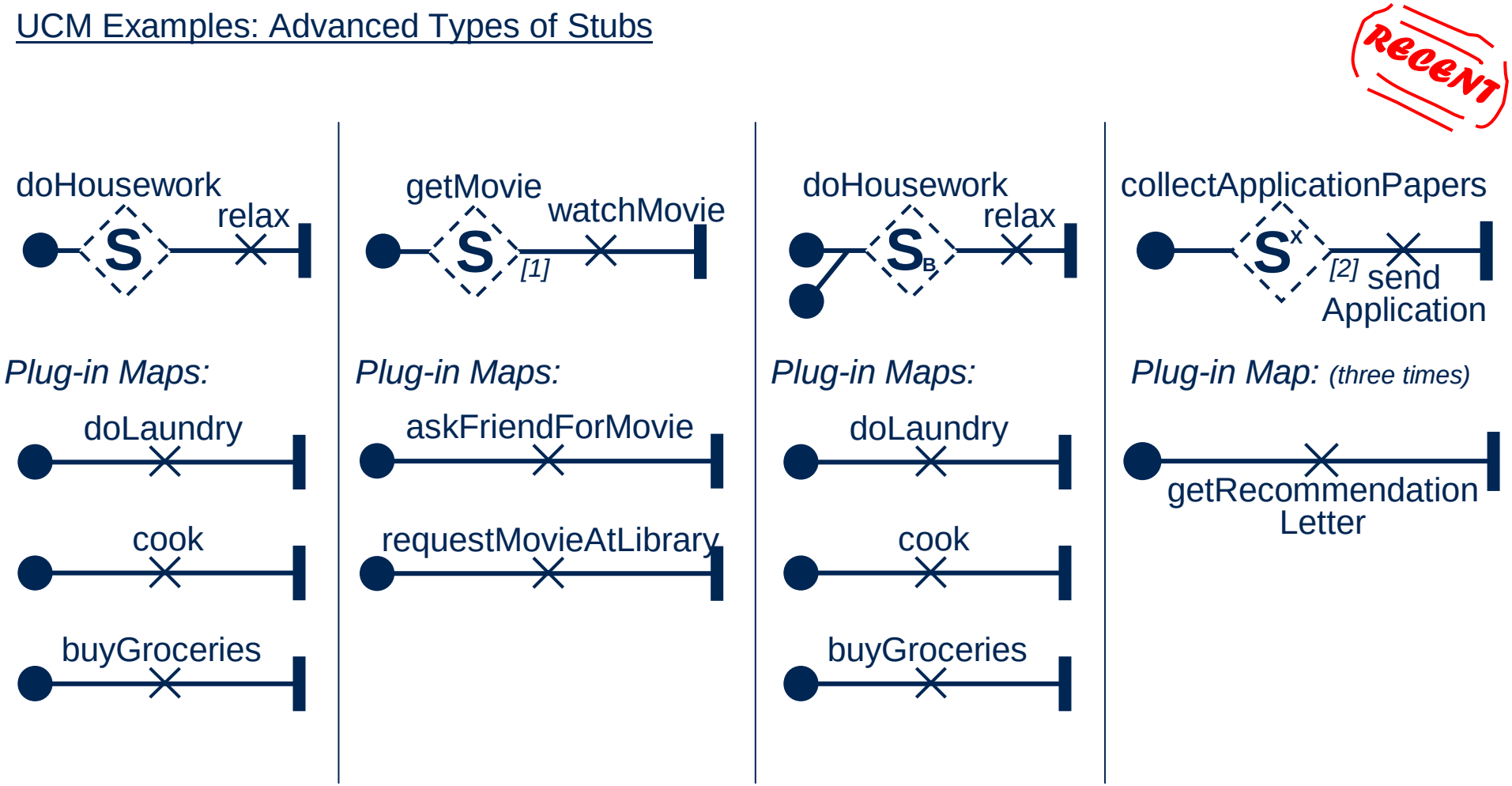
- Establishes relationship of component on parent map with component on plug-in map (in addition to bindings of stub's in/out-paths with start/end points on a plug-in map)



- Option c: the parent component plays a role e.g. in an architectural or behavioral pattern

Use Case Maps Notation – Advanced Stubs

UCM Examples: Advanced Types of Stubs

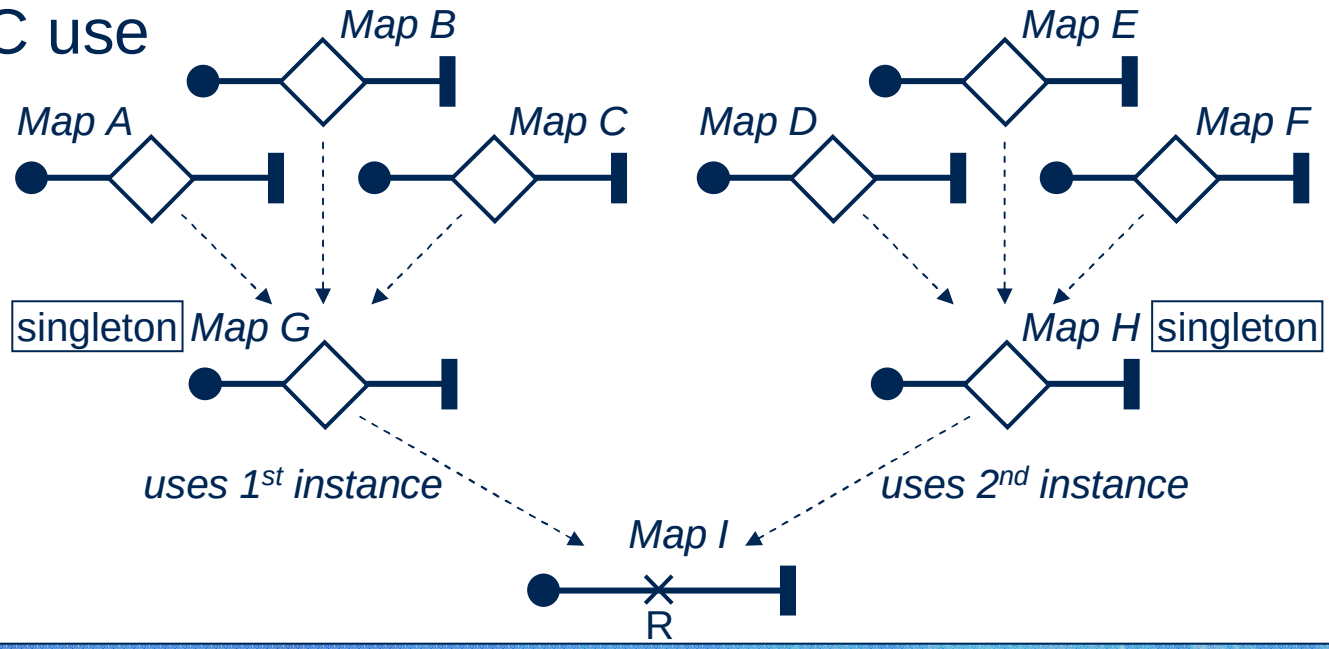


Synchronizing Stub with synchronization threshold

Blocking Stub with replication indicator

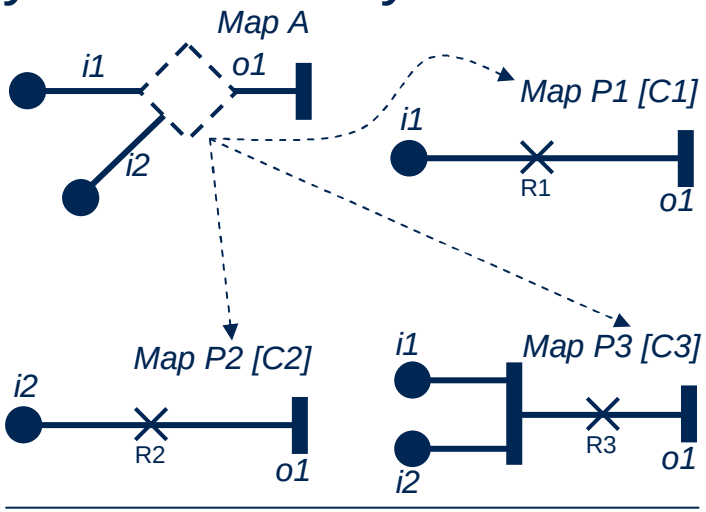
Use Case Maps Notation – Singleton Maps

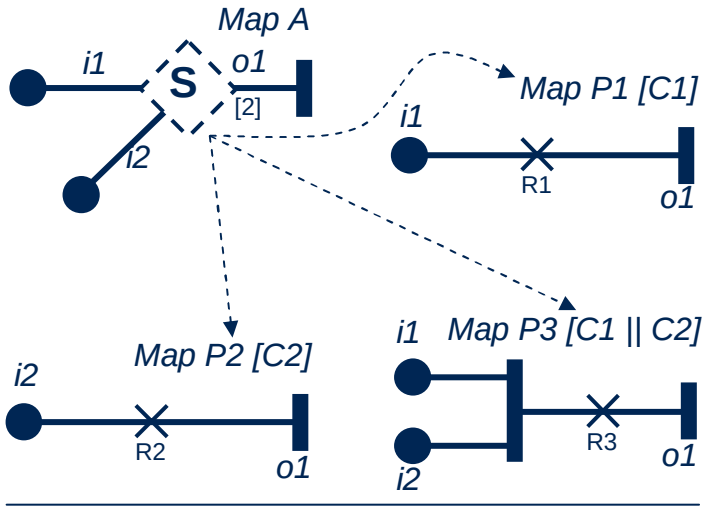
- Case 1: Map G is a singleton and therefore the stubs on Map A, B, and C use the same instance of Map G
- Case 2: Map I is not a singleton and therefore the stubs on Map G and Map H use different instances of Map I
- Case 3: A groups of stubs may want to use the same instance of a plug-in map but a different instance than other stubs. Achieved with an intermediate layer (e.g., the stubs on Map A, B, and C use the same instance of Map I, but the stubs on Map D, E, and F use a different instance of Map I)

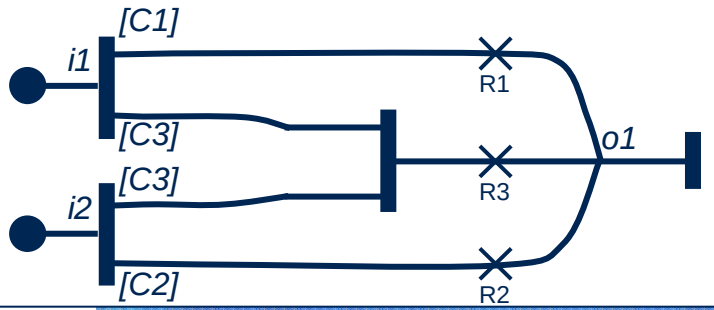


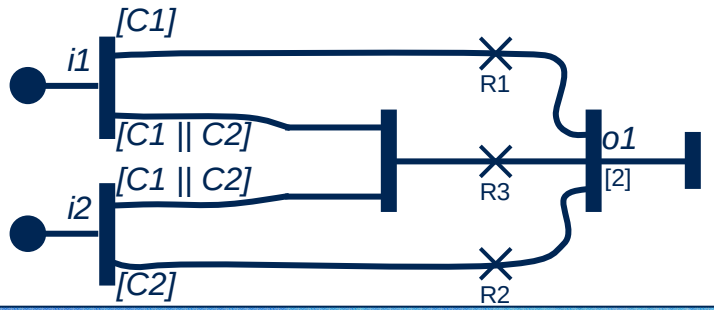
Use Case Maps – Semantics (1)

- OR-fork: exclusive OR
- AND-fork: always all paths
- Protected component: only one active path at a time
- Dynamic and synchronizing stubs:









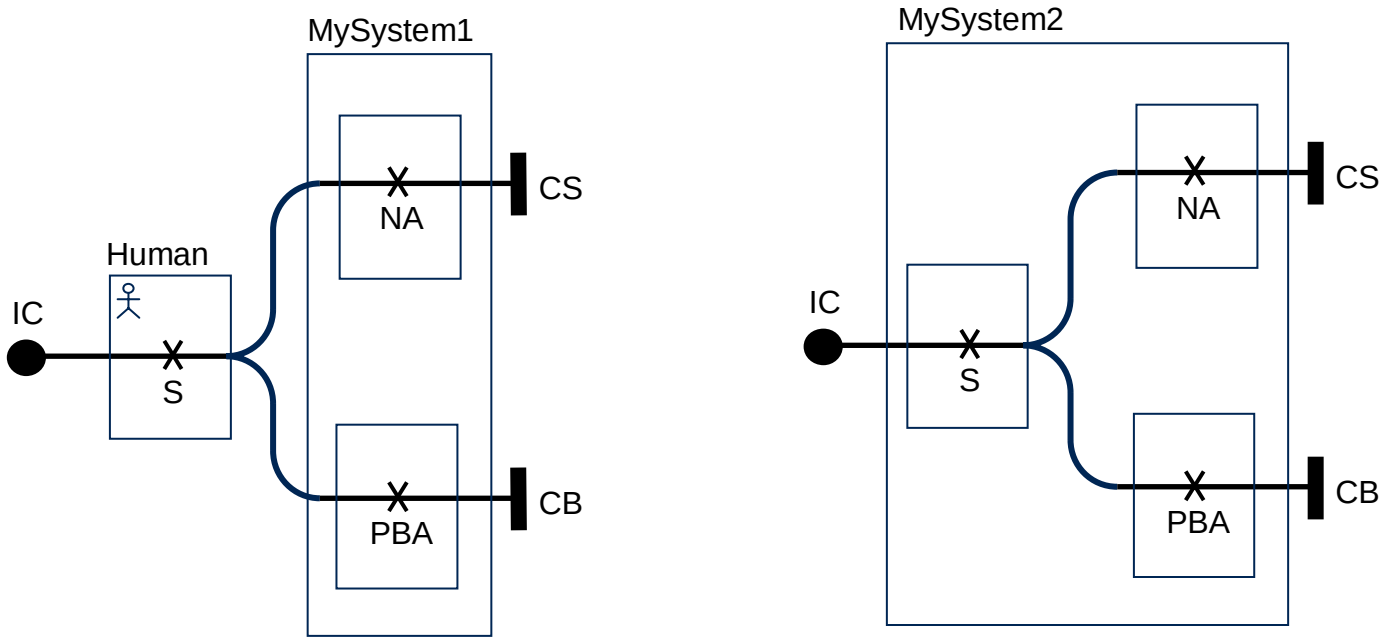
Use Case Maps – Semantics (2)

- Waiting place and timer
  - Transient = a trigger is counted only if a scenario is already waiting
  - Persistent = all triggers are counted (i.e., remembered)
- Waiting place
  - Scenario is allowed to continue if CW = true or the trigger counter > 0
  - Warning is issued if CW = false and the trigger counter = 0
- Timer: Scenario is allowed to continue along …
  - Regular path if CT = true
  - Regular path if CT = false and CTO = false and trigger counter > 0
  - Timeout path if CT = false and CTO = true
  - Timeout path if CT = false and CTO = false and trigger counter = 0

Use Case Maps Summary

- Model scenario concepts – mainly for operational and functional requirements
- Use Case Maps (UCMs) provide …
  - Visual description of behavior superimposed over entities (from software architecture to actors to hardware)
  - Easy graphical manipulation of use cases/scenarios
  - Fairly simple, intuitive, low learning curve
  - Enhanced consistency and completeness
  - Single use case/scenario view
  - Combined system view – use case maps integrate many scenarios
    - Enables reasoning about potential undesirable interactions of scenarios
  - Convey a lot of information in a compact form
  - Effective learning tool for people unfamiliar with the domain
  - Document while you design

Where is the System Boundary?



- Same scenarios!
- May assign responsibilities to system under design or to external entities (e.g., human or other systems)
- May assign start/end points to actors (human or machine)
- UCM *actors* (웃) supported in language

Why Use Case Maps?

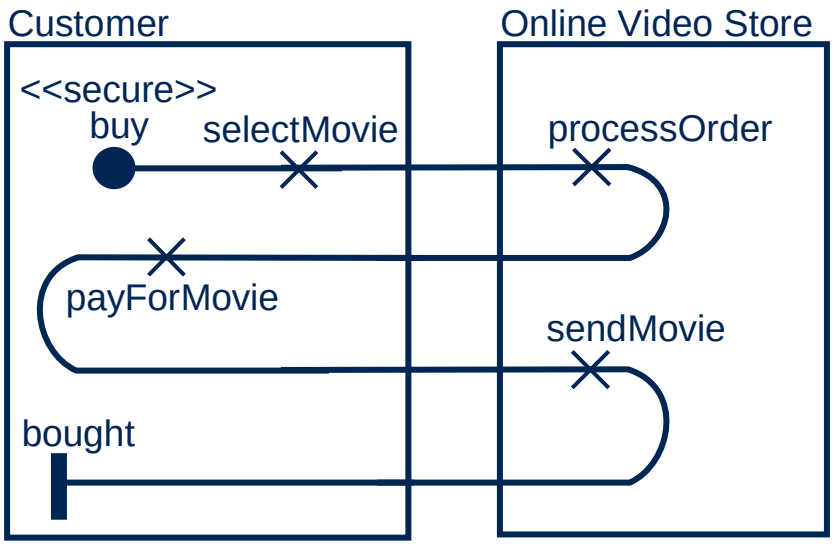
- Bridge the modeling gap between use cases, requirements, and design
  - Link behavior and structure in an explicit and visual way
  - Provide a behavioral framework for making (evaluating) architectural decisions at a high level of design → architectural reasoning
  - Characterize the behavior at the architecture level once the architecture is decided
- Provide ability to model dynamic systems where scenarios and structures may change at run-time
  - E-commerce applications, Web services
  - Distributed systems based on agents
- May be transformed
  - Smooth transition to design models (e.g., MSC/ sequence diagrams)
  - Connections to performance models and testing models

UCM Transformations

- Why
  - Different notations are more suitable at different points of the development cycle
  - Reuse scenario information to bridge the gap between phases
  - In the spirit of OMG's Model Driven Architecture (MDA) vision
    - Platform-independent model → Platform-specific model
- How
  - Use Case → UCM
  - UCM → MSC and UML sequence diagrams, for design
  - UCM → LQN, for performance analysis
  - UCM → LOTOS and SDL, for requirements prototyping and validation
  - UCM → TTCN-3, for system-level testing
  - UCM → UML2, UCM → Petri Nets
  - Code → UCM (OMG's Architecture-Driven Modernization)

Extensibility: Metadata

- URN can be extended with metadata (name/value pairs)
  - Can be attached to any URN model element and exploited by specialized tools

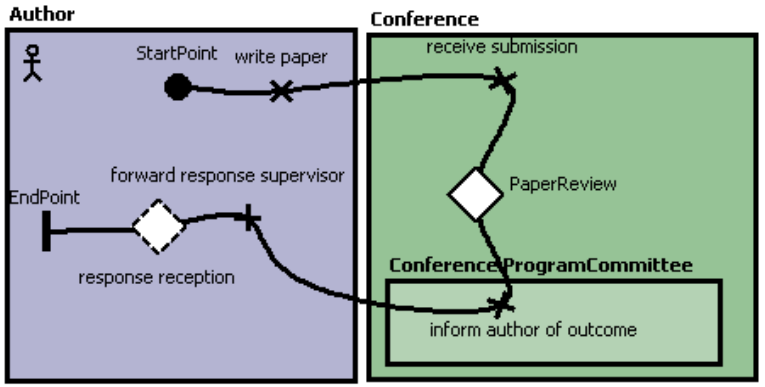


- For example, <<secure>> indicates that the scenario requires secure communication between the Customer and the Online Video Store
- The metadata definition (name = value) is:
  - Communication = secure

From Use Cases to Use Case Maps

- Title: Submit Paper
  1. Author writes a paper
  2. Conference receives submission
  3. INCLUDE Review Paper
  4. Conference Program Committee informs author of outcome
  5. Author forwards response to supervisor

  Extension Point ➔ response reception



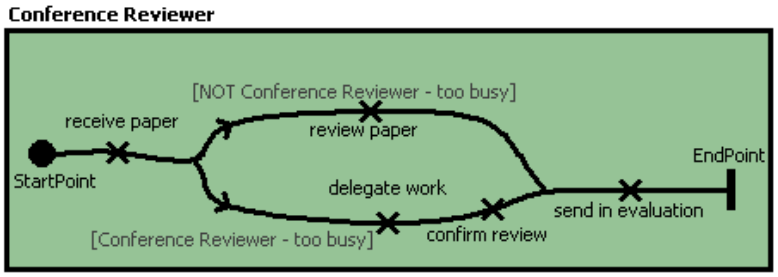
- Title: Review Paper
  1. Conference Reviewer receives paper
  2. Conference Reviewer reviews the paper
  3. Conference Reviewer sends in evaluation

  1.a. Conference Reviewer is too busy

  1.a.1. Conference Reviewer delegates work

  1.a.2. Conference Reviewer confirms review

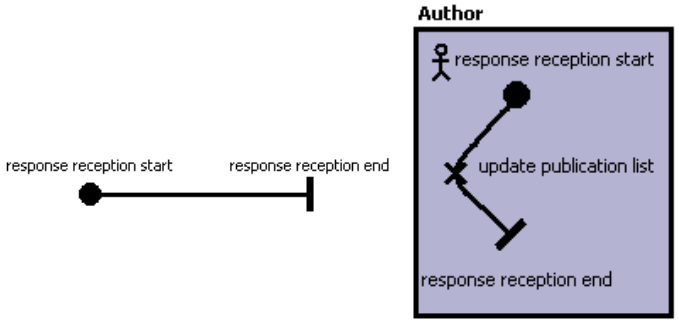
  1.a.3. GOTO 3



- Title: Update Publications

  At Extension Point response reception

  1. Author updates publication list

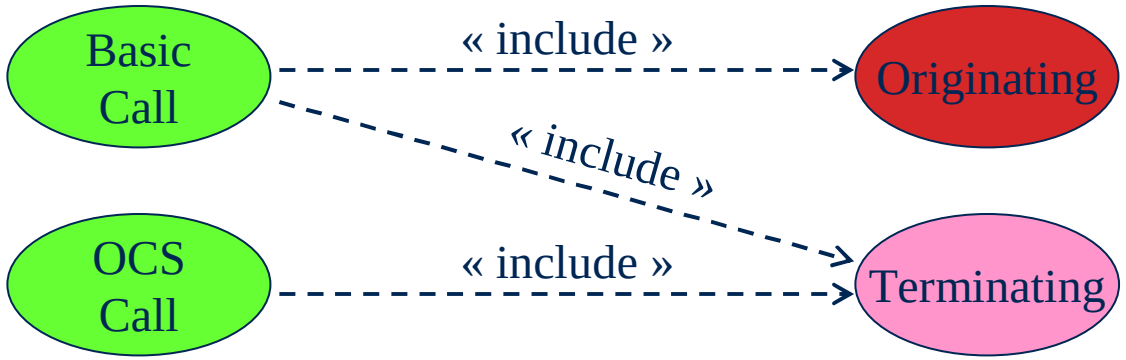


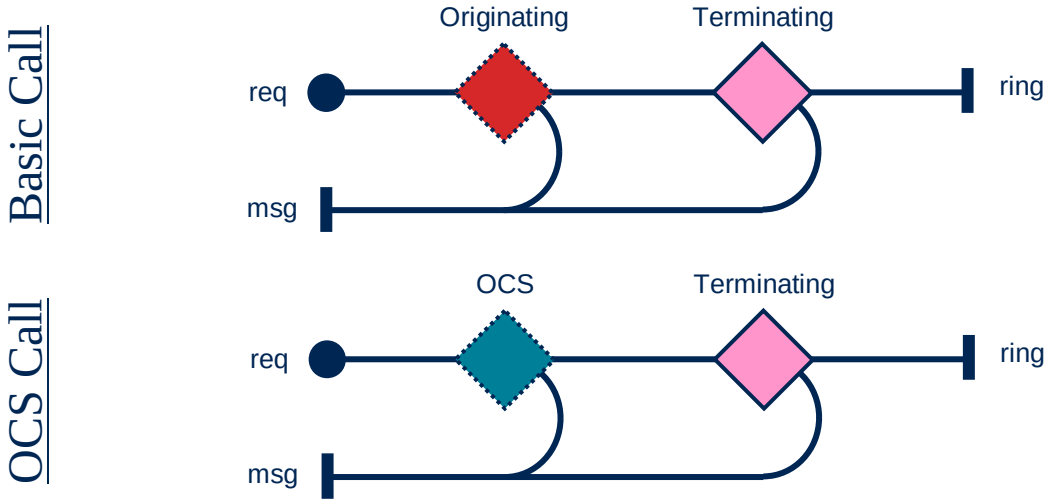
Include Relationship

- Helps clarify a use case by isolating and encapsulating complex details and by improving consistency
- Base use case requires included use case for completion

