Carleton UNIVERSITY

McGill

uOttawa
L'Université canadienne
Canada's university

2 December, 2013

Introduction to the Aspect-oriented User Requirements Notation (AoURN): Aspects, Goals, and Scenarios

Gunter Mussbacher

ECE, McGill University, Canada ◄► gunter.mussbacher@mcgill.ca

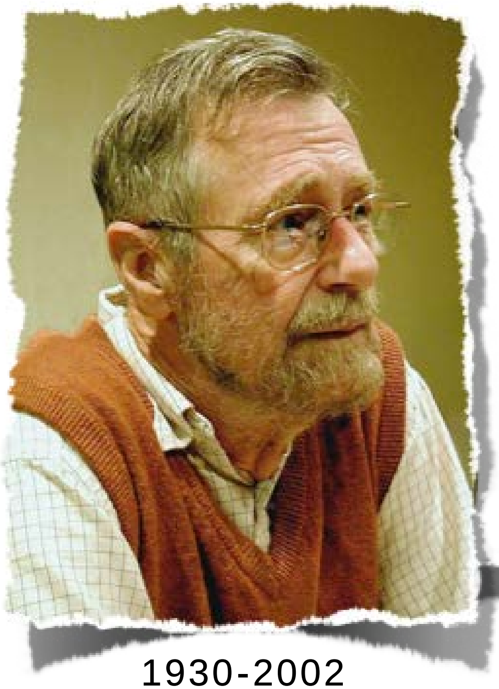
Table of Contents

- Introduction to Aspect-oriented Requirements Engineering (AORE)
- The Aspect-oriented User Requirements Notation (AoURN): Learning by Example
- Advanced Features of AoURN
  - Interleaved and Semantics-Based Composition
  - Analysis Features
- Conclusion and References

Introduction to Aspect-oriented Requirements Engineering (AORE)

Separation of Concerns

This is what I mean by focusing one's attention upon a certain aspect; it does not mean completely ignoring the other ones, but temporarily forgetting them to the extent that they are irrelevant for the current topic.



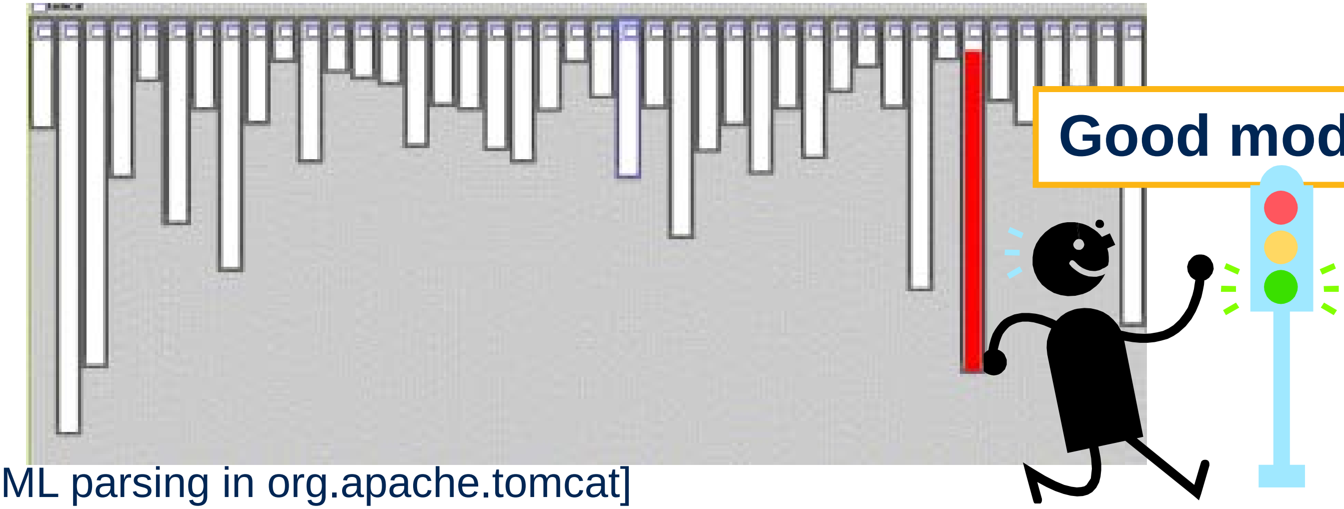
1930-2002

Such a separation, even if not perfectly possible, is yet the only available technique for effective ordering of one's thoughts that I know of. I usually refer to it as a separation of concerns.

Edsger Dijkstra, 1976

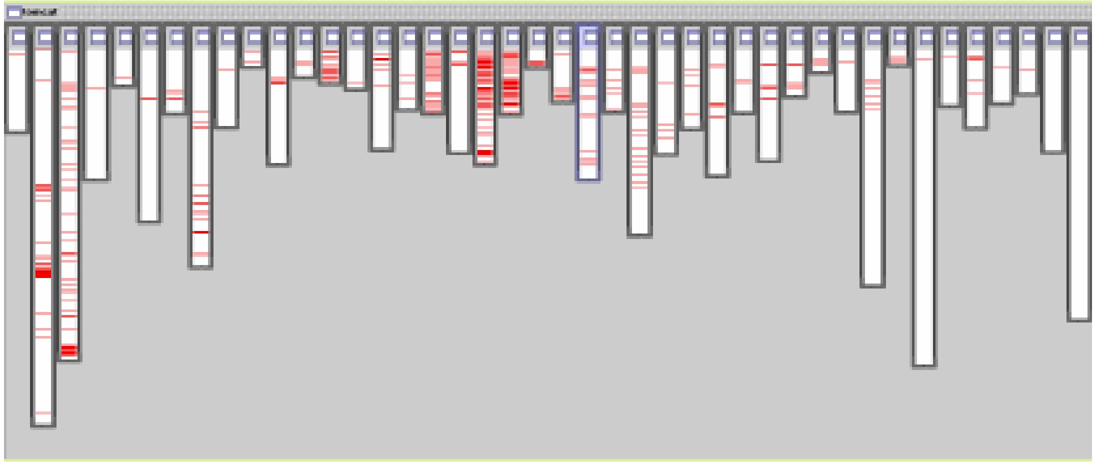
Source: E. Dijkstra, A Discipline of Programming, Prentice Hall, 1976, pp. 210

Crosscutting Concerns Affect Modularization



Good modularization

[XML parsing in org.apache.tomcat]

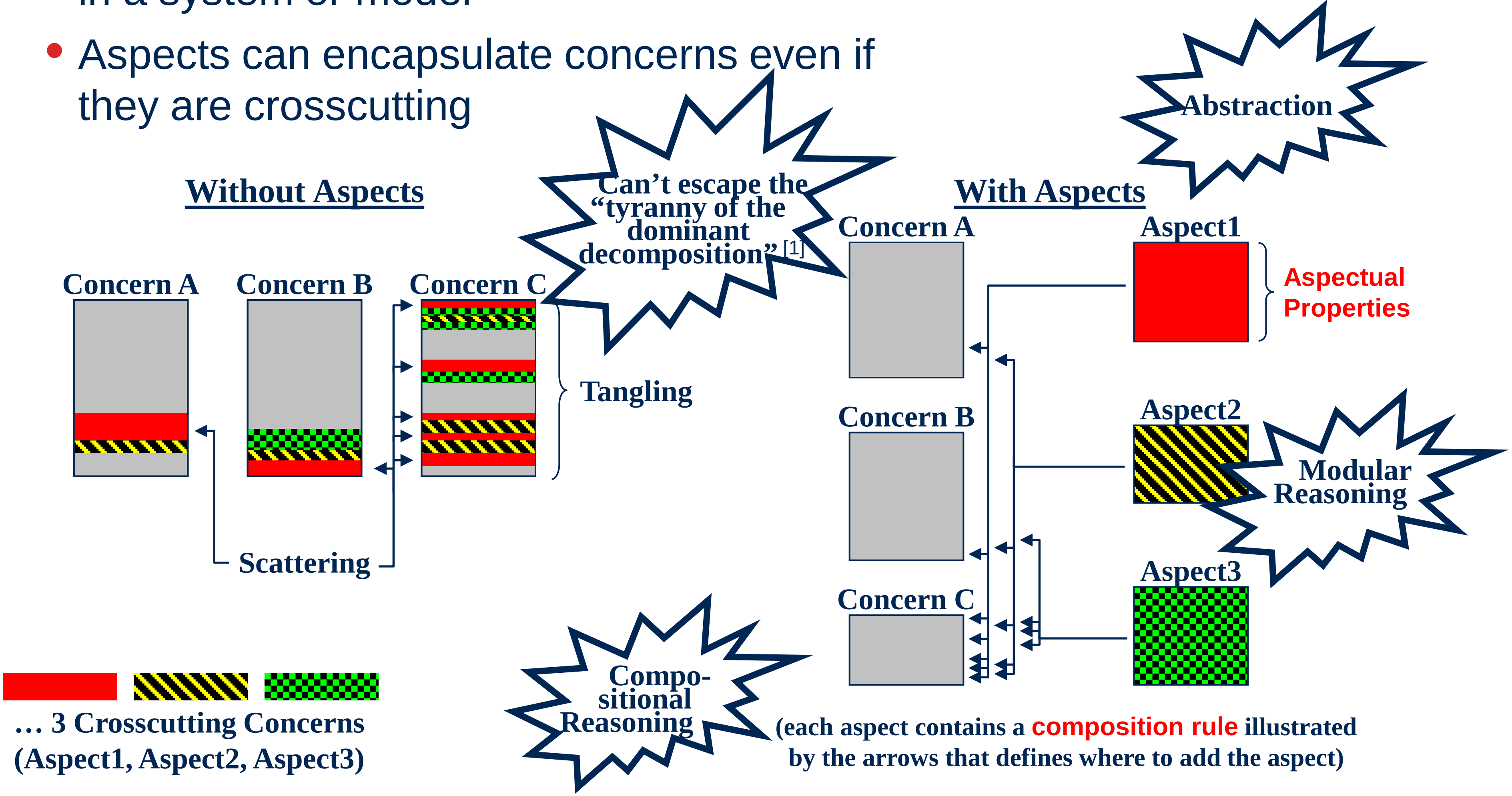


Bad modularization

[logging in org.apache.tomcat]

Overview of Aspect-oriented Modeling (AOM)

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



🟥 ▨ 🟩

… 3 Crosscutting Concerns (Aspect1, Aspect2, Aspect3)

(each aspect contains a composition rule illustrated by the arrows that defines where to add the aspect)

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

Main Value of Aspect-Orientation

- **Abstraction**: abstract away from the details of how crosscutting concerns might be scattered and tangled with the properties of other concerns
  - Helps with the identification of crosscutting concerns at early stages of software development
- **Modularization**: keep crosscutting concerns separated regardless of how they affect or influence various other concerns, so that we can reason about each concern in isolation – **Modular Reasoning**
  - Addresses problems caused by scattering and tangling
- **Composition**: the various concerns need to relate to each other in a systematic and coherent fashion so that one may reason about the global or emergent properties of the system – **