- *Solution*:
- Use static stubs on the path representing a base use case
  - Stubs hide the details contained in their plug-ins (the included use case)
  - The plug-in can be reused in multiple stubs, hence improving consistency among the UCMs

Include Relationship – Example



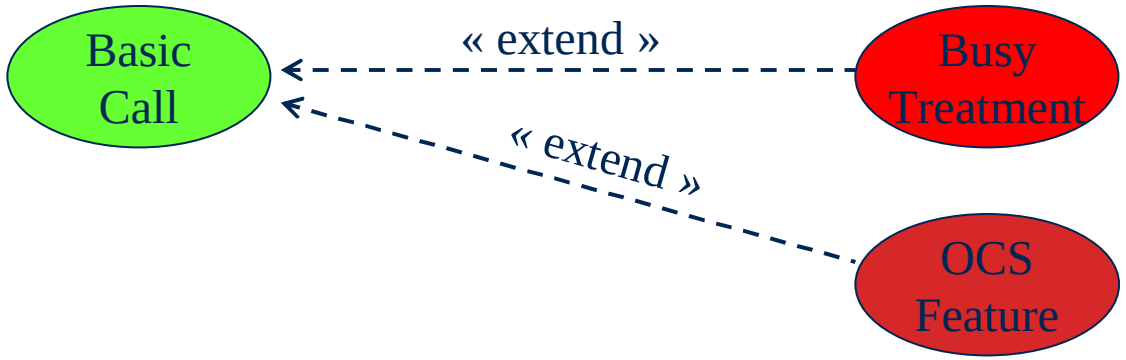


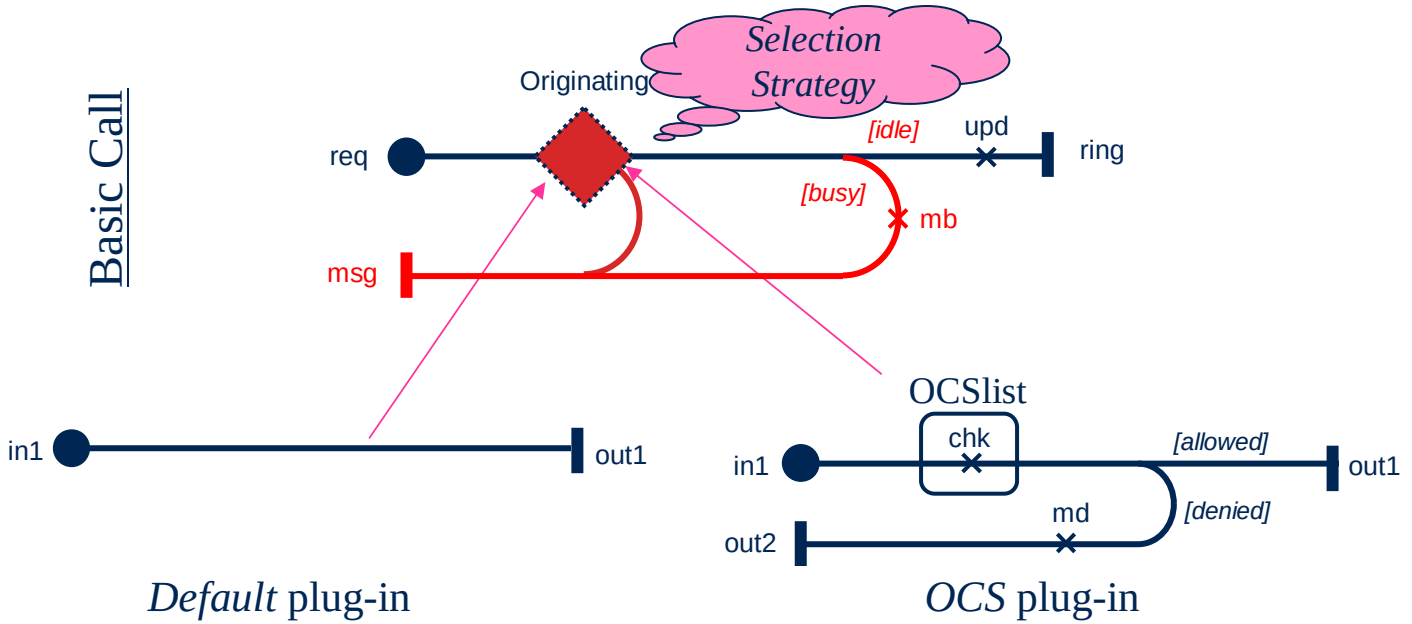
Extend Relationship

- Shows that part of a use case is:
  - (potentially) optional
  - Executed only under certain conditions
  - Inserted at an extension point in a base use case
  - Not required for the completion of base use case

- *Solution*:
- Use (guarded) OR-forks or dynamic stubs on a base use case
  - Extension points are visual
  - For dynamic stubs, there is a default plug-in that represents the original base case

Extend Relationship – Example



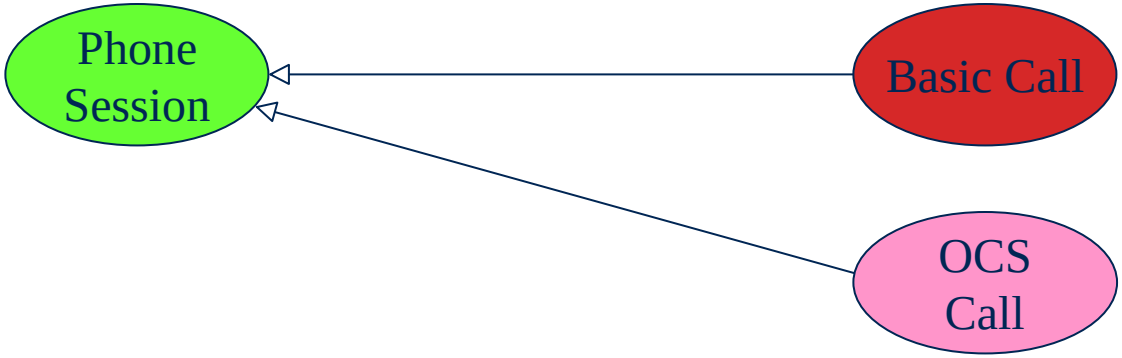


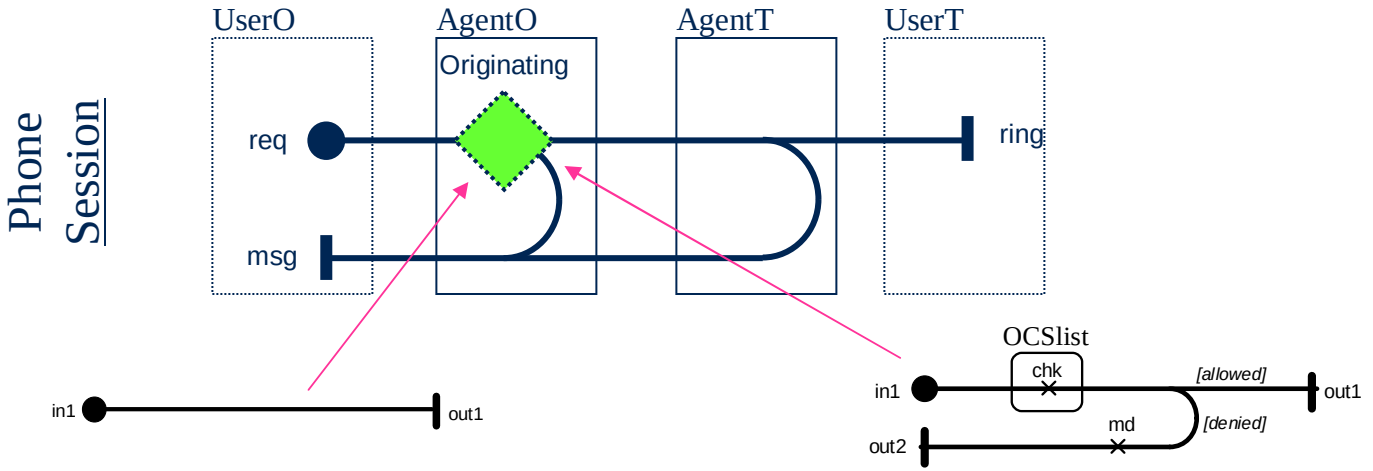
Generalization Relationship

- Used when two or more use cases have commonalties in behavior, structure, and purpose
- The shared part can then be described in a new parent use case specialized by child use cases

- *Solution*:
- Use OR-joins and OR-forks, or multiple dynamic stubs
  - Parent use case contains dynamic stubs for diverging behavior
  - Child use case is parent + plug-ins

Generalization Relationship – Example





Basic Call = Phone Session + Originating plug-in

OCS Call = Phone Session + OCS plug-in

Use Cases and GRL?

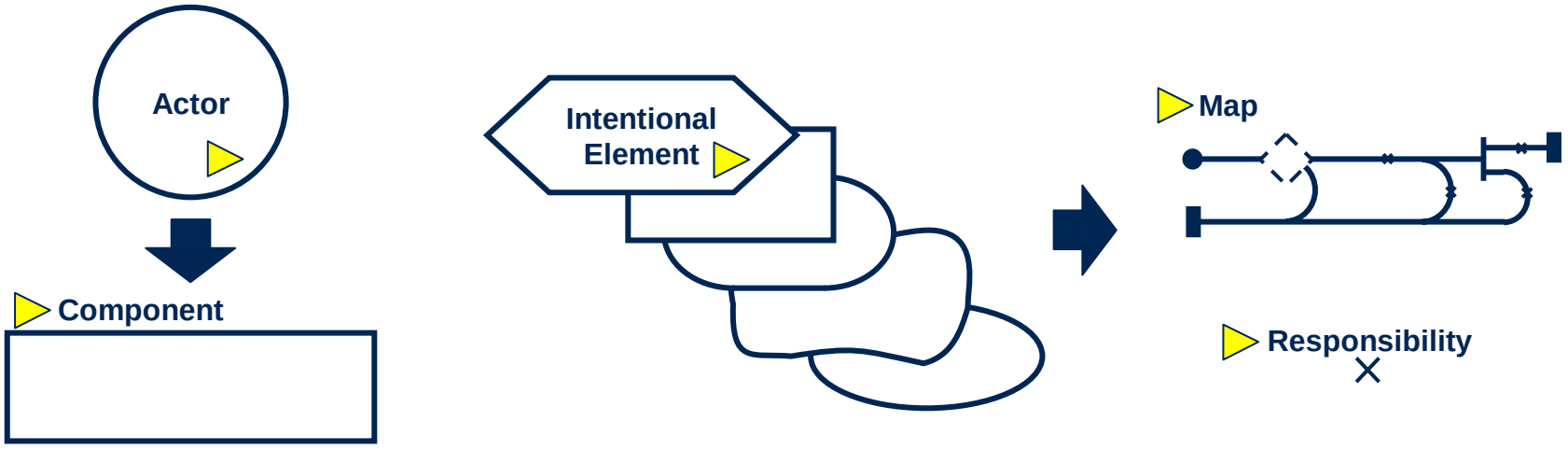
- Use cases are often weak at capturing non-functional aspects
- Misuse cases focus on some NF aspects, especially security and safety
  - The threatens, mitigates, aggravates relations could be mapped to GRL contributions
  - Mis-use cases are more specific and specialized than GRL for this domain
  - Support trade-off analysis and rationale documentation, like GRL
- They also allow to find new mitigation scenarios
  - They often become functions of sub-systems

Integrating GRL and UCM

- Traceability between:
  - Goals/tasks and UCMs (or UCM scenario definitions)
  - Tasks and UCM responsibilities (different granularity)
  - Requirements management
  - Others…
- Enables completeness and consistency analysis
- Underspecification and overspecification
  - Discovery of new goals and scenarios, removal of unnecessary goals and scenarios
  - Examples:
    - Why is there a UCM scenario without any link to a GRL goal?
    - Why is there a GRL goal without any link to a UCM scenario?
- Refinements of alternative solutions
  - From GRL (identification) to UCM (evaluation)

From GRL Models to UCM Models – URN Links

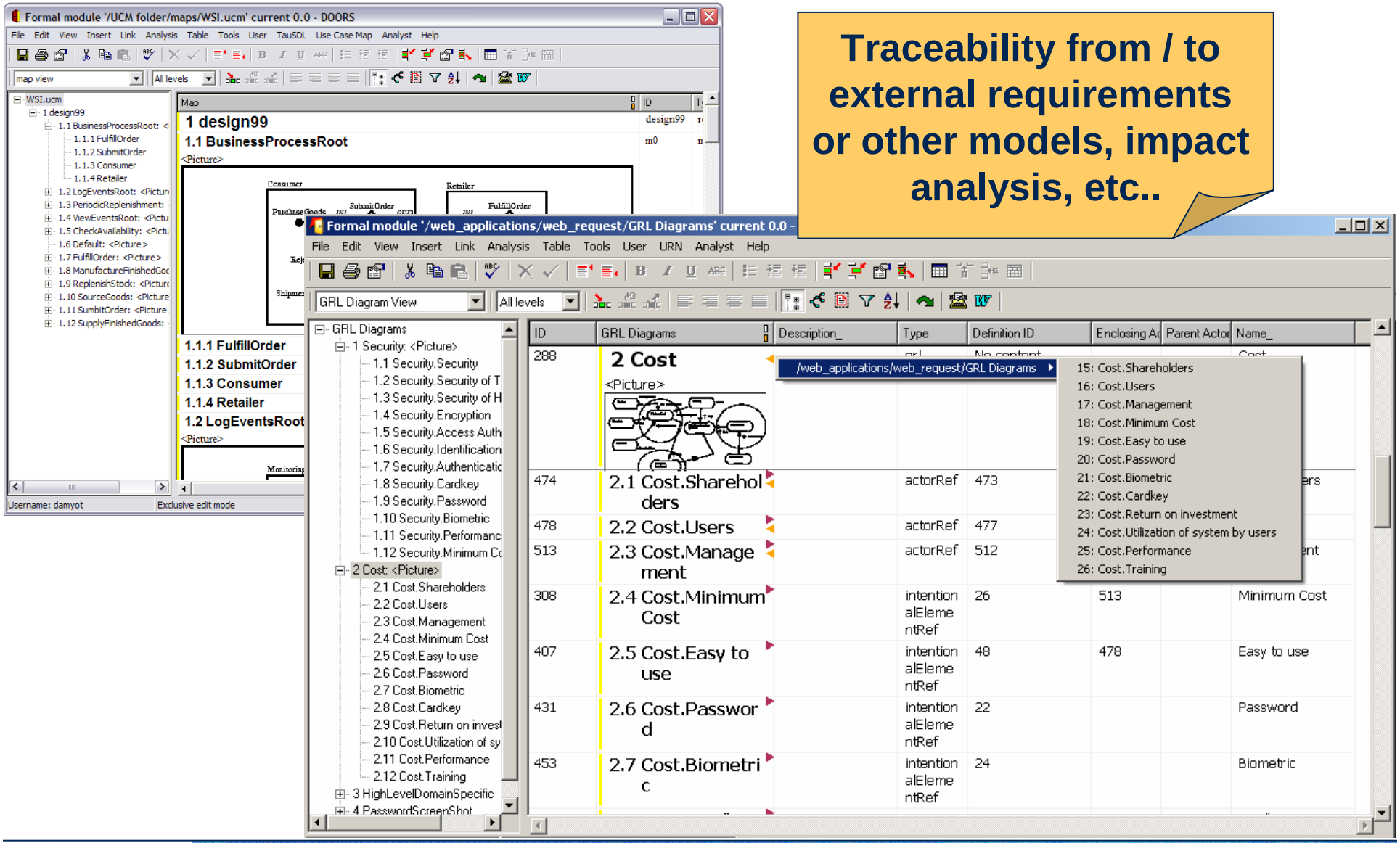
- URN (Typed) Links establish traceability relationships
  - Connect any pair of URN model elements
  - Most frequently, URN links are used to trace …
    - Actors in GRL models to components in UCM models
    - Tasks in GRL models to maps or responsibilities in UCM models



- Evaluation of the impact of strategies on the operational and architectural aspects, using URN links
- User-defined links for requirements management

URN link:

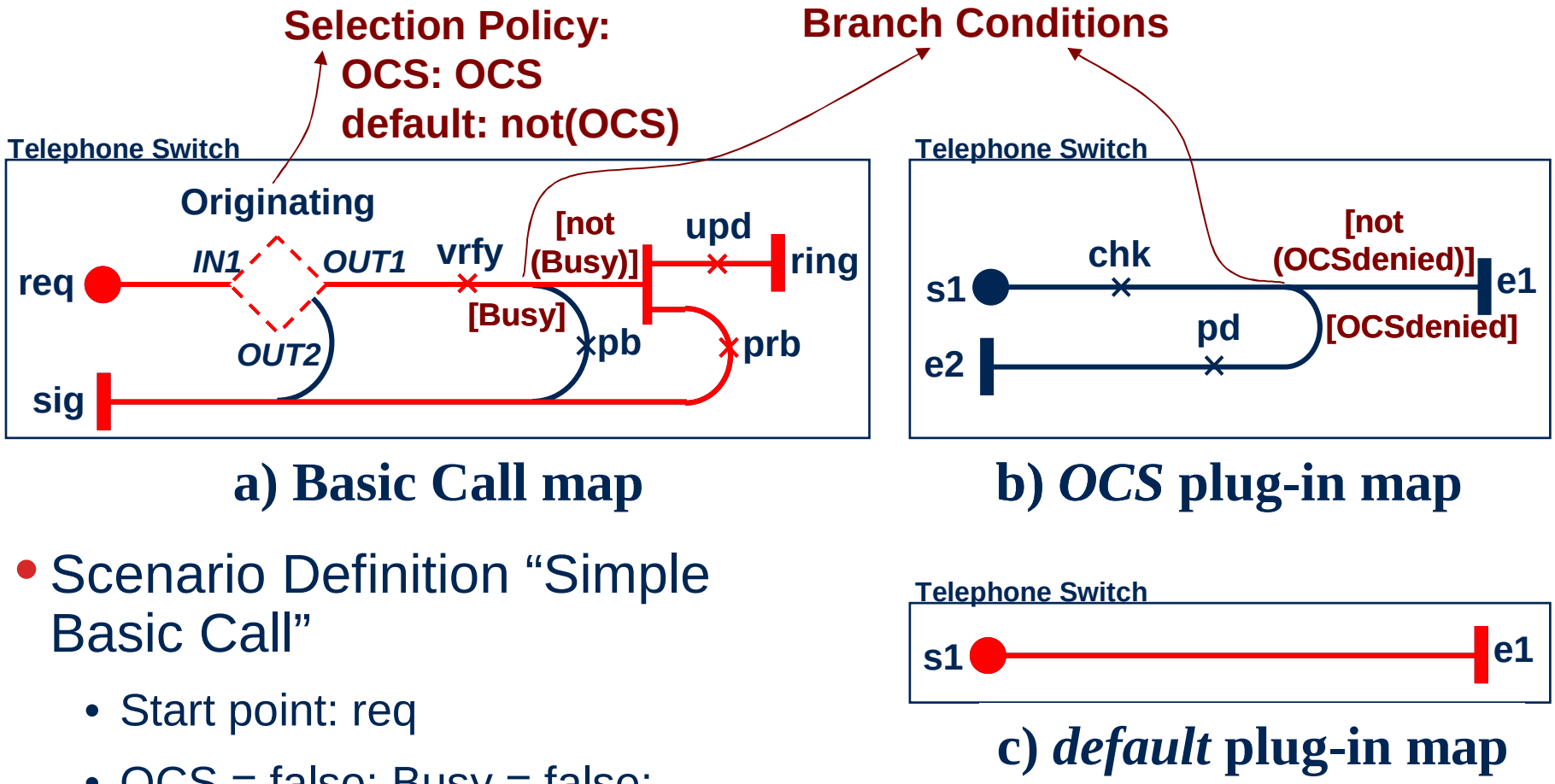
Requirements Management



Traceability from / to external requirements or other models, impact analysis, etc..

Use Case Maps – Scenario Execution (1)

UCM Example: Tiny Telephone System



a) Basic Call map

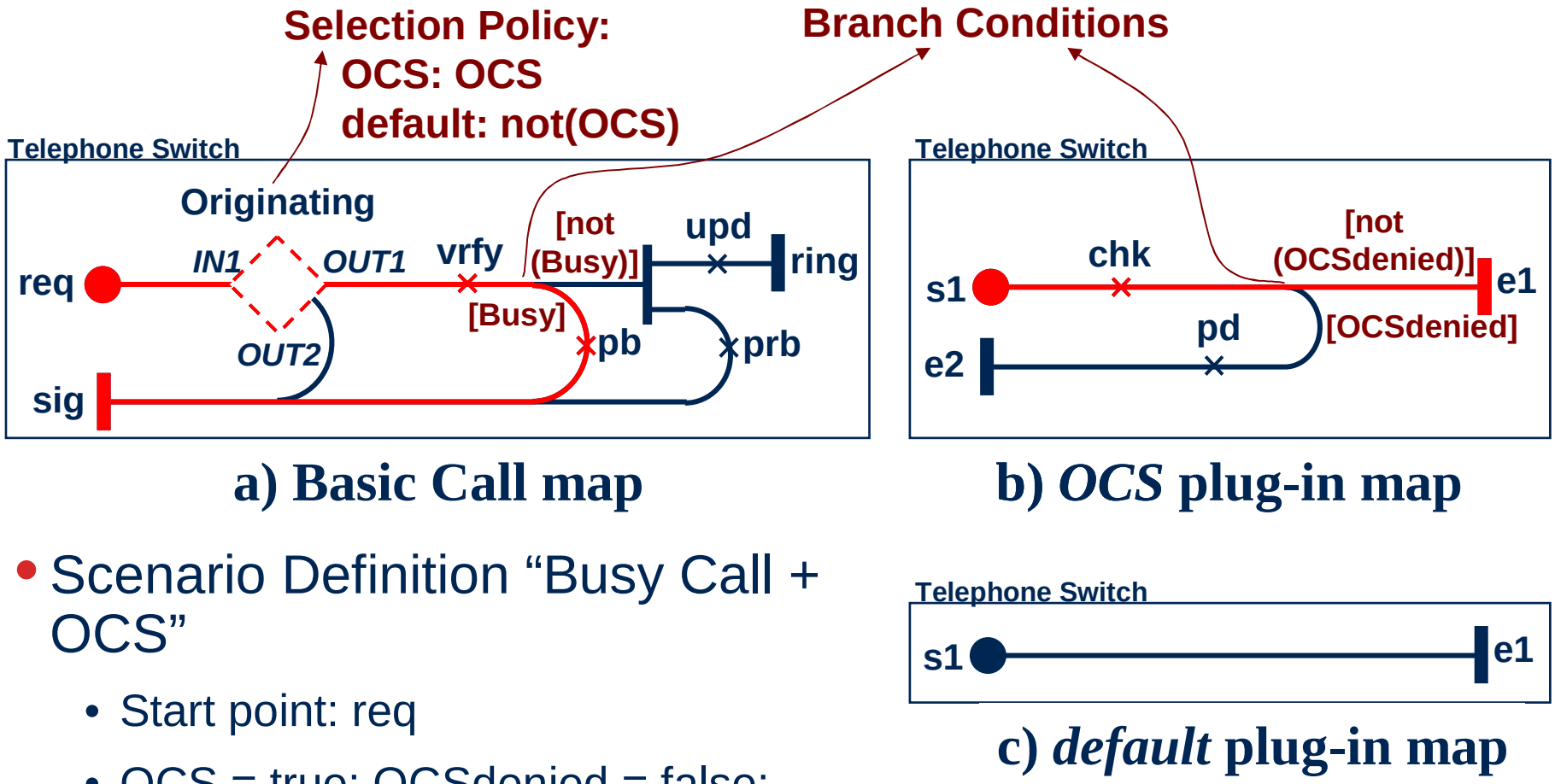
b) *OCS* plug-in map

c) *default* plug-in map

- Scenario Definition "Simple Basic Call"
  - Start point: req
  - OCS = false; Busy = false;
  - End points: ring, sig

Use Case Maps – Scenario Execution (2)

UCM Example: Tiny Telephone System



a) Basic Call map

b) *OCS* plug-in map

c) *default* plug-in map

- Scenario Definition "Busy Call + OCS"
  - Start point: req
  - OCS = true; OCSdenied = false; Busy = true;
  - End point: sig

Use Case Maps – Traversal Mechanism (1)

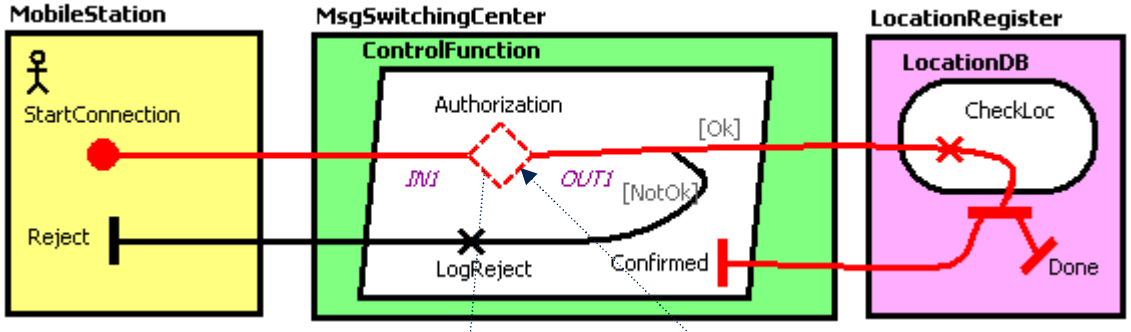
- UCM scenarios describe one path through the UCM model (only one alternative at any choice point is taken)
  - Set of initial values for the variables used in conditions and responsibilities
  - Start points triggered, end points reached
  - Possibly pre/post conditions
- jUCMNav's traversal mechanism executes the UCM model given UCM scenario description(s) (i.e. highlights the scenario(s))
  - Intuitive interpretation aligned with UCM semantics except for dynamic stubs which are deemed to contain an XOR for the selection of a single plug-in map
- Extraction of individual scenarios

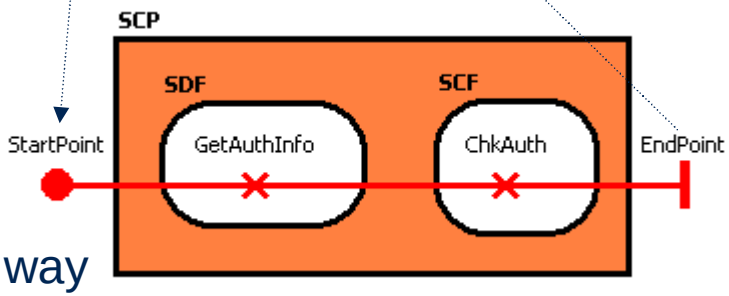
Use Case Maps – Traversal Mechanism (2)

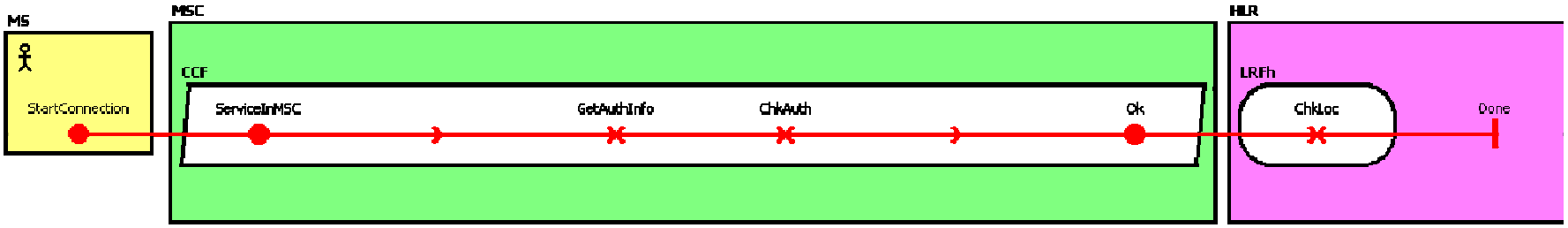
- Two options
  - Deterministic (only one alternative at any choice point can be enabled)
  - Non-deterministic (randomly choose an alternative from all enabled ones)
- Boolean, Integer, and Enumeration variables are evaluated and can be changed by responsibilities during the traversal of the UCM model
  - Variables are used in expressions for any alternative of a choice point
  - Conditions attached to selection points
- Groups of scenarios can be run together
  - Useful for regression testing

Scenario Export – UCM Model

- Scenarios can be exported to:
  - UCM model where all scenarios are linearized
    - Stubs flattened and choices resolved (but documented with special waiting places)
  - UCM model where all scenarios are linearized and well-formed
    - From graph to "tree" (especially for AND-joins)
    - Some concurrency may be lost along the way

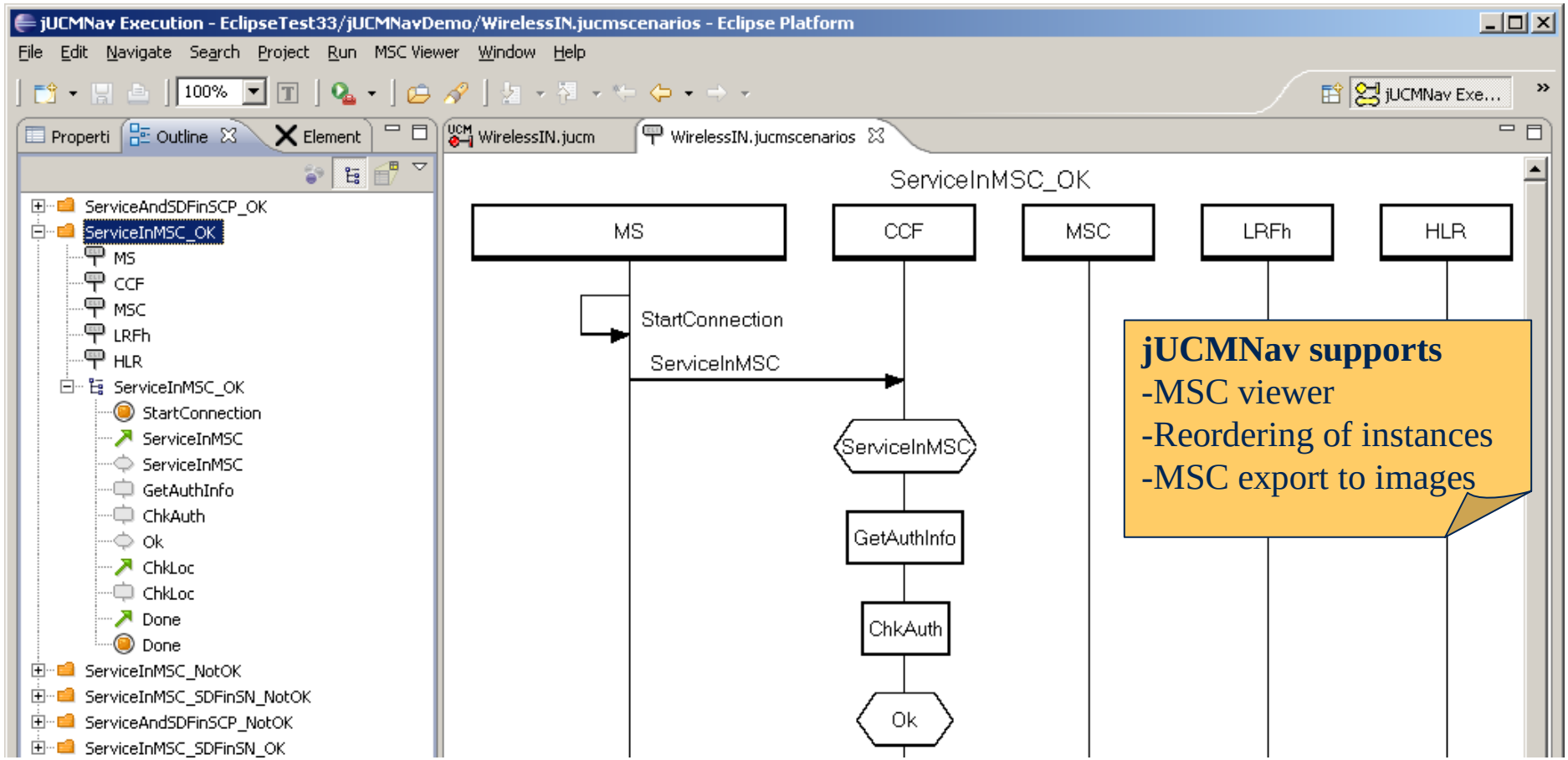






Scenario Export – MSC

- Scenarios can be exported to:
  - MSC model with one diagram per scenario
  - Can be visualized with embedded MSC viewer



Key Points - Scenario Definitions

- Improves understanding of (lengthy) scenarios
- Validation and regression testing

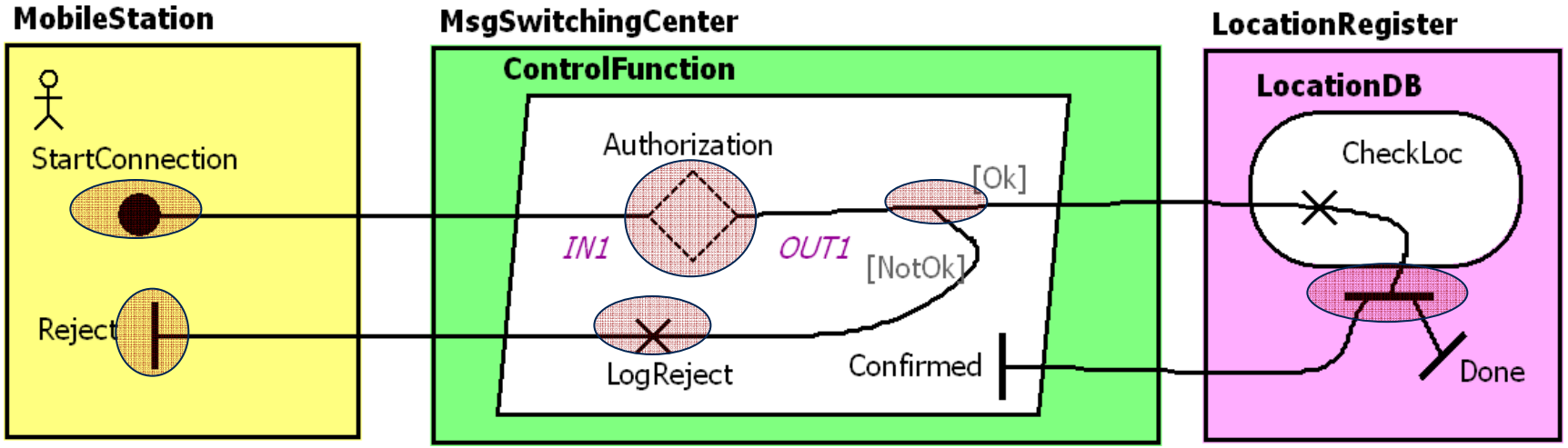
- Path data model is not a problem domain data model

- Scenario definitions are the foundation for more advanced functionality based on UCM path traversal mechanisms (highlight, transformations)
- Much value in a tool-supported translation

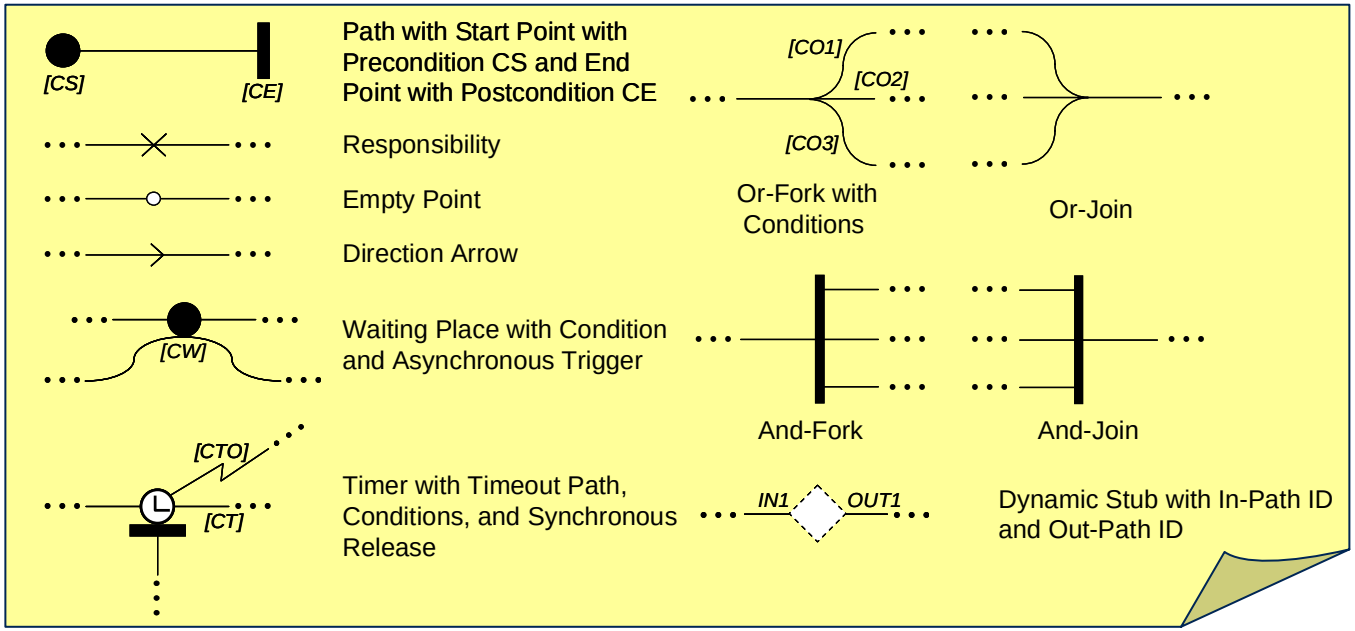
UCM Example I – Context

- New service for wireless network
  - Where to put the service logic?
  - Where to put the service data?

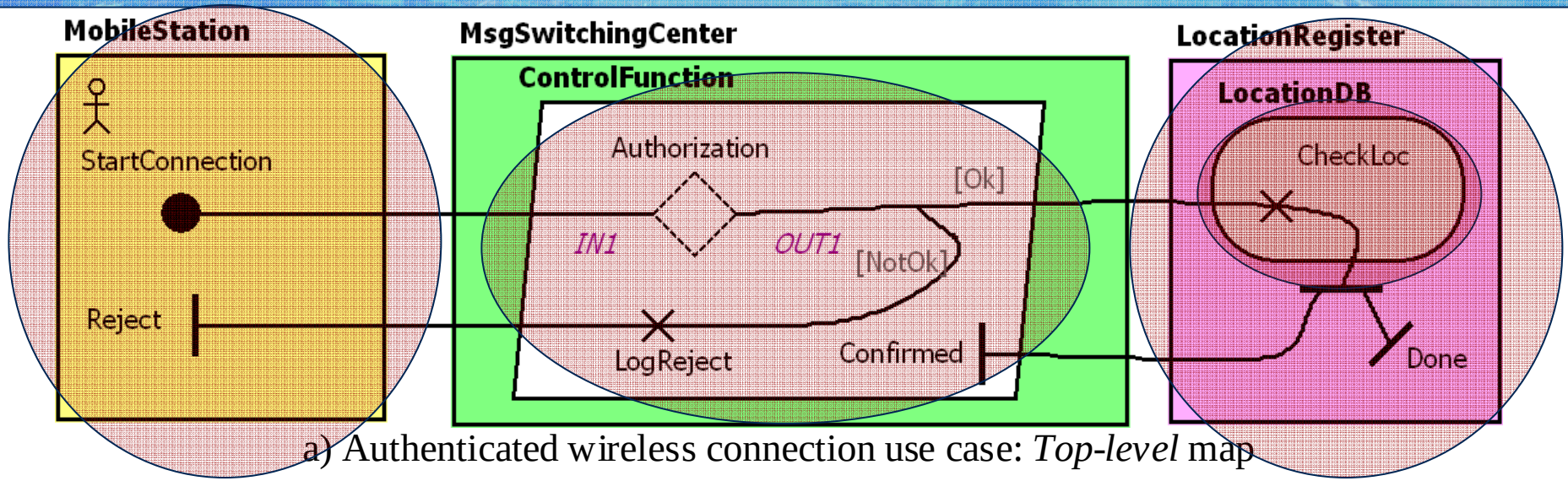
UCM Example I – Path Nodes



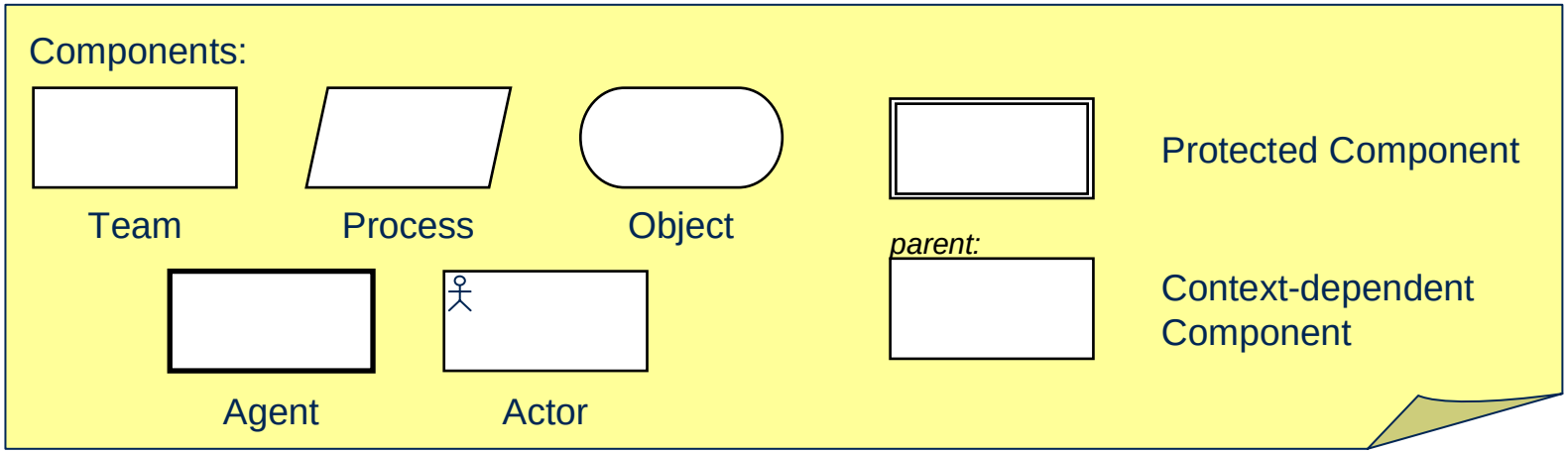
a) Authenticated wireless connection use case: *Top-level* map



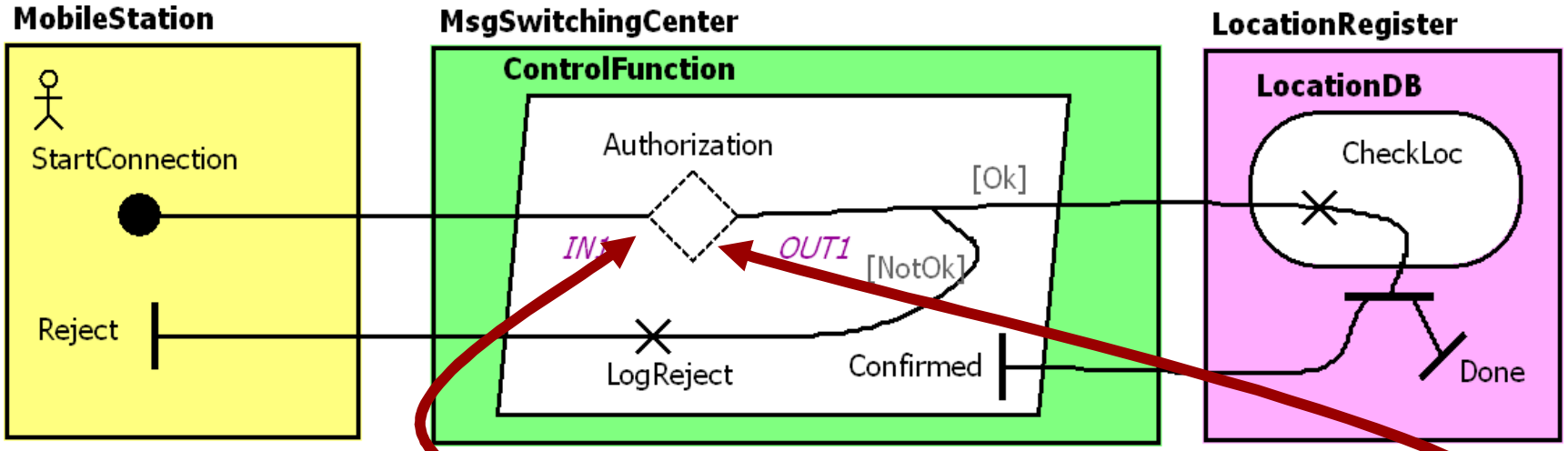
UCM Example I – Components



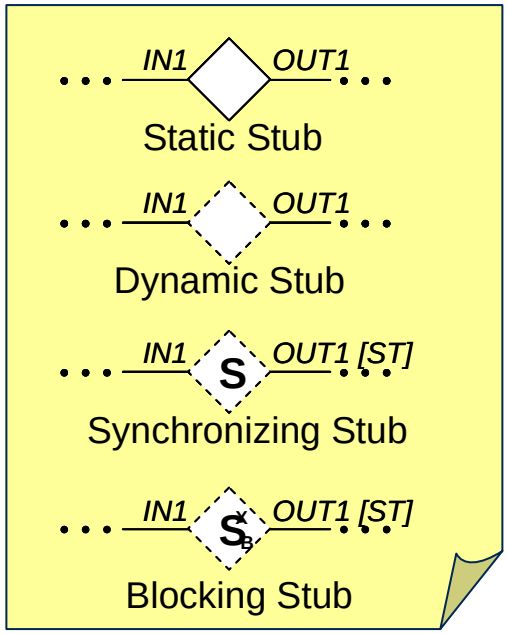
a) Authenticated wireless connection use case: *Top-level* map

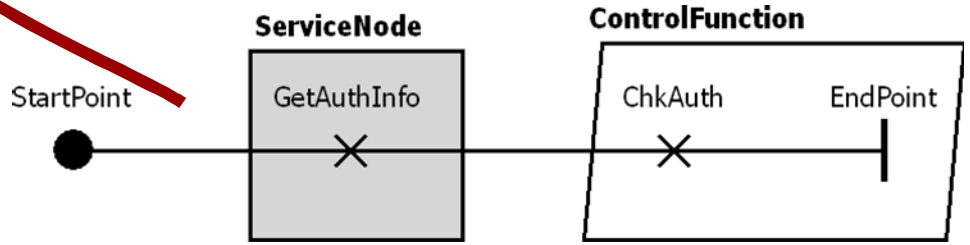


UCM Example I – Stubs and Plug-ins

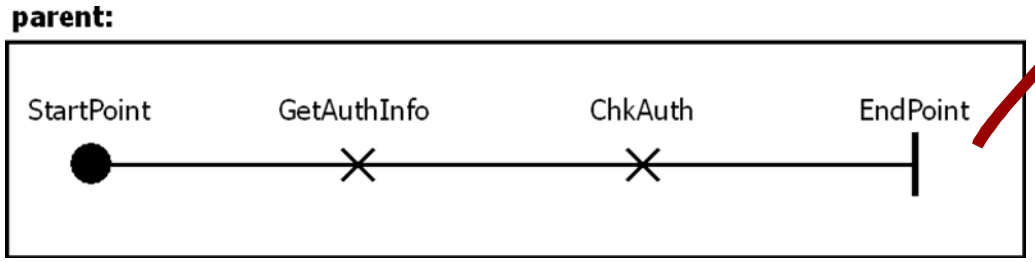


a) Authenticated wireless connection use case: *Top-level* map



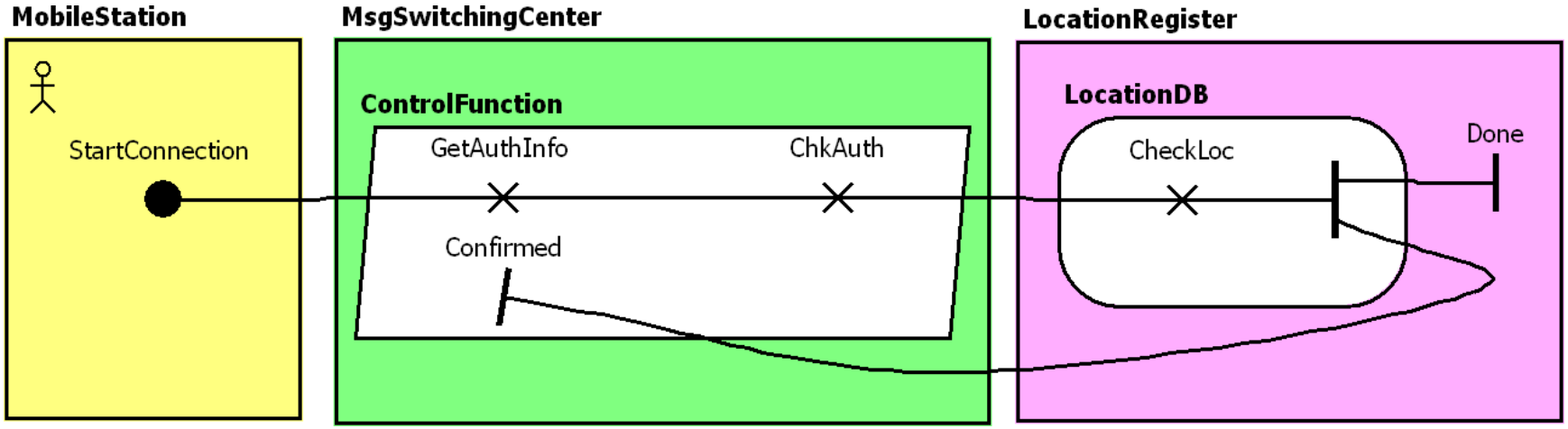


b) Service in MSC, data in external service node: *SvcInMSC_DataInSN* map

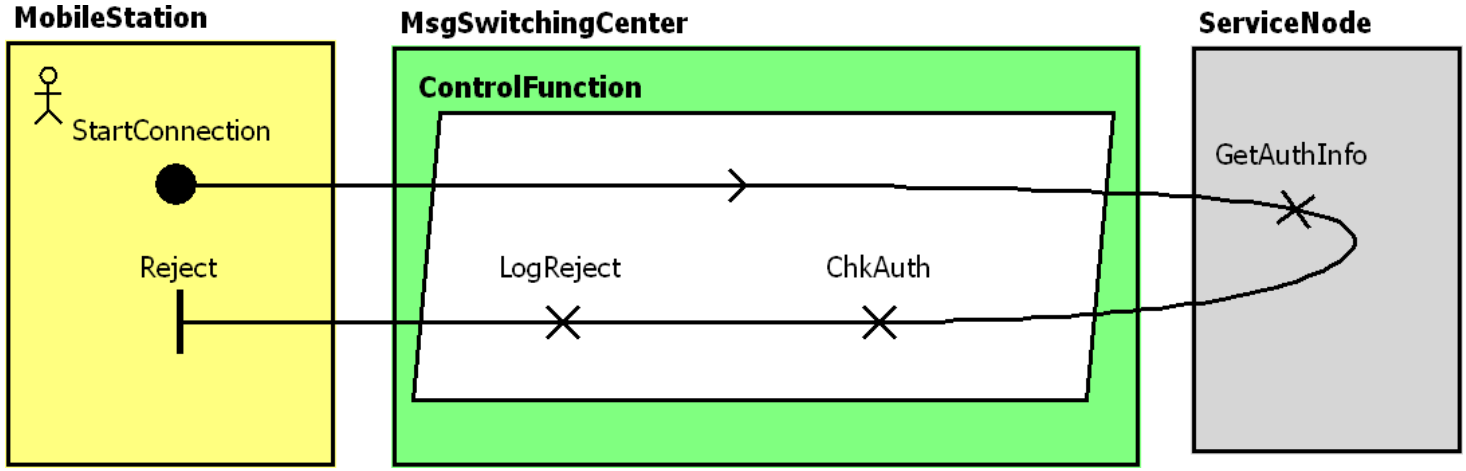


c) Service and data in MSC: *SvcInMSC* map

UCM Example I – Scenarios Exported to UCM Model

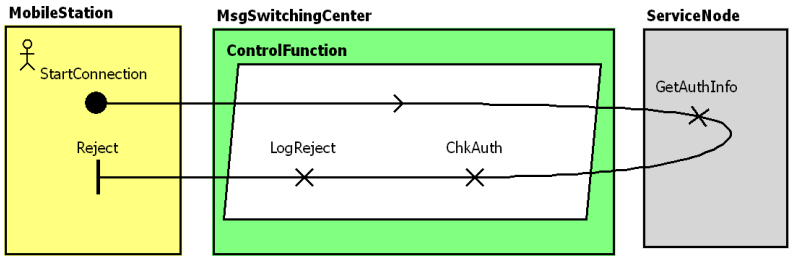


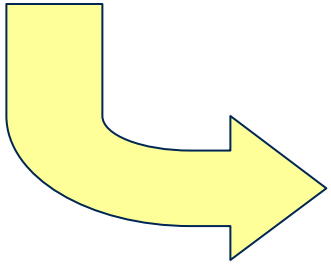
ServiceInMSC_OK: StartConnection, authorization variable is *true* ([Ok]), SvcInMSC plug-in selected

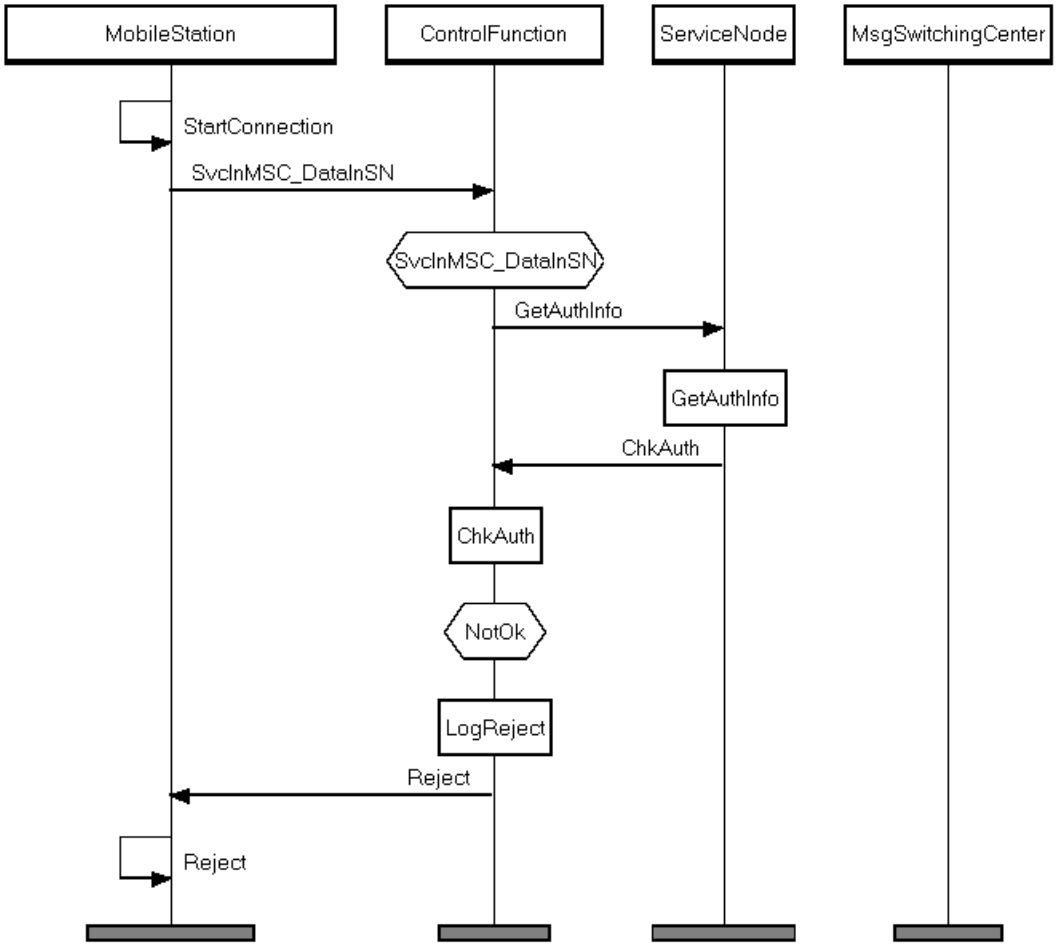


ServiceInMSC_DataInSN_NotOK: StartConnection, authorization variable is *false* ([NotOk]), SvcInMSC_DataInSN plug-in selected

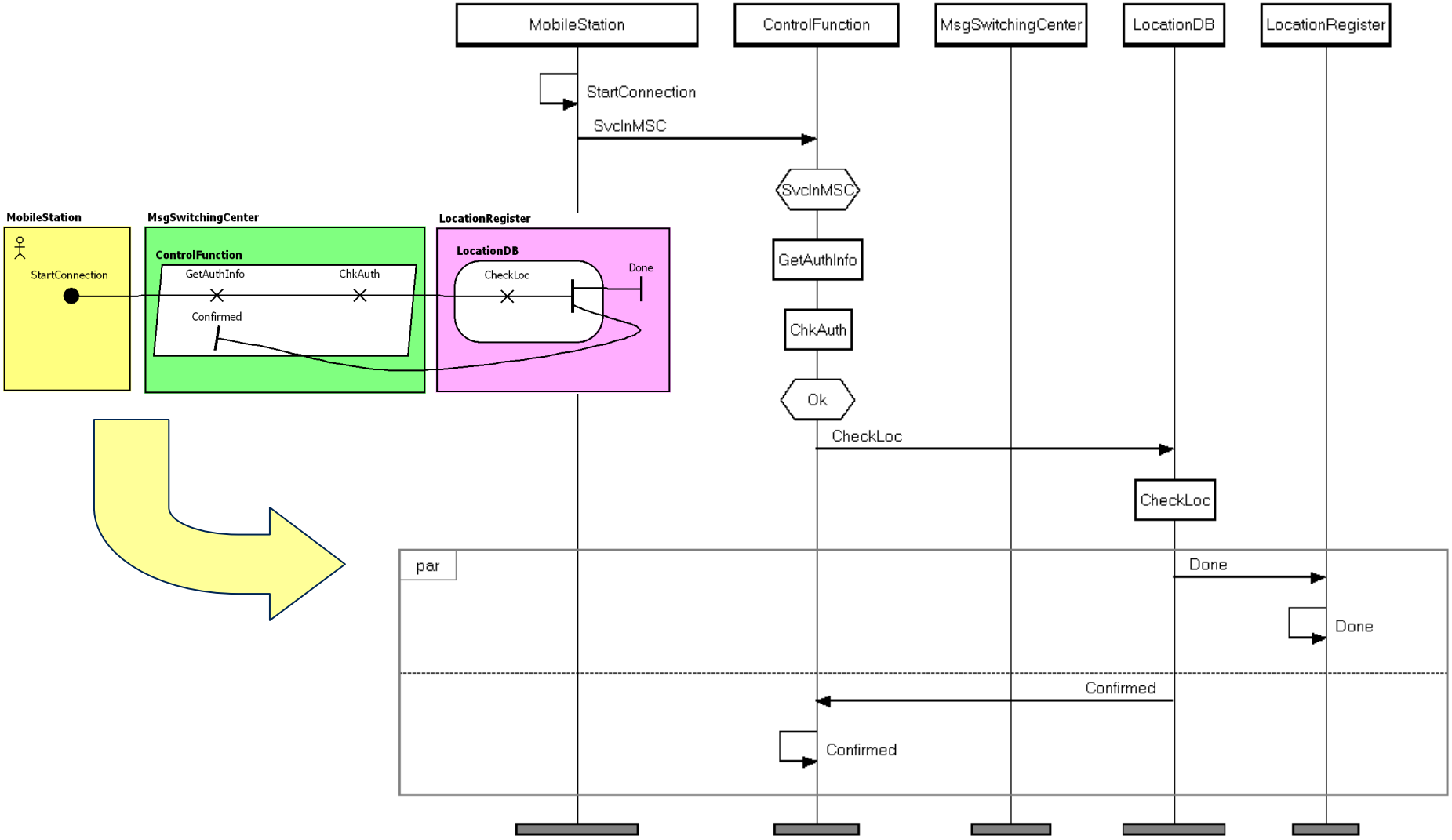
UCM Example I – Scenario Refinement with MSCs (1)







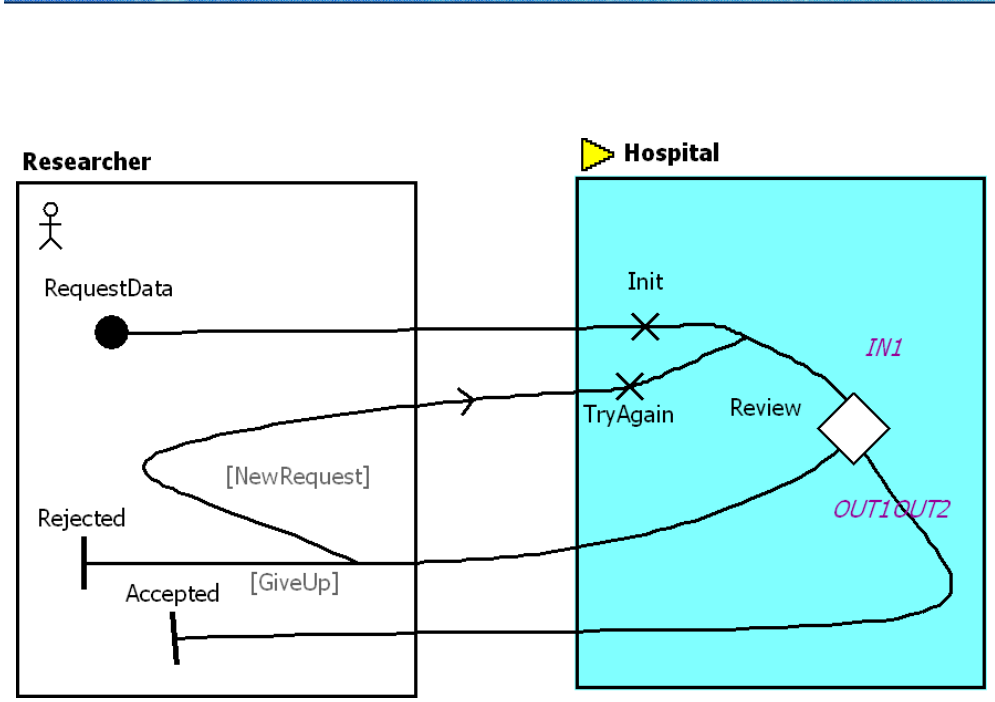
UCM Example I – Scenario Refinement with MSCs (2)

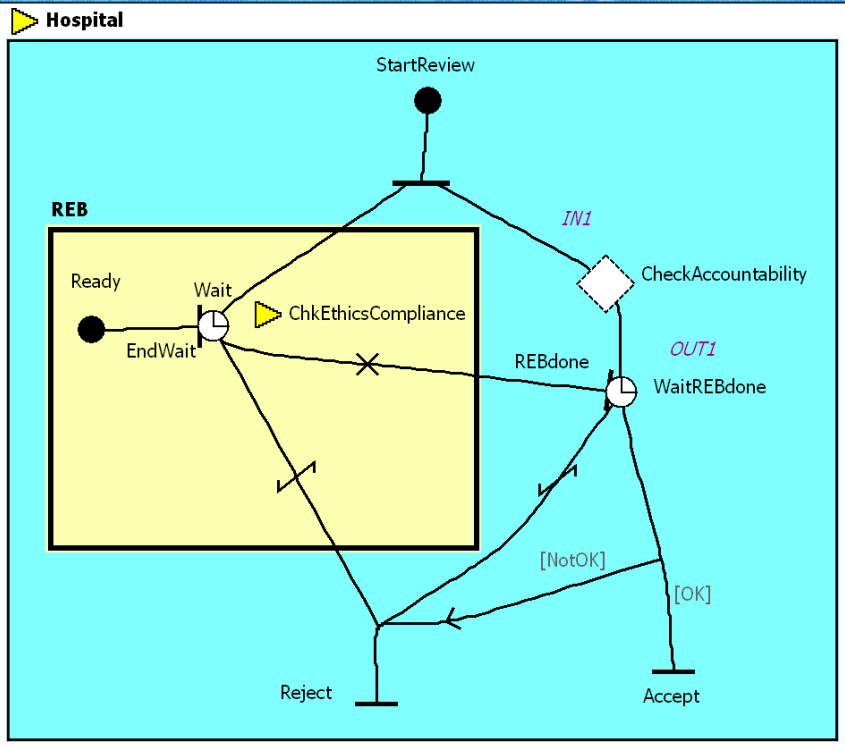


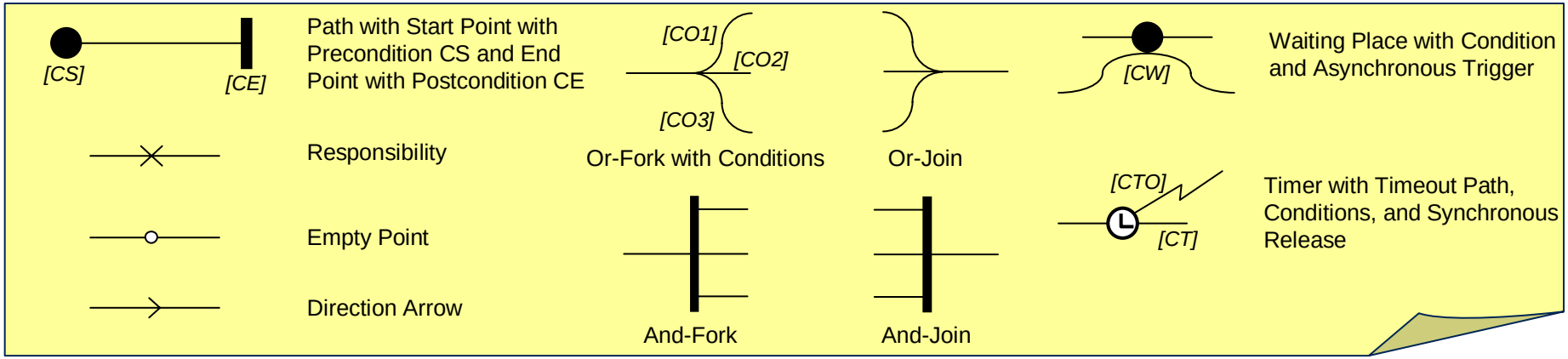
UCM Example II – Context

- GRL model that addresses privacy protection in a hospital environment
  - Researchers want access to patient data but the Health Information Custodian (HIC – i.e., the hospital) needs to protect patient privacy, as required by law (PHIPA in Ontario).
  - The process of accessing databases must ensure privacy. As required by law, a Research Ethics Board (REB) is usually involved in assessing privacy risks for the research protocol proposed by a researcher.
  - DB administrators also want to ensure that DB users are accountable for their acts.

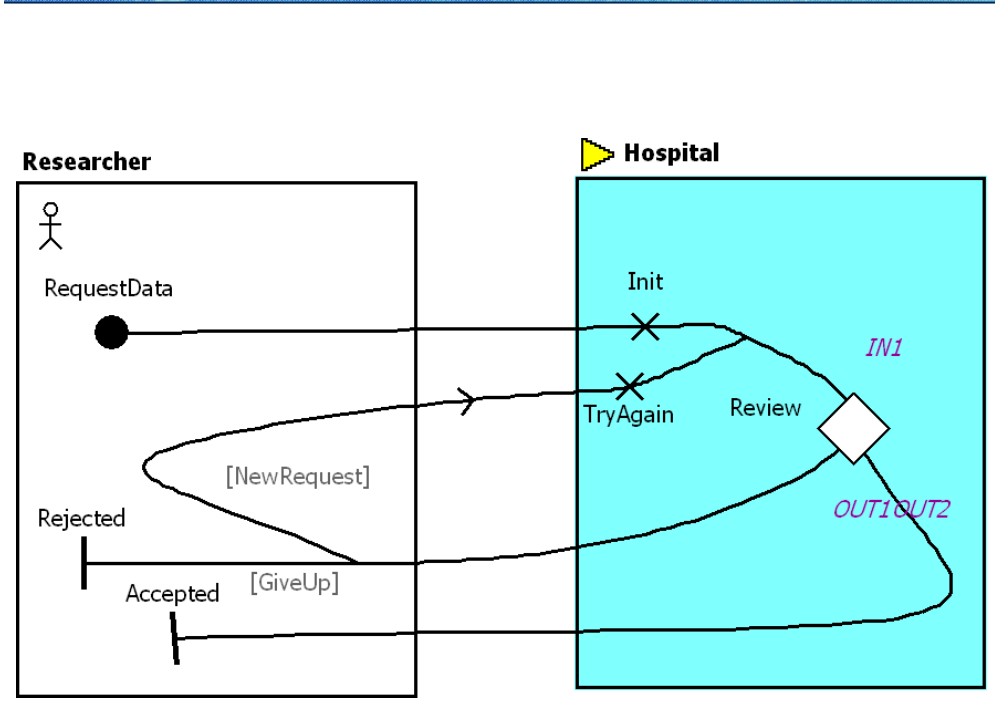
UCM Example II –Path Nodes

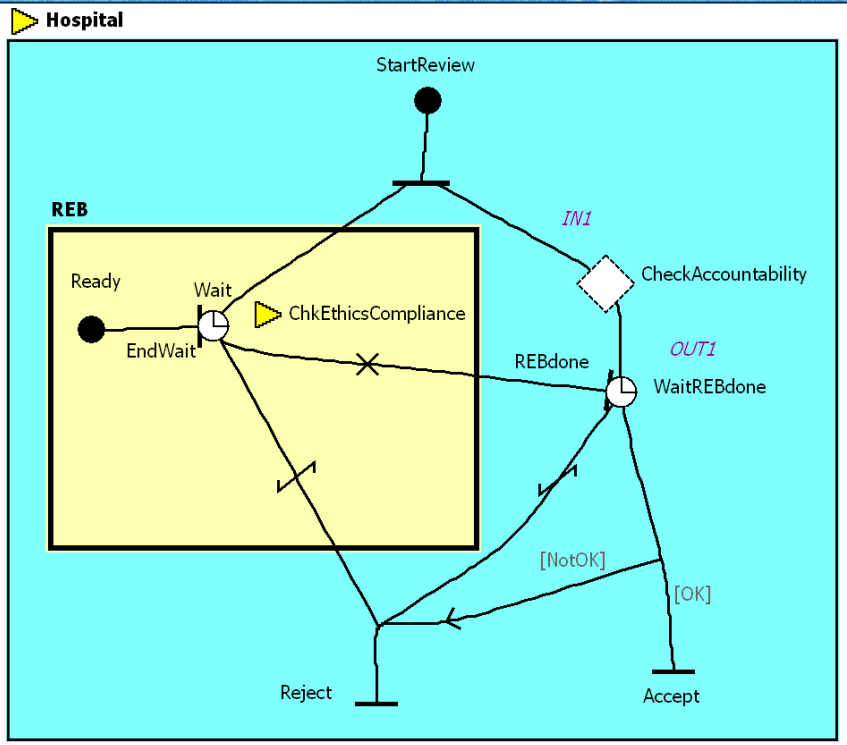






UCM Example II – Components





Components:

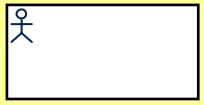

Team


Process

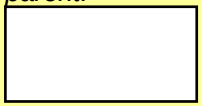

Object


Protected Component

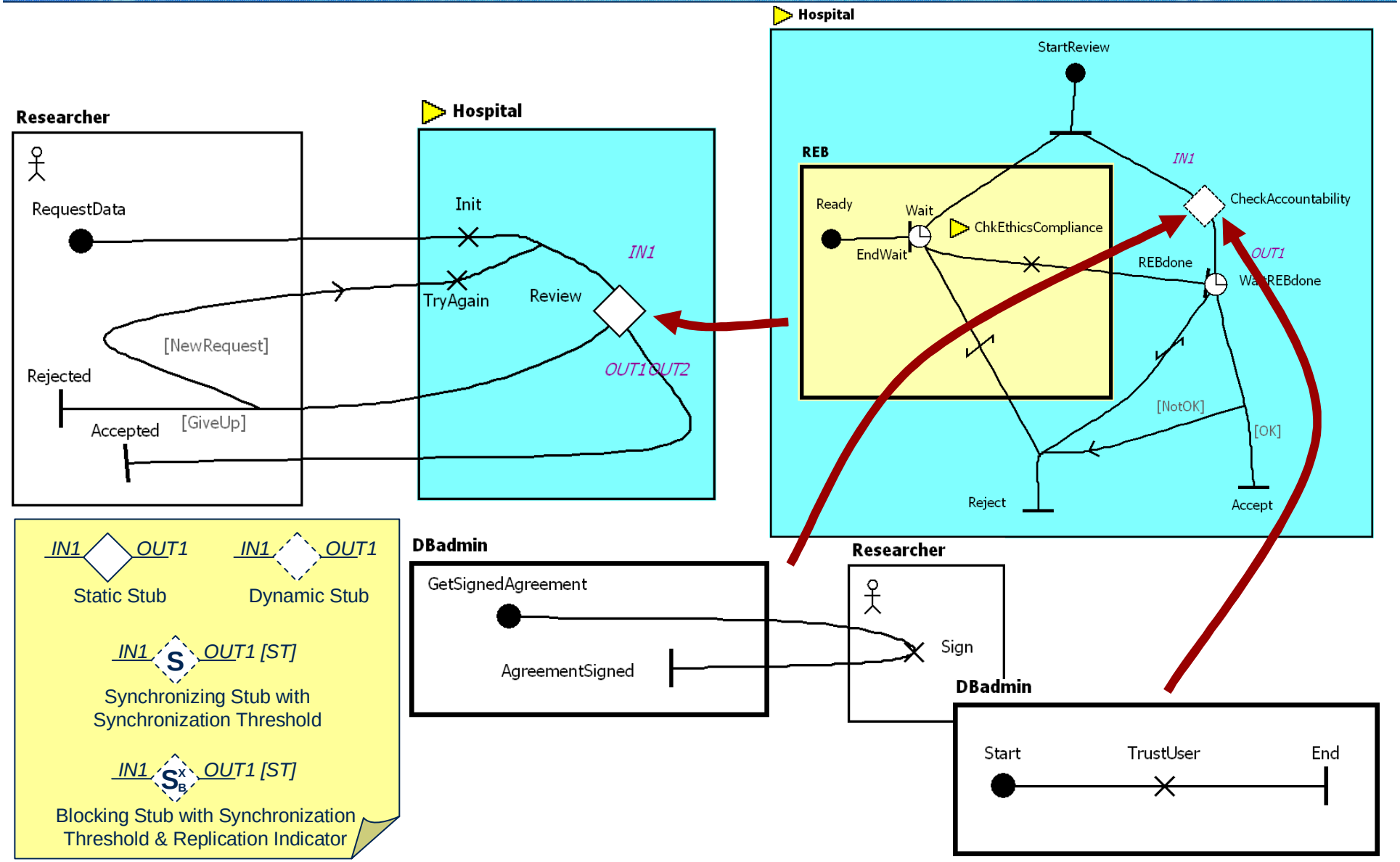

Agent


Actor

parent:

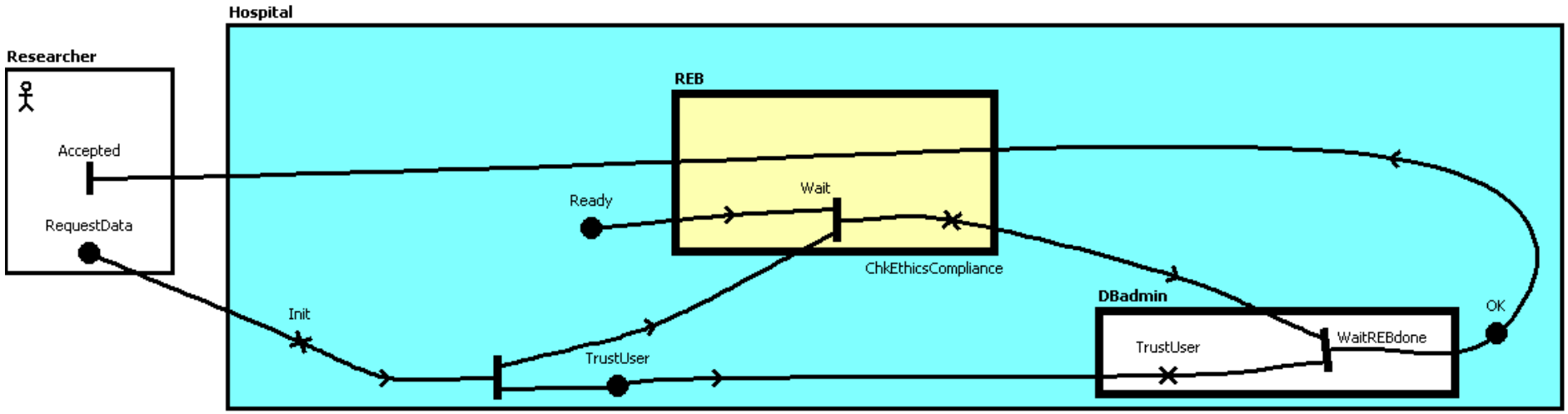

Context-dependent Component

UCM Example II – Stubs and Plug-ins

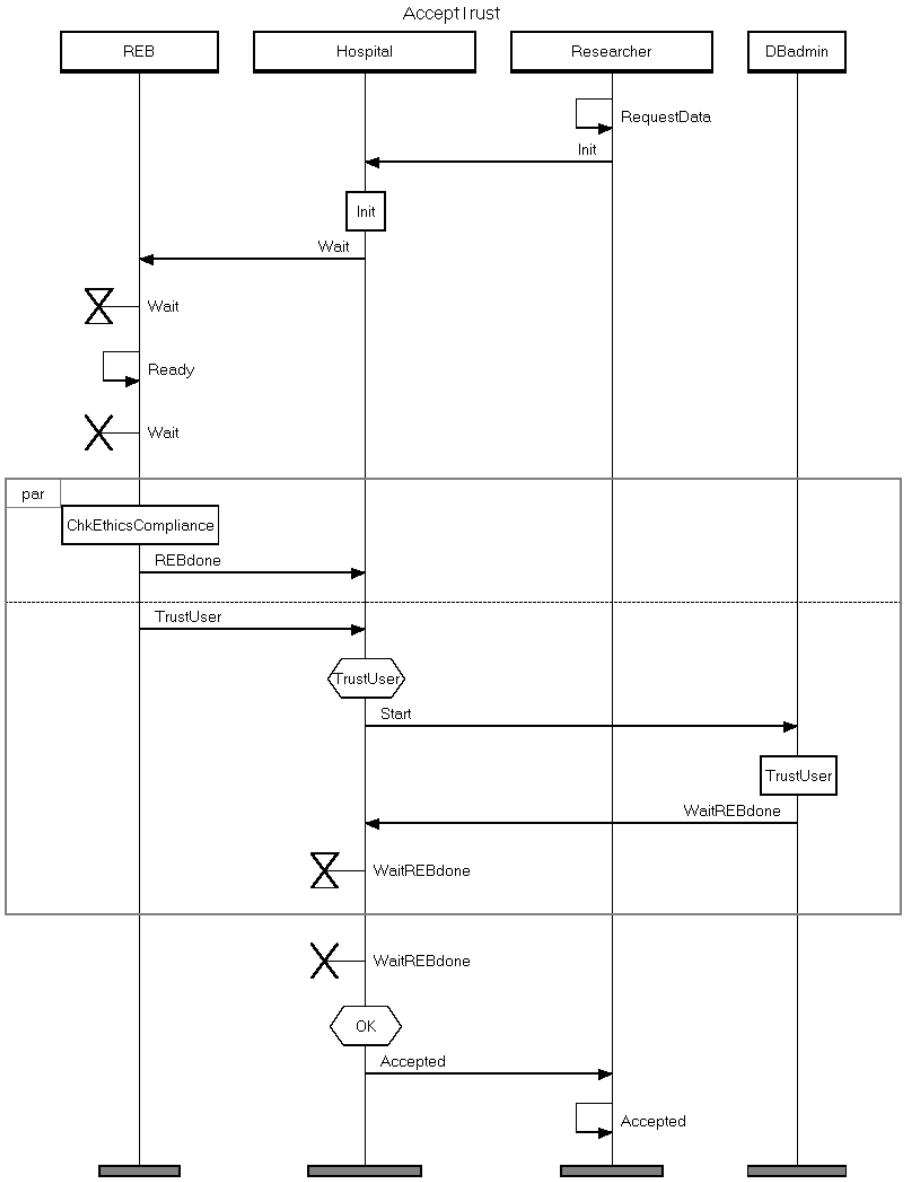


UCM Example II – Visualization of Scenario as UCM

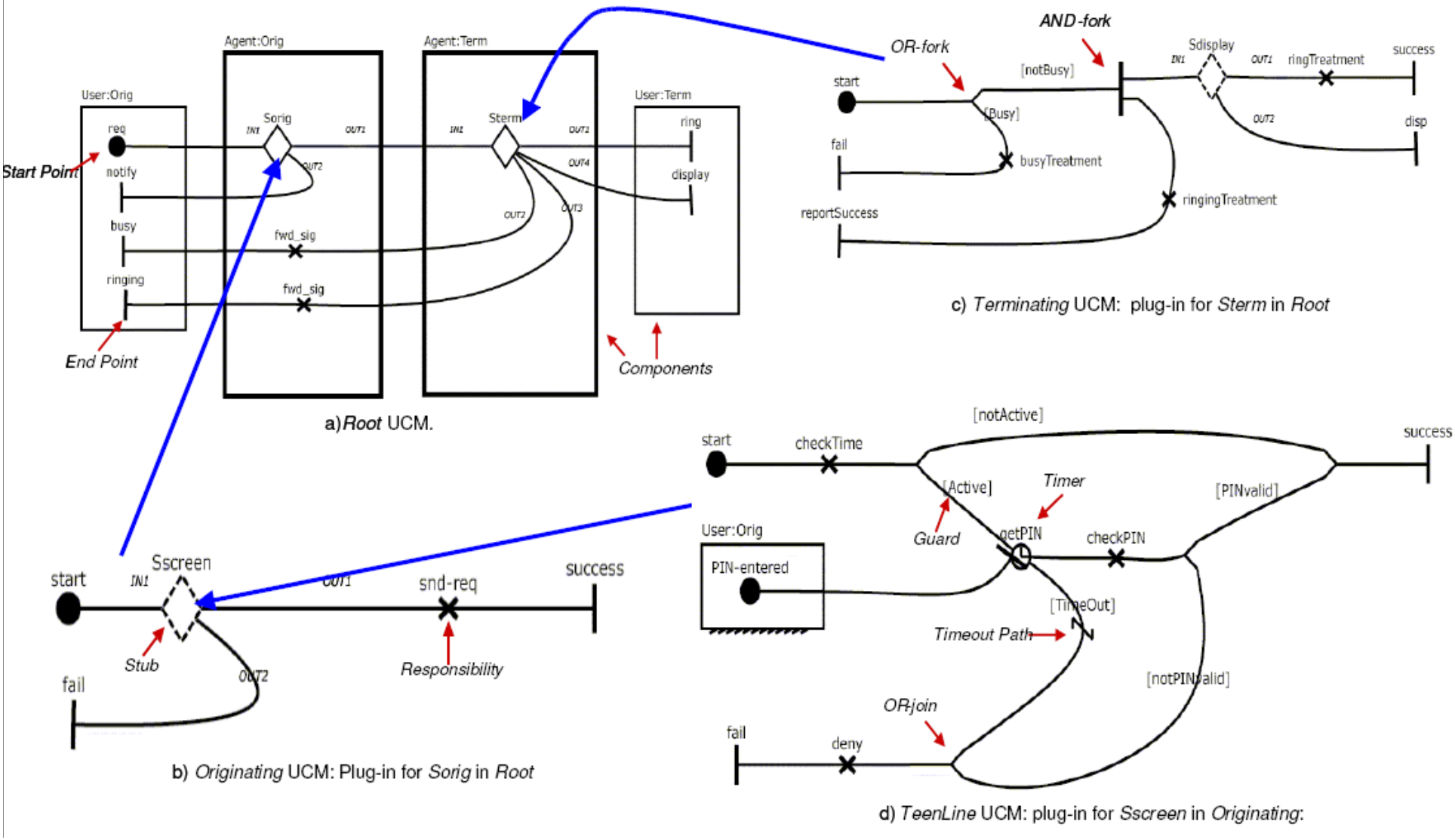
- Start Points
  - RequestData
  - Ready
- Variables
  - CheckAccountability = TrustUser; ReviewResult = OK,
- Traversed scenario can be visualized as a UCM



UCM Example II – Visualization of Scenario as MSC



UCM Example III – Telephony Features

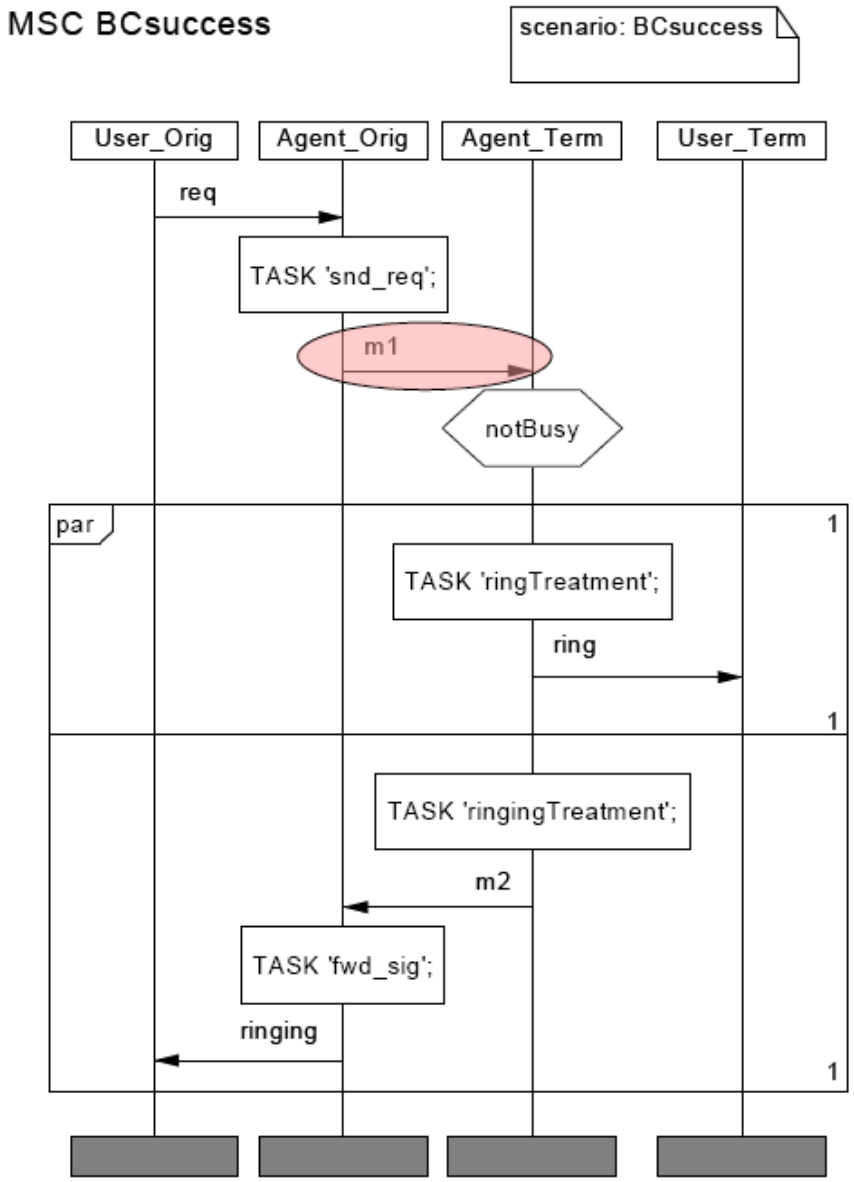


UCM Example III – Scenario Definitions

Number	Scenario Name	Variables								Start Points	
		Busy	*OnOCSList*	*PINvalid*	*TLactive*	*getPIN_ timeout*	*subCND*	*subOCS*	*subTL*	*req*	*PIN-entered*
01	BCbusy	T	-	-	-	-	F	F	F	X	
02	BCsuccess	F	-	-	-	-	F	F	F	X	
03	OCSbusy	T	F	-	-	-	F	T	F	X	
04	OCSdenied	F	T	-	-	-	F	T	F	X	
05	OCSsuccess	F	F	-	-	-	F	T	F	X	
06	CNDdisplay	F	-	-	-	-	T	F	F	X	
07	OCS_CNDdisplay	F	F	-	-	-	T	T	F	X	
08	TL_CNDActiveBusy	T	-	T	T	F	T	F	T	X	X
09	TL_CNDActiveDisplay	F	-	T	T	F	T	F	T	X	X
10	TL_CNDnotActiveBusy	T	-	-	F	-	T	F	T	X	
11	TL_CNDPINInvalid	-	-	F	T	F	T	F	T	X	X
12	TL_CNDTimeOut	-	-	-	T	T	T	F	T	X	
13	TL_CNDnotActiveDisplay	F	-	-	F	-	T	F	T	X	
14	TLnotActiveSuccess	F	-	-	F	-	F	F	T	X	
15	TLActiveSuccess	F	-	T	T	F	F	F	T	X	X

UCM Example III – Sample MSC

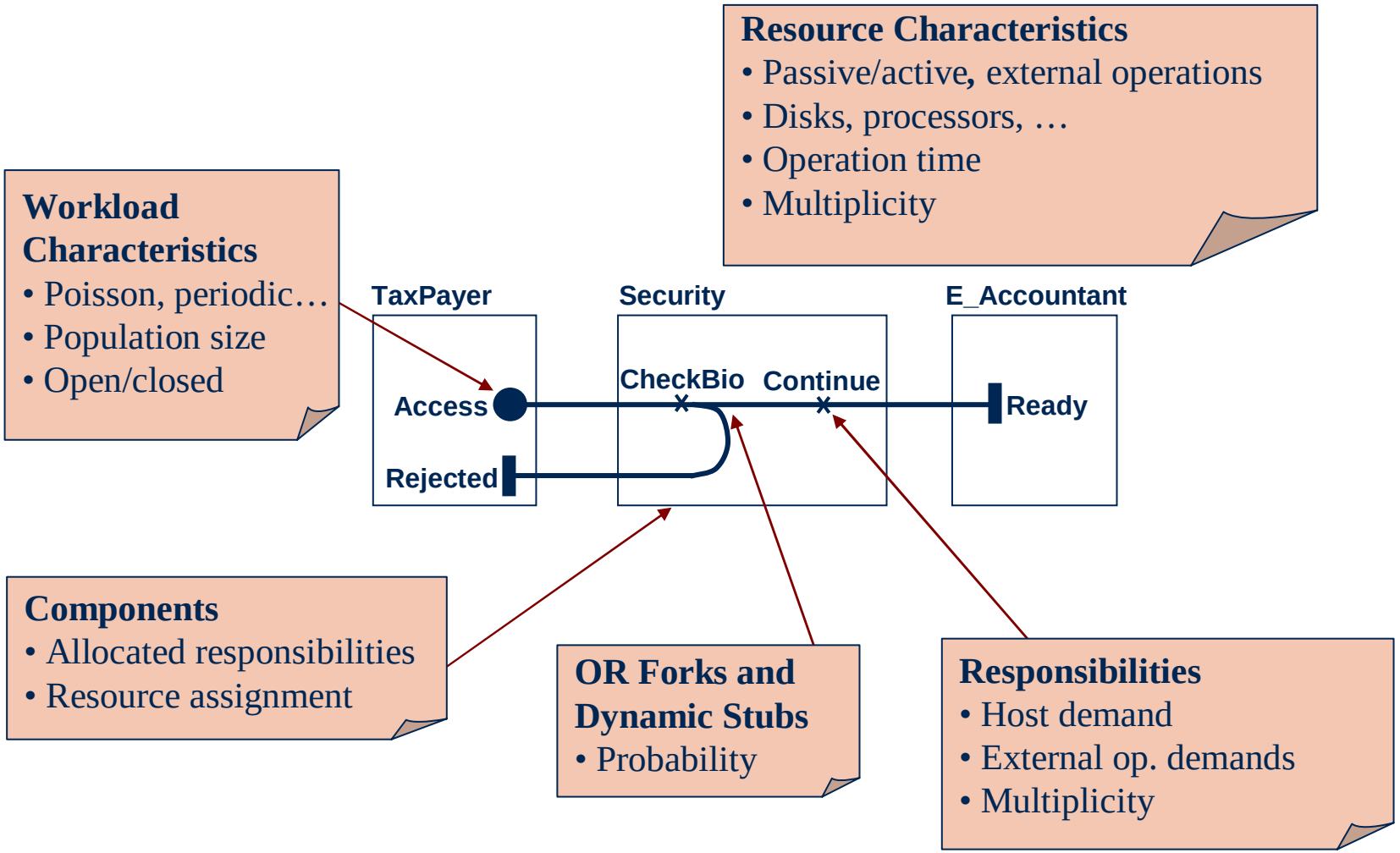
```
mscdocument BCsuccess;
msc BCsuccess;
User[Orig]: instance;
Agent[Orig]: instance;
Agent[Term]: instance;
User[Term]: instance;
text 'scenario: BCsuccess';
User[Orig]: out req,1 to Agent[Orig];
Agent[Orig]: in req,1 from User[Orig];
        action 'snd_req';
        out m1,2 to Agent[Term];
Agent[Term]: in m1,2 from Agent[Orig];
        condition [notBusy];
all: par begin;
Agent[Term]: action 'ringTreatment';
        out ring,3 to User[Term];
User[Term]: in ring,3 from Agent[Term];
par;
Agent[Term]: action 'ringingTreatment';
        out m2,4 to Agent[Orig];
Agent[Orig]: in m2,4 from Agent[Term];
        action 'fwd_sig';
        out ringing,5 to User[Orig];
User[Orig]: in ringing,5 from Agent[Orig];
par end;
Agent[Term]: endinstance;
Agent[Orig]: endinstance;
User[Orig]: endinstance;
User[Term]: endinstance;
endmsc
```



Performance Analysis

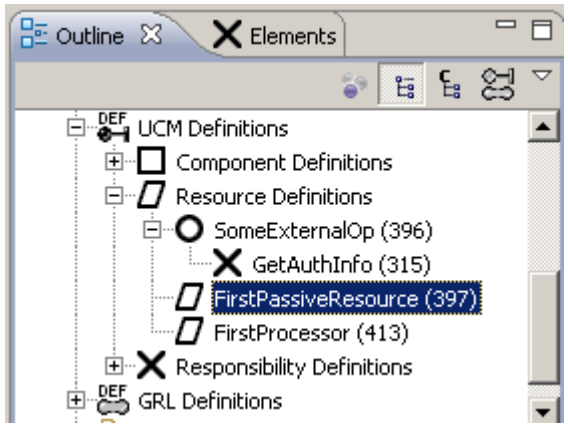
- Recall URN Example I
- Which of the three wireless IN alternative architectures is the best for this scenario?
  - Service and Data in MsgSwitchingCenter
  - Service in MsgSwitchingCenter, Data in ServiceNode
  - Service and Data in ServiceControlPoint
- Different complementary approaches
  - Qualitative analysis with GRL strategies
  - Transformations to MSCs for impact on messaging
  - UCM performance annotations and transformation to CSM for quantitative analysis

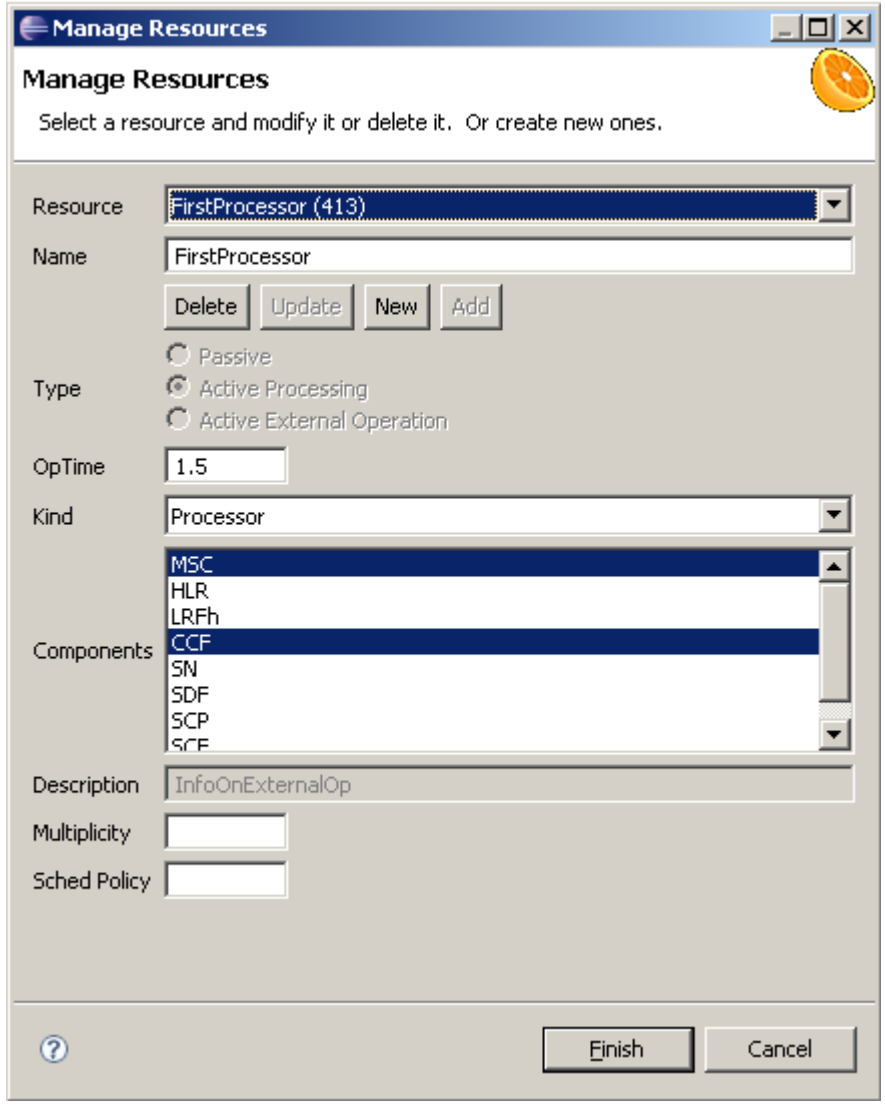
Performance Annotations for UCMs



- Automated translation to *Core Scenario Model (CSM)* for analytical evaluations and simulations

Resource Management





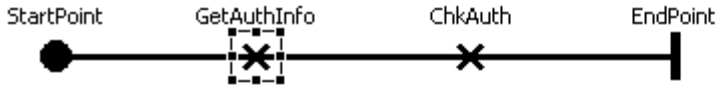
Demand and Workload Management

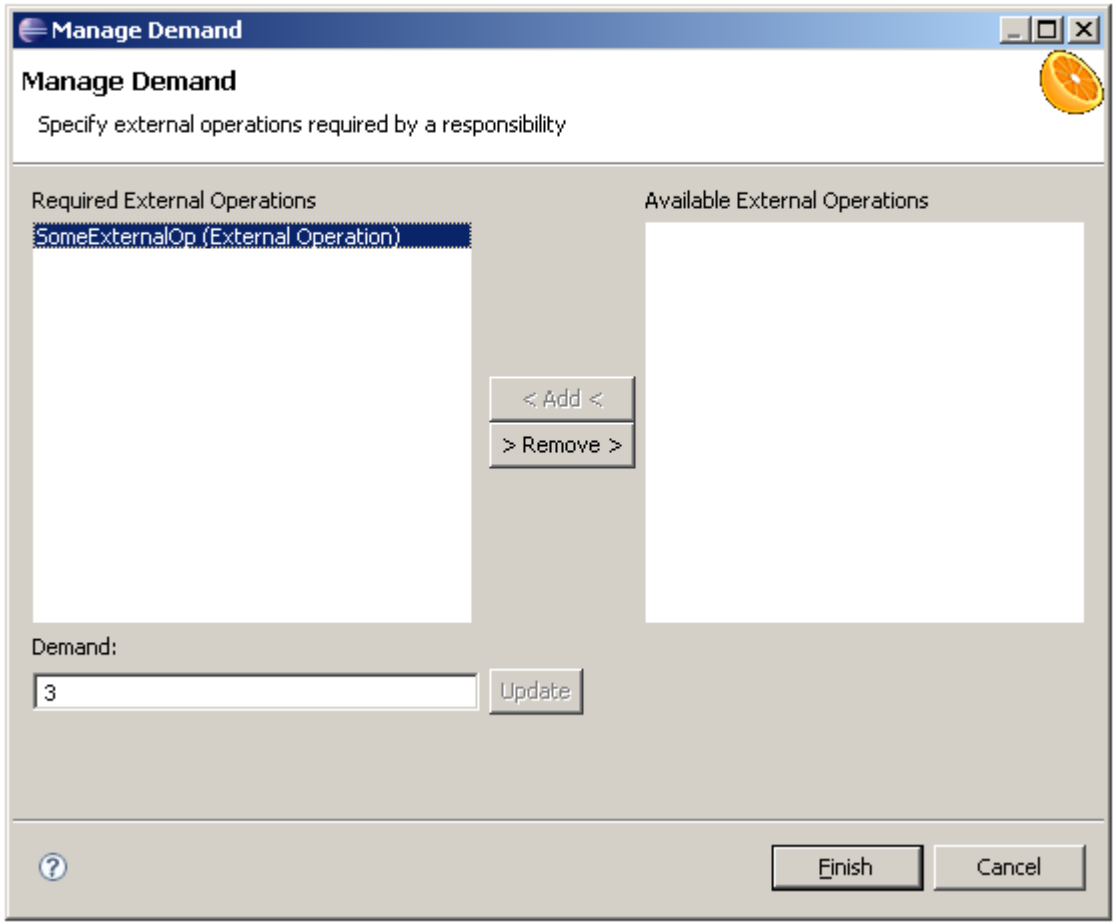
Properties | Problems | Key Performa

Property	Value
Miscellaneous	
hostDemand	0.4
repetitionCount	2

Properties | Problems | Key Performance Indica

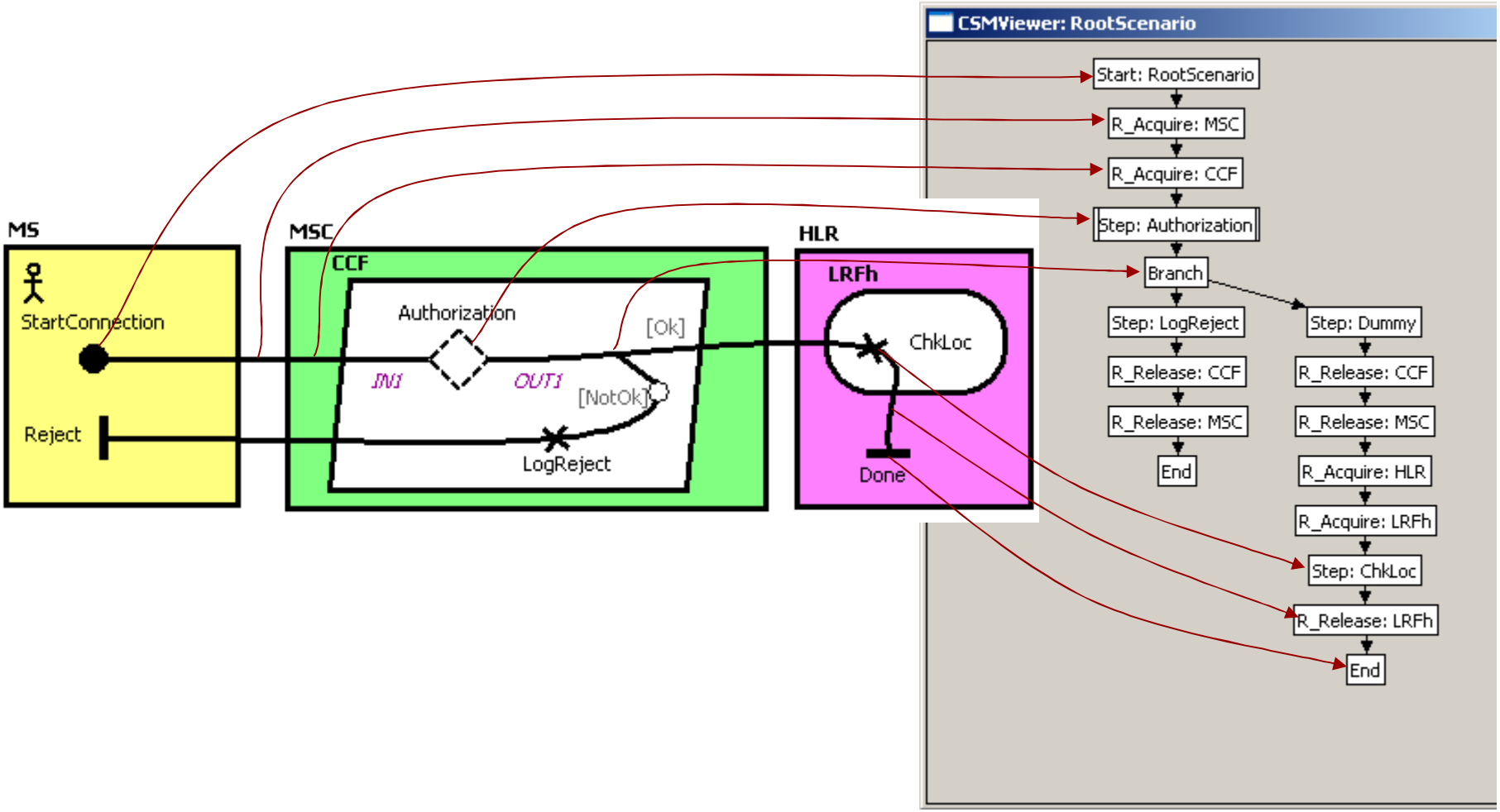
Property	Value
Workload	
arrivalParam1	0.5
arrivalParam2	0.8
arrivalPattern	PoissonPDF
closed	true
coeffVarSeq	
description	Sample arrival for start poi
externalDelay	
id	534
Metadata	[click to edit]
name	Workload
population	1000
value	





From UCM to Core Scenario Model (CSM)

- Export CSM (XML) from URN model
- Translation of CSM file to LQN, QN, stochastic Petri Nets…

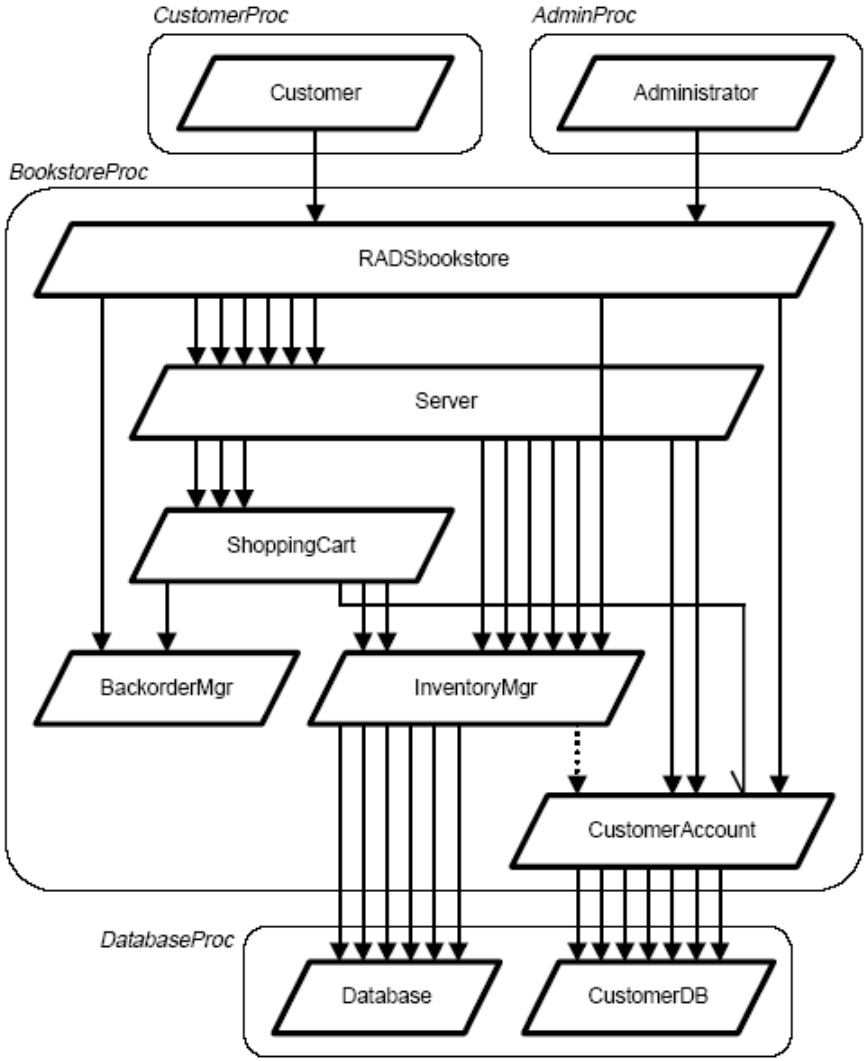


LQN Generation from UCMs (1)

- Layered Queueing Networks
  - Capture the workload within activities (operations connected in sequence or in parallel) which belong to an *entry* of a *task* (e.g. *method* of an operating system *process*) running on a *host device* (usually a *processor*)
- Solving LQN models
  - Analytic solver (LQNS) or simulator (LQSim)
  - Both developed at Carleton University
  - Solver is faster but more limited than simulator

LQN Generation from UCMs (2)

- Useful for various types of analyses
  - Sensitivity (importance or impact of parameters)
  - Scalability (what if there are more users/requests?)
  - Concurrency (what if there are more/fewer threads?)
  - Deployment and configuration (different hardware allocation)
- Quantitative evaluation of architecture!



Source: D.B. Petriu et al., 2003

Typical Performance Analysis Results…

- General statistics
  - Elapsed time, system time…
- Measured quantities
  - Service demands, number of blocking and non-blocking calls, call delays, synchronization delays
- Service times
  - For every entry and activity, with confidence intervals and variances (where relevant)
- Throughputs and utilizations for every entry and activity, with confidence intervals
- Utilizations and waiting times for devices (by entry)

UCM-Based Testing?

- Based on UCM Testing Patterns
  - Grey-box test selection strategies, applied to requirements scenarios
  - Manual
- Based on UCM Scenario Definitions
  - UCM + simple data model, initial values and start points, and path traversal algorithms
  - Semi-automatic
- Based on UCM Transformations
  - Exhaustive traversal
  - Mapping to formal language (e.g., LOTOS, ASM)
  - Automated

Comparison

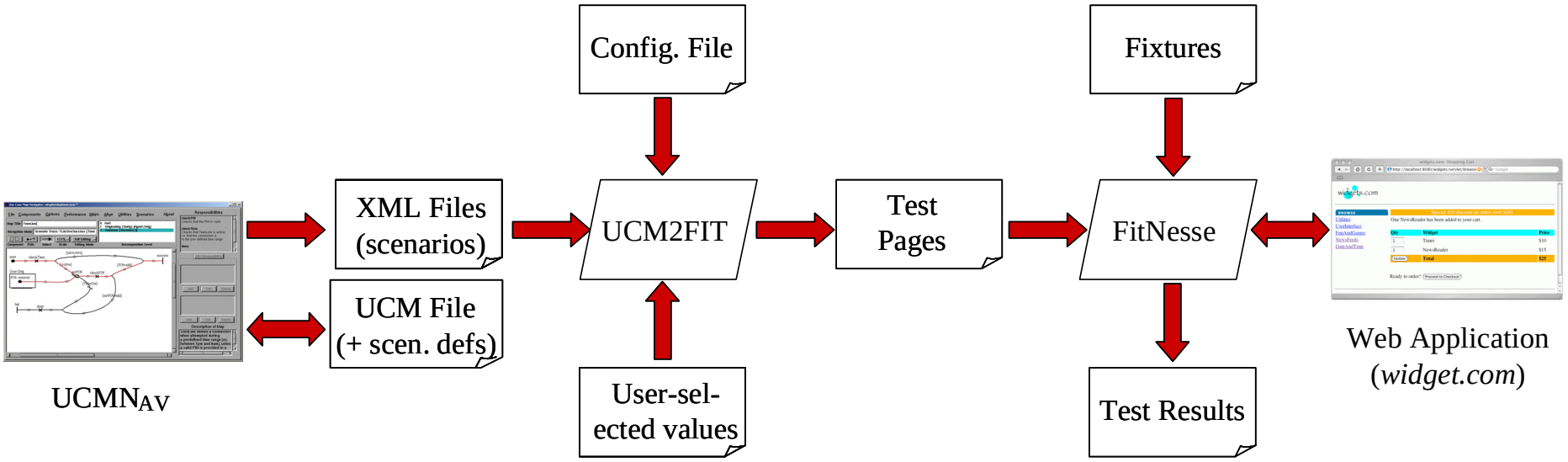
	Testing Patterns	Scenario Definitions	Automatic Transformations
Automation	✗	○	✓
Unfeasible scenarios	✗	✓	✗✗
Communication	✗	✓	○
Exhaustiveness	✗	✓	✗✗
Coverage	✓	✗	✓✓
Scalability	✗	✓	✗
Model Evolution	✗✗	✓	✓
Usability	✓	○	✓
Transformations	✗✗	✓✓	✓
Maturity	○	○	✗
Tool Support	✗✗	✓	○

Towards Test Case Generation

- Communication and calls
  - Messages, parameters, interfaces, protocols...
- Data
  - Must ensure that the scenario is feasible
- Temporal information
  - UCM timers currently have no quantitative time
- Implementation, sequencing, execution, clean-up
- Many other challenges!

- There are however some partial results available…
  - Use of jUCMNav, scenario definitions, and *Fitnesse* to generate executable test cases for a typical Web application

Test Generation for Web Applications



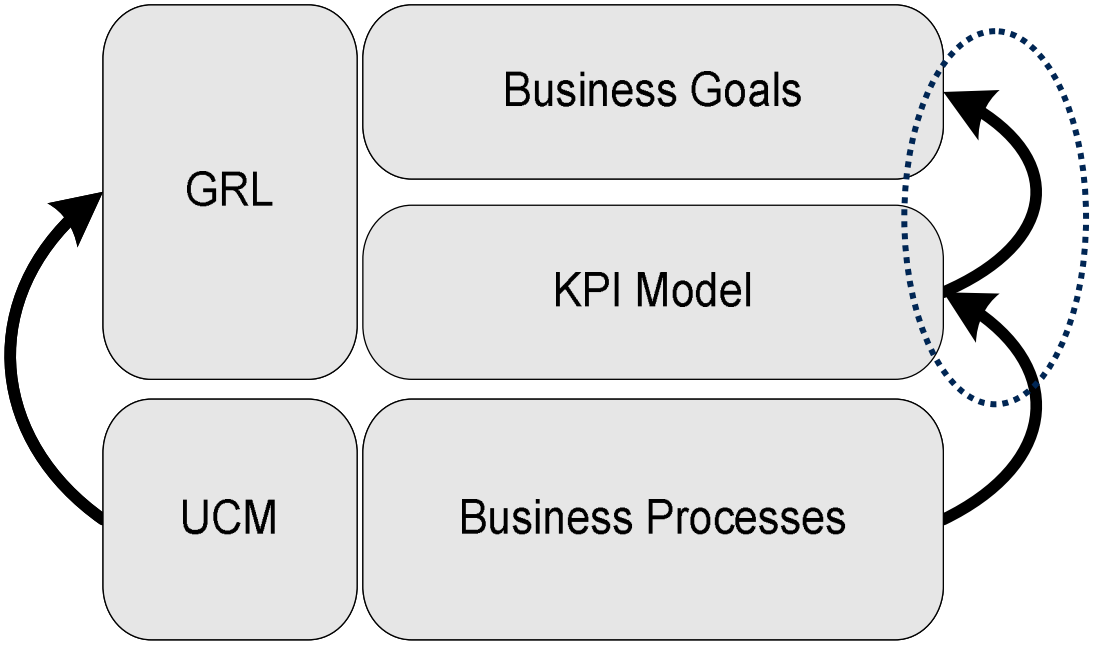
Source: Amyot, Roy, and Weiss, 2005

URN for Business Process Modeling?

- In a BPM tool we need to answer the W5 questions
  - URN can answer Where, What, Who, When, and Why
- Use Case Maps (UCM)
  - Responsibilities (What)
  - Components (Who and Where)
  - Scenarios and causal relationships (When)
- Goal-oriented Requirement Language (GRL)
  - Tasks (What)
  - Actors (Who and Where)
  - Business or system goals and rationales (Why)
- GRL & UCM
  - Link processes to business goals

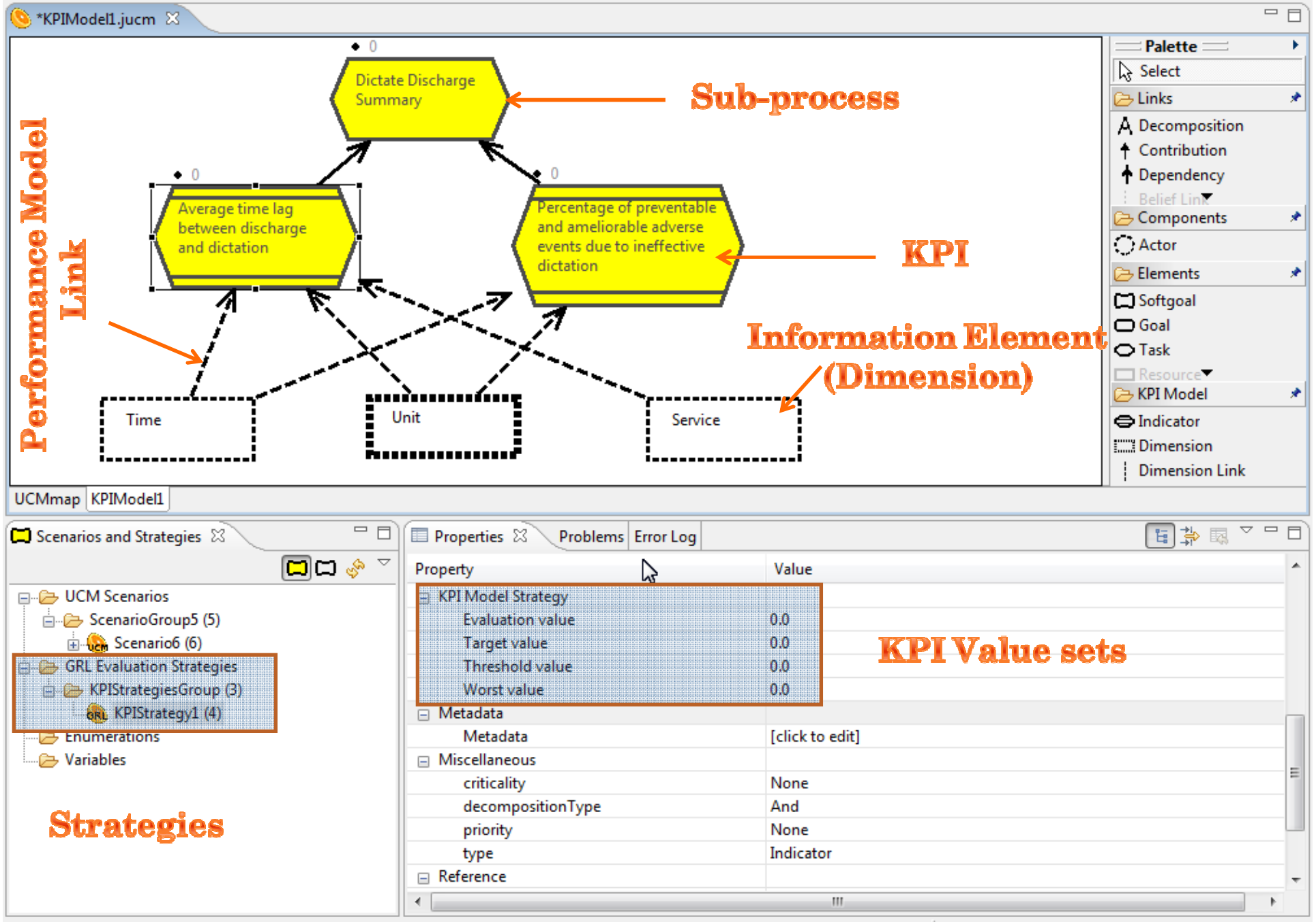
Business Process Analysis and Monitoring

- How can we model and monitor business processes and determine how well they meet their business goals and performance requirements?
- Can monitoring information be used to better align business processes and goals?

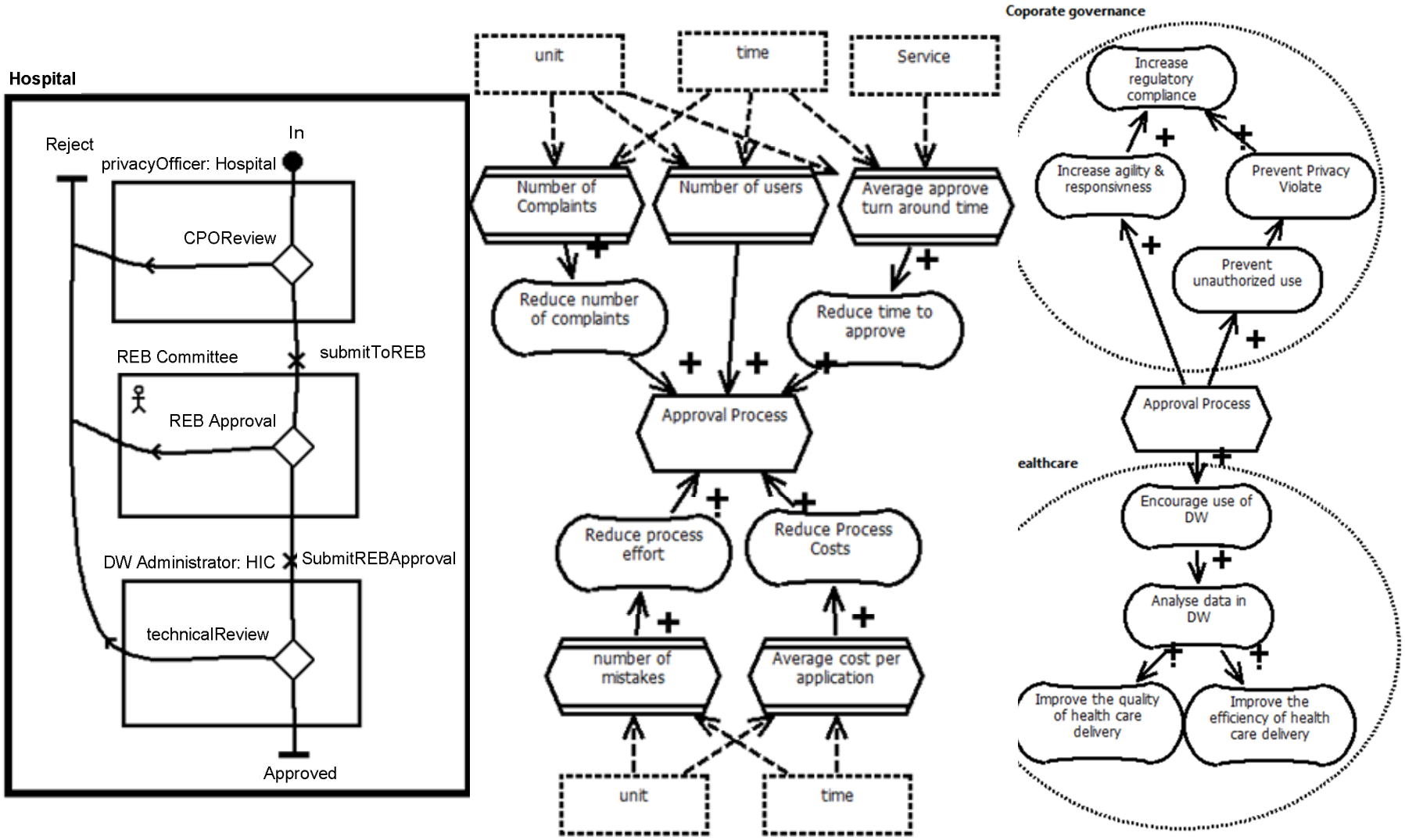


- KPI … Key Performance Indicator

GRL Editor with Key Performance Indicators

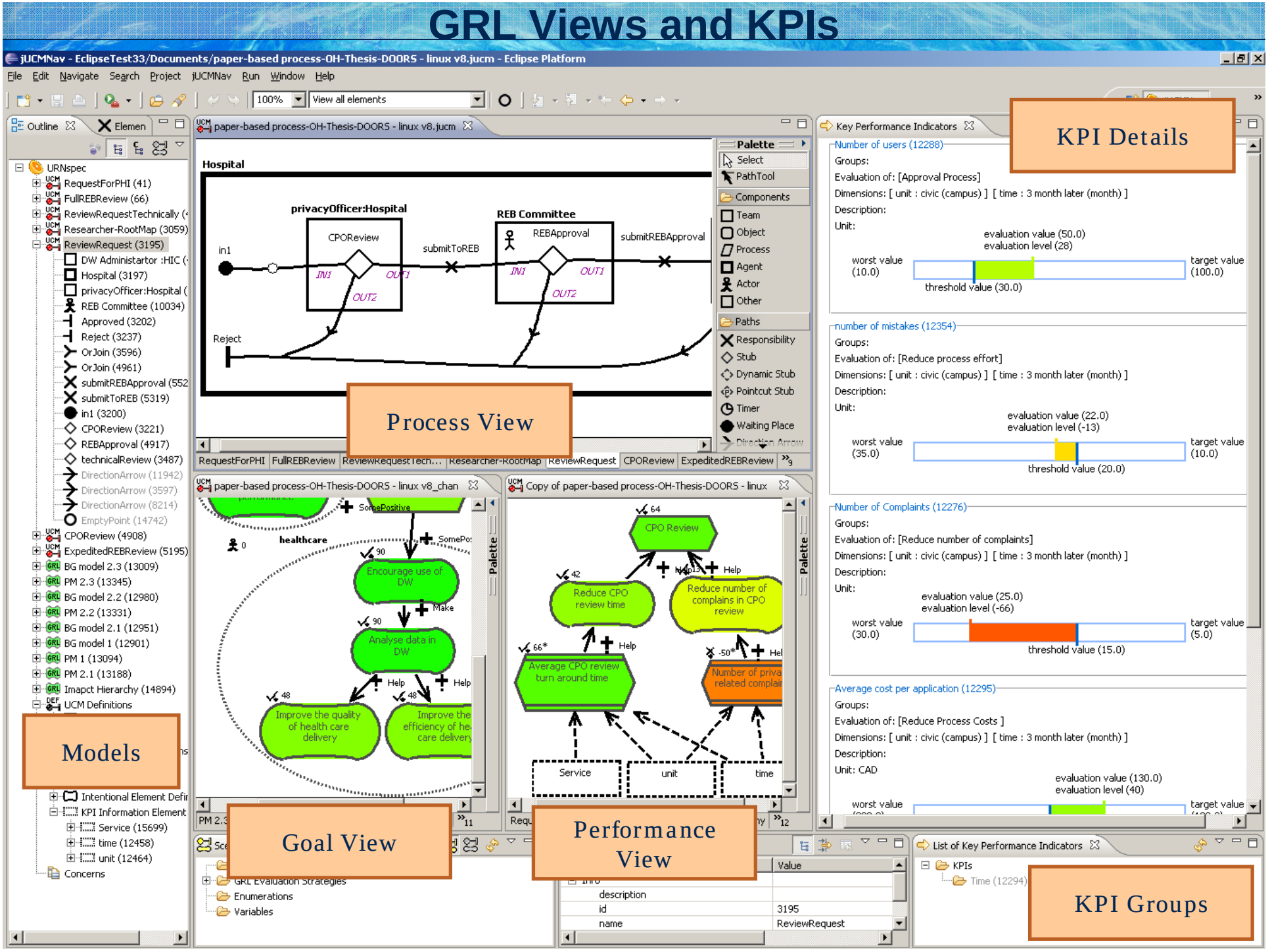


Three Connected Views in a URN Model with KPI

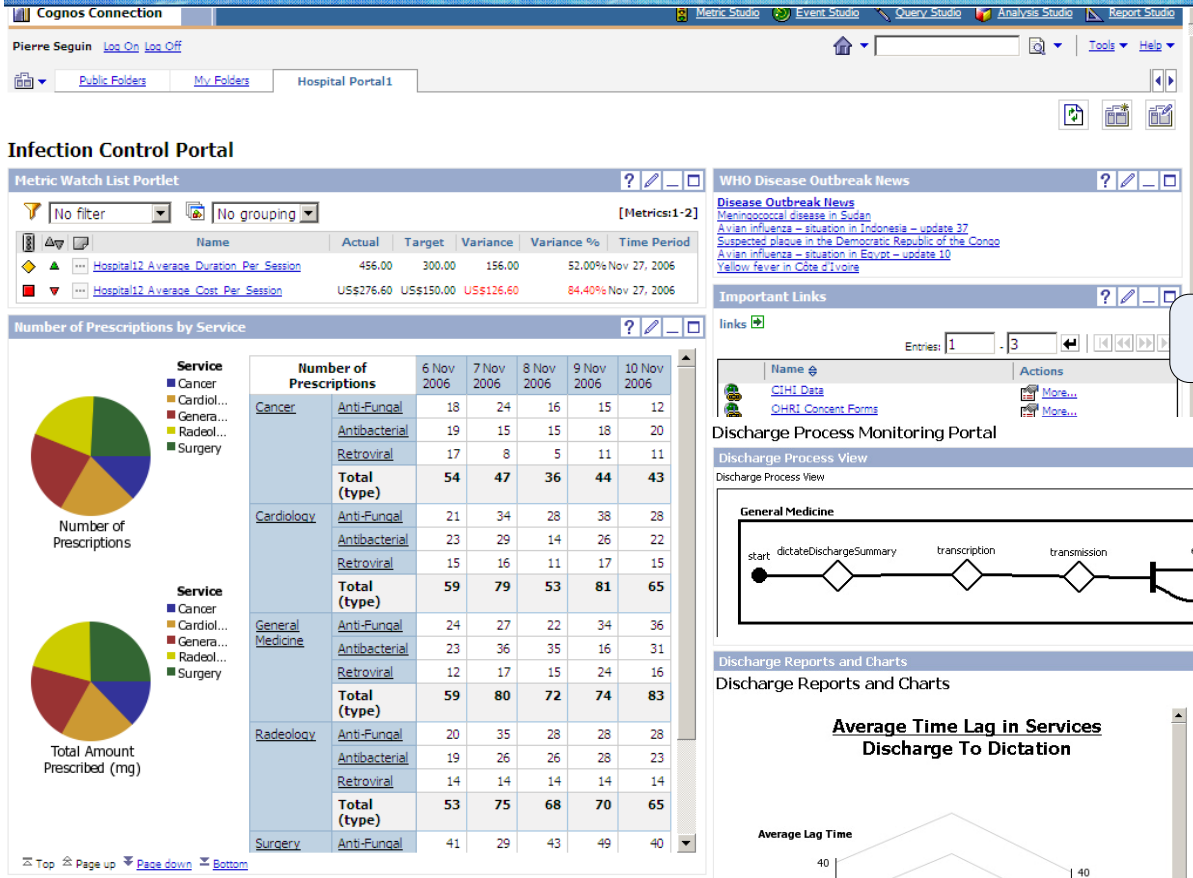


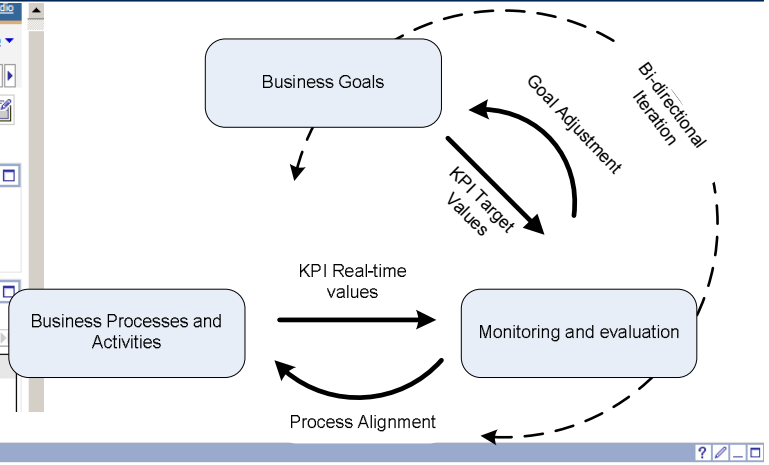
Process Model Performance Model Goal Model

GRL Views and KPIs



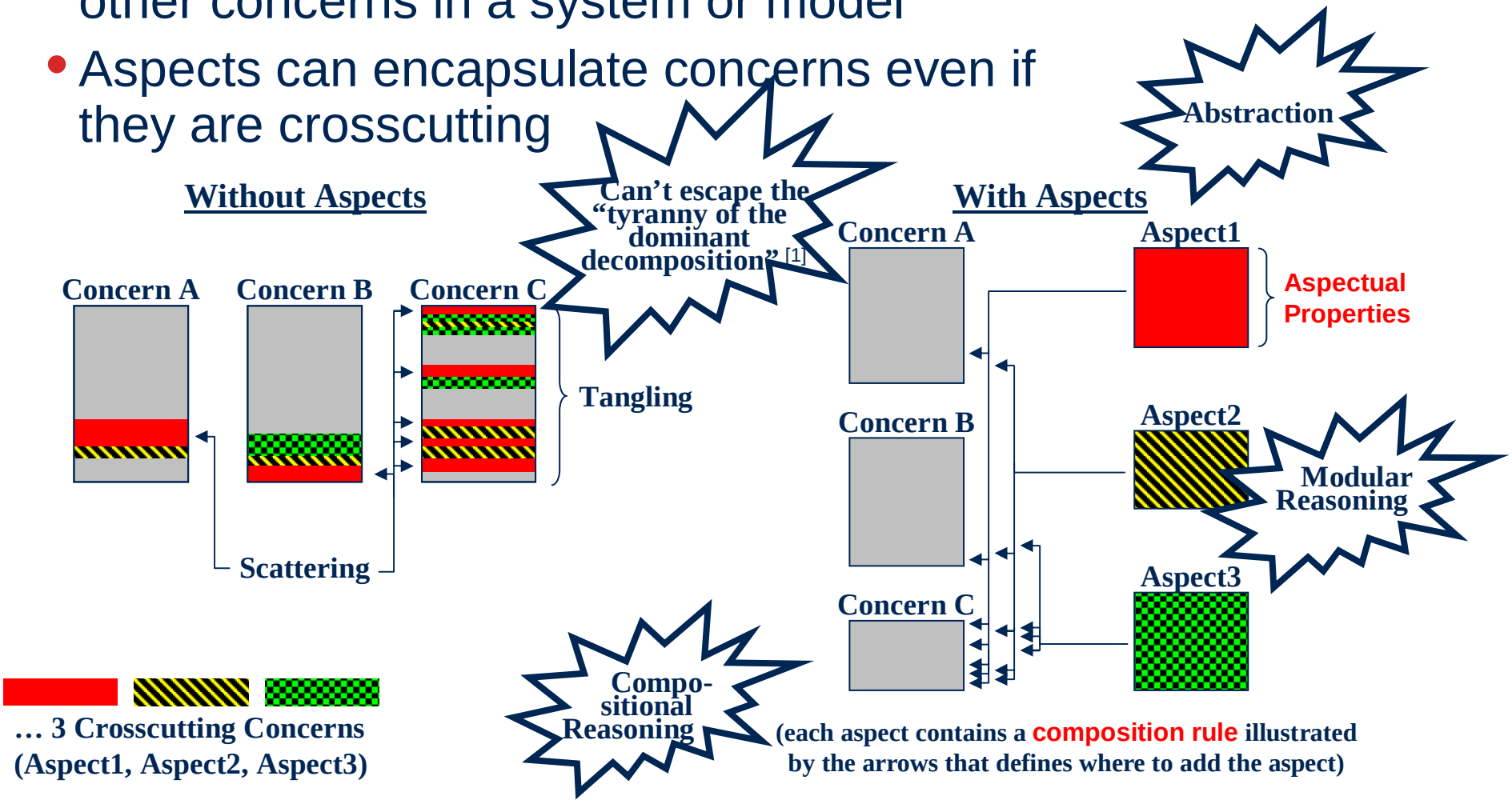
Integration with BI Tools (Cognos 8)





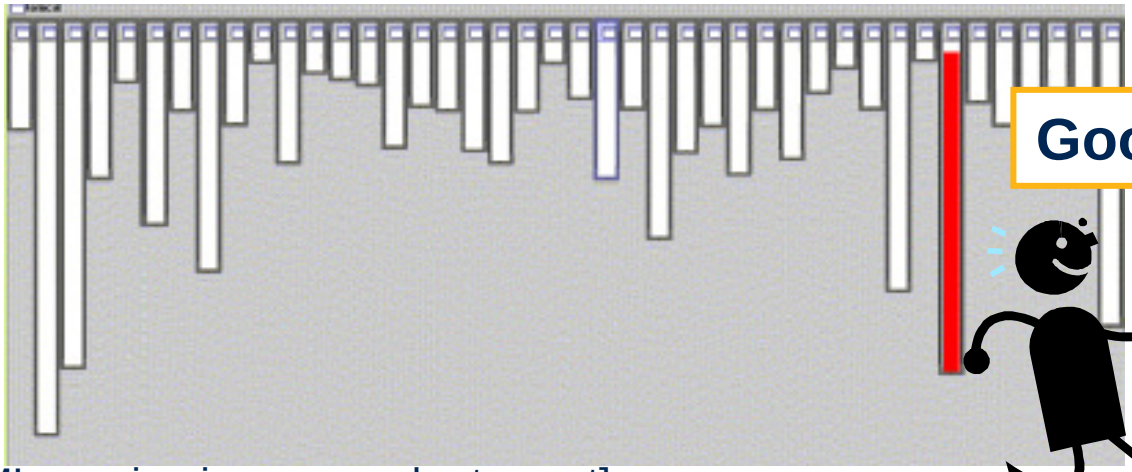
Support for Aspect-oriented Modeling in URN

- Aspects address the problem of one concern crosscutting other concerns in a system or model
- Aspects can encapsulate concerns even if they are crosscutting

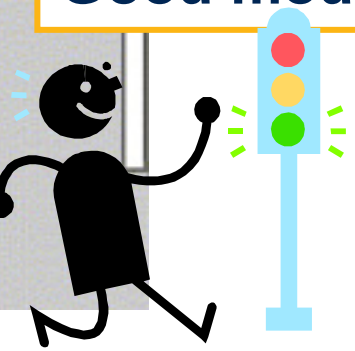


[1] Tarr, P., Ossher, H., Harrison, W., and Sutton, S.M.: N degrees of separation: Multidimensional separation of concerns. ICSE 99

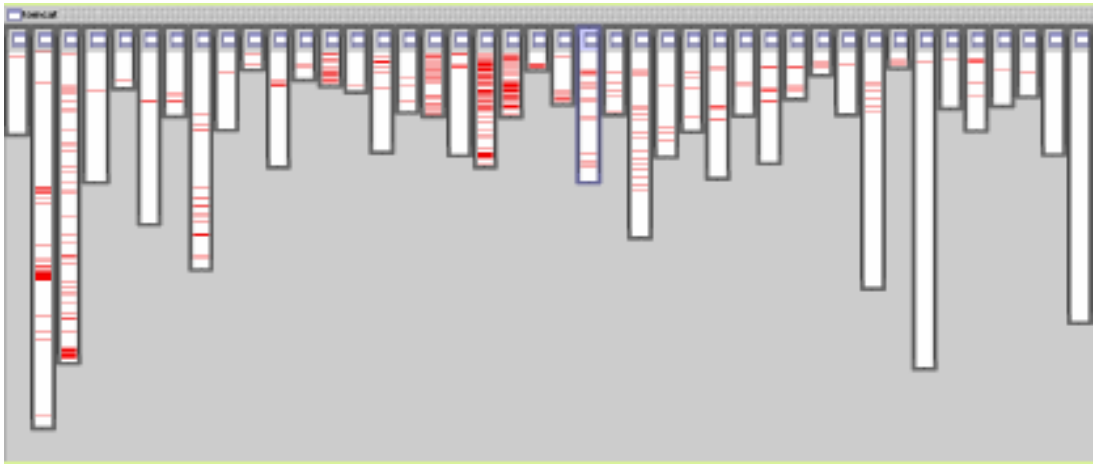
Crosscutting Concerns Affect Modularization



Good modularization



[XML parsing in org.apache.tomcat]

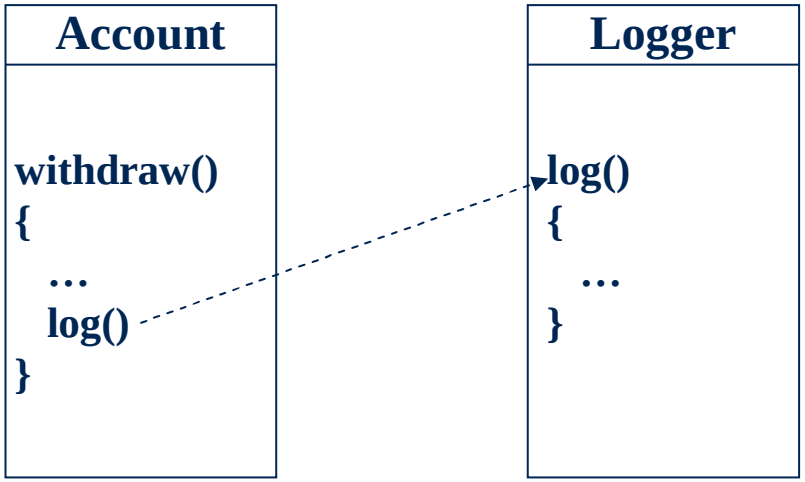


Bad modularization

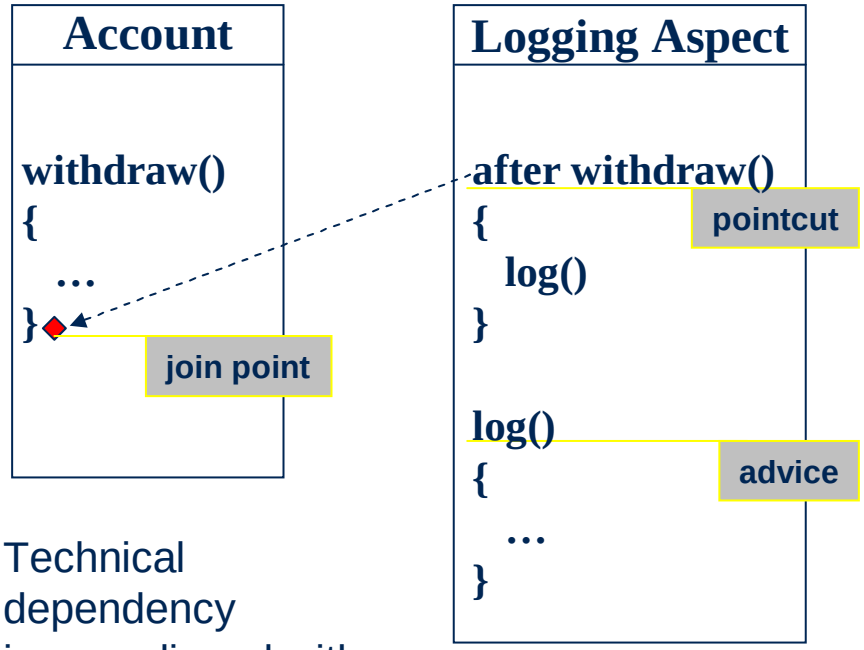
[logging in org.apache.tomcat]

Aspects-oriented Modeling – Dependencies

- Aspects make it possible to align technical dependencies with logical dependencies to a greater extent than before



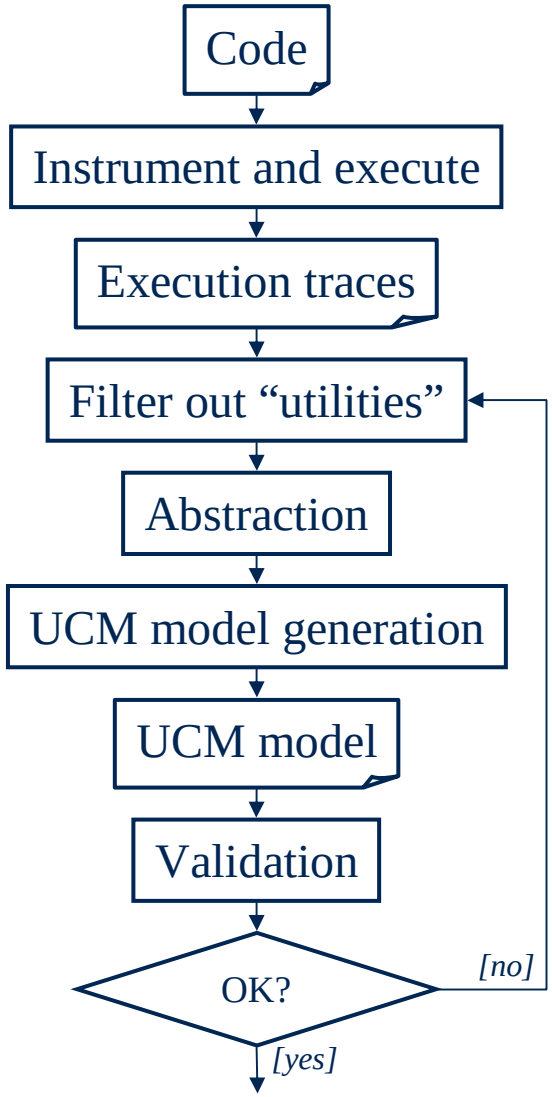
Account is dependent on Logger due to technical reasons only. Logically, an account is completely independent from a Logger. On the contrary, the Logger is logically dependent on what needs to be logged (i.e. the Account).



Technical dependency is now aligned with logical dependency.

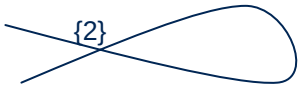
Generating Models from Code?

- Execution traces can help us understand functionalities and other dynamic aspects in an existing program
- But they are usually huge and impossible to understand
  - Sometimes millions of events!
- Need abstraction and visualisation
- UCMs provide and abstract view of scenarios



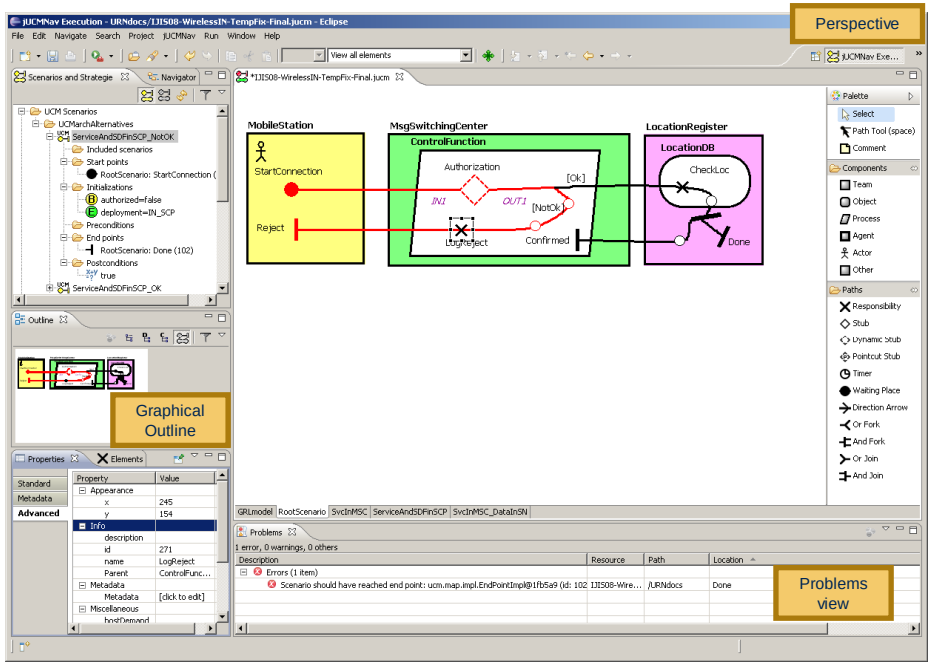
Source: A. Hamou-Lhadj et al., 2005

Correspondence of UCM Elements (Example)

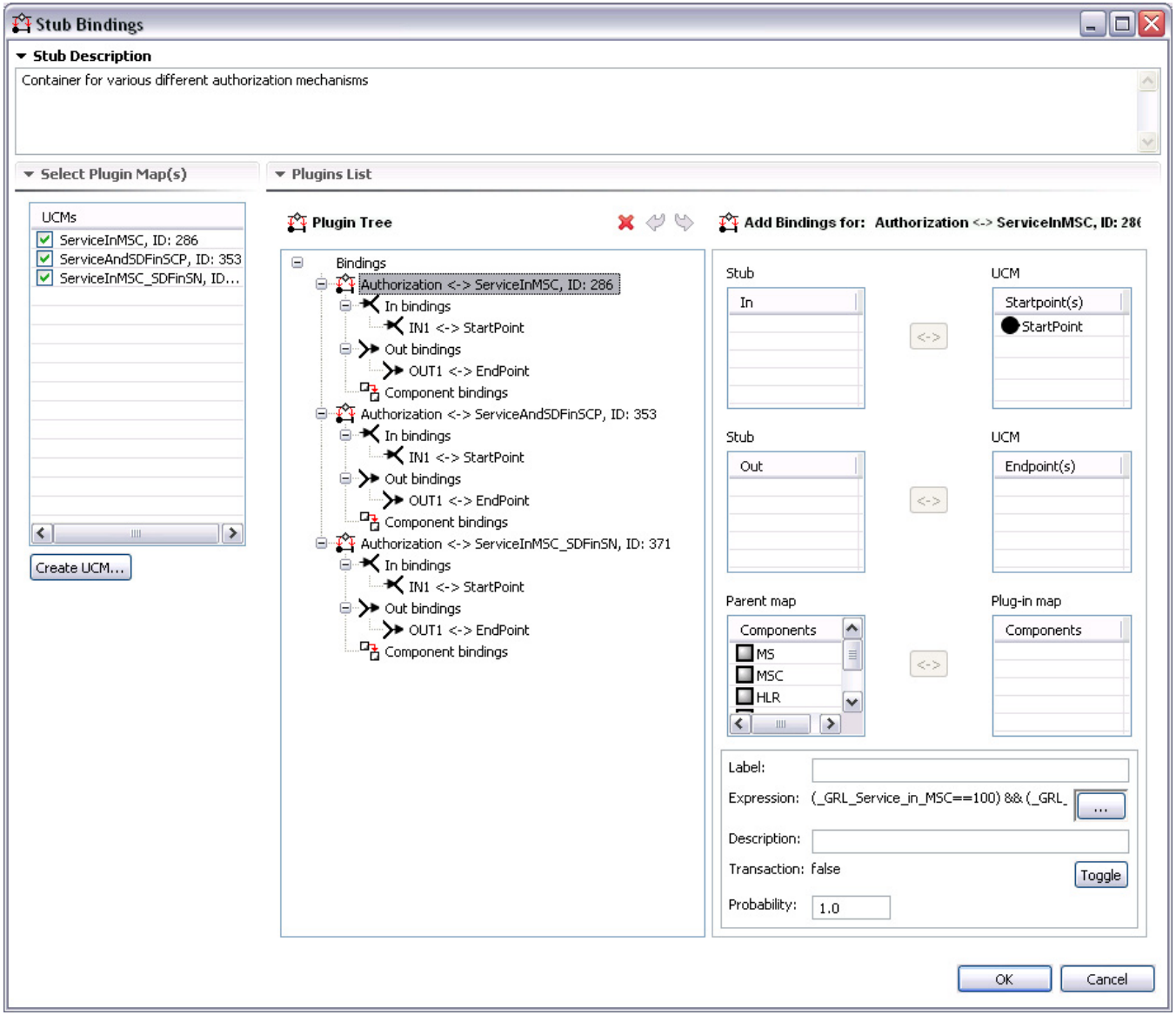
Trace element	UCM element	Symbol
Package	Component (Agent), shown as a rectangle with thick border.	
Class	Component (Team), shown as a rectangle with narrow border.	
Object	Component (Object), shown as a rounded-corner rectangle.	
Thread	Component (Process), shown as a parallelogram.	
Beginning / End of trace	Start point (circle) / End point (bar) (also used as connectors for linking sub-scenarios to the parent stub)	
Instruction	Responsibility (shown as a X on a path)	
Block of 3 or more instructions in the same class/object	Stub (diamond) with the name of the first instruction that is not a constructor. This stub contains a plug-in (another sub-map) showing the sub-sequence with one responsibility per instruction.	
Repeated instruction	Responsibility with repetition count (number between curly brackets)	
Repeated sequence	Loop (with loop count between curly brackets)	
Condition	Condition (between square brackets)	[cond] {2}

jUCMNav (Eclipse Plug-in)

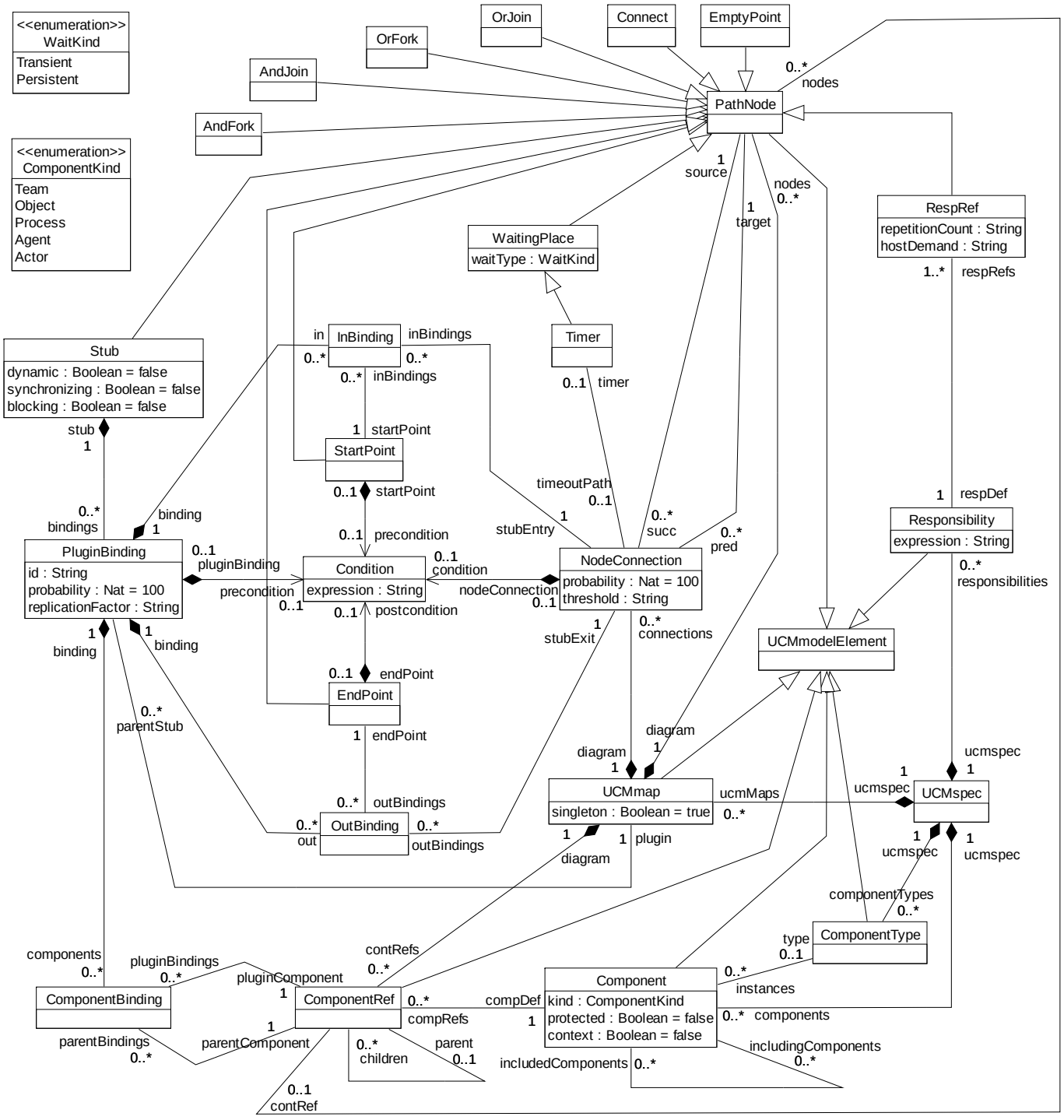
- Features for UCM
  - Scenario definitions
  - Traversal mechanism with color highlight and problems view
  - Many diagrams per model
  - Definitions and references of components and responsibilities
    - Drag&drop from outline or via properties
  - Auto-layout
  - Export graphics (.bmp, .gif, .jpg)
  - Export to Message Sequence Charts (MSC)
  - URN links (for integration with GRL)
  - Export to DOORS



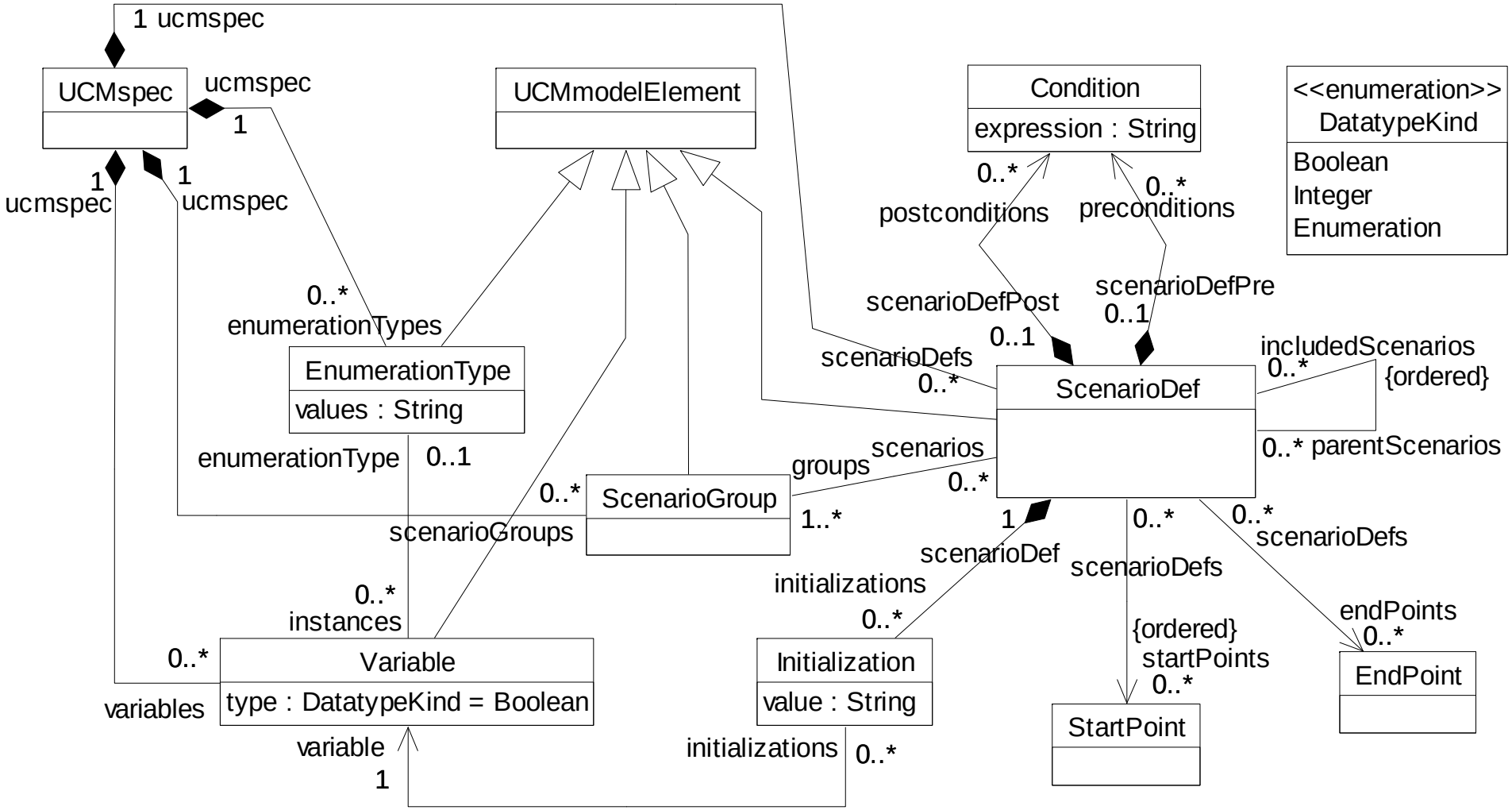
jUCMNav – Stub Plug-in Bindings



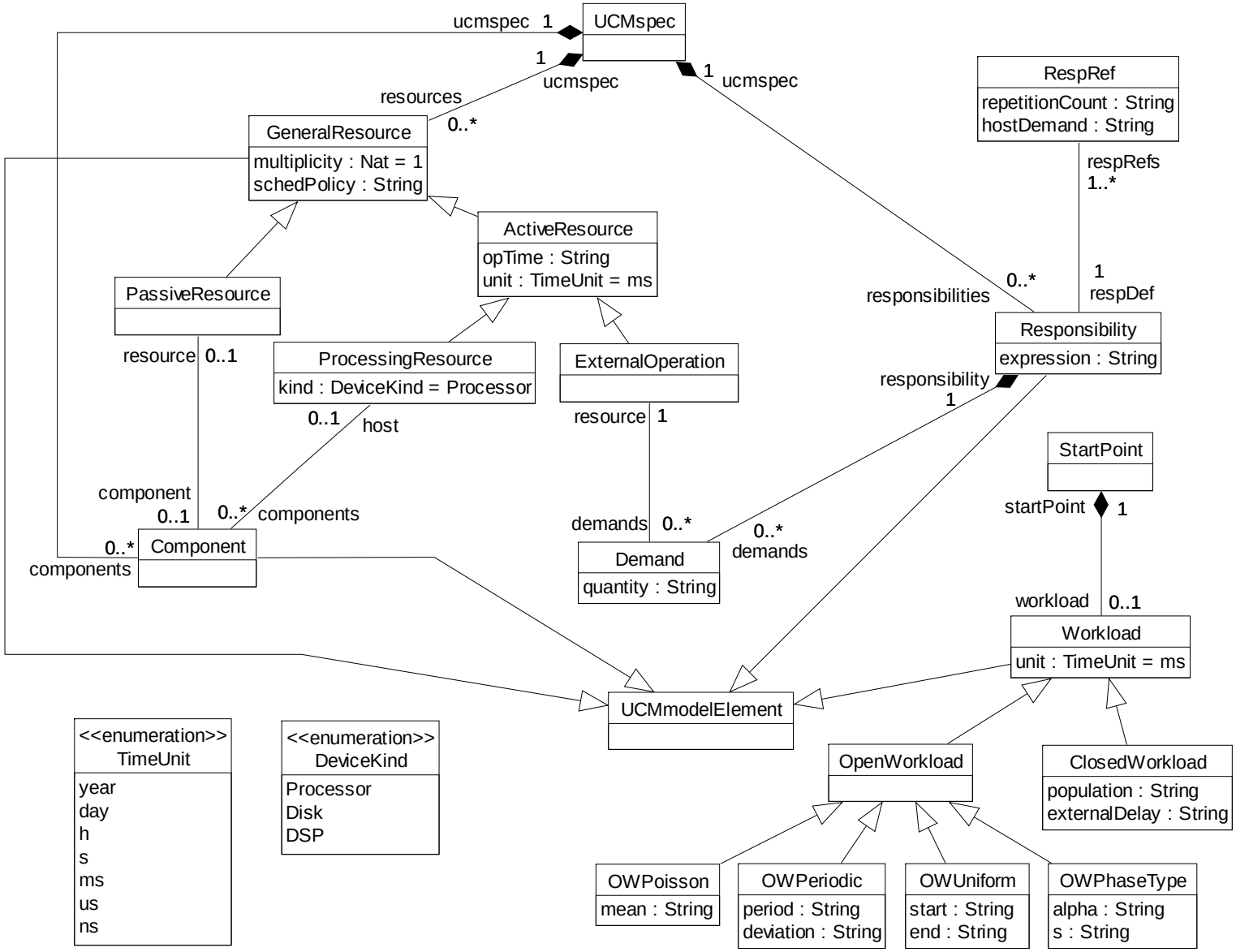
UCM Abstract Metamodel



UCM Scenarios Abstract Metamodel



UCM Performance Abstract Metamodel



Several Known URN Application Domains

- Telecommunication / telephony services
- Wireless systems
- Object-oriented software
- Multi-agent systems
- Web applications and Web services
- Railway control systems
- Embedded systems
- User interfaces
- Access control procedures
- Network protocols
- e-Business applications
- Supply chain management
- e-Health applications
- Business process management
- Software product lines
- Operating systems
- Information retrieval systems
- Vehicle communication systems
- …

Typical Usage of URN

- Modeling and documentation
  - User and system requirements, rationales
- Analysis of business goals
  - Evaluations of alternative requirements or solutions
  - Discovery of tradeoffs that can optimize the stakeholders' degree of satisfaction for conflicting goals
- Architecture analysis
  - Based on NFRs and design constraints
  - Performance analysis
- Generation of individual scenarios
  - Training, documentation
  - Detection of conflicts
  - Transformation to MSC and test cases
- Reverse-engineering
  - Abstract dynamic view of existing system

Key Points – URN Puzzle

- Goal-based and scenario-based notations complement each other
- URN fills a void in UML and ITU-T languages
- UCMs offer more capabilities than UML 1.X use case diagrams and activity diagrams
- UCMs offer features different from UML 2.x diagrams
- URN fits well into goal / scenario-based software develpmnt methodologies
- GRL provides the decision making framework for software engineering activities
- URN supports early activities in SW develpmnt, bringing together stakeholders with expertise in many different areas
- UCMs provide a good basis for design-time feature interaction detection and for model construction
- UCMs and GRL can be used iteratively and independently

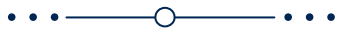
Conclusions

- URN
  - Allows engineers to specify or discover requirements for a proposed or an evolving system, and review such requirements for correctness and completeness.
  - Is usable in industry and in standardization bodies
  - Combines goals & scenarios
  - Helps bridging the gap between informal and formal concepts, and between requirements models and design models
  - Big benefits for little modeling investment, even when used informally
- GRL
  - For incomplete, tentative, (non-functional) requirements
  - Capture goals, objectives, alternatives, and rationales
- UCM
  - For operational and functional requirements
  - Enables analysis and transformations
  - Architectural alternatives and dynamic systems

Appendix: Notation Overview – UCM (Behavior)

Path with Start Point with Precondition CS and End Point with Postcondition CE

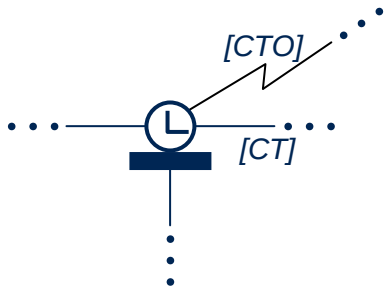
Responsibility



Empty Point

Direction Arrow

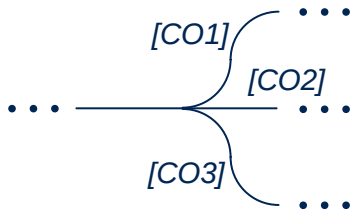
Waiting Place with Condition and Asynchronous Trigger



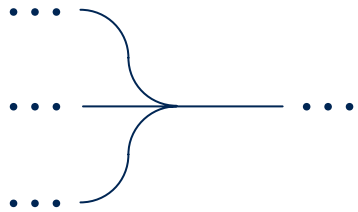
Timer with Timeout Path, Conditions, and Synchronous Release

Static Stub with In-Path ID and Out-Path ID

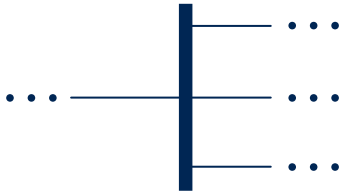
Dynamic Stub with In-Path ID and Out-Path ID



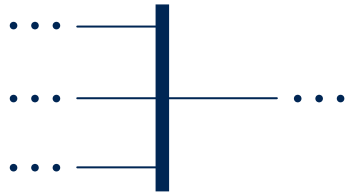
Or-Fork with Conditions



Or-Join



And-Fork



And-Join

Synchronizing Stub with In-Path ID, Out-Path ID, and Synchronization Threshold

Blocking Stub with In-Path ID, Out-Path ID, Synchronization Threshold, and Replication Indicator

Appendix: Notation Overview – UCM (Structure)

Components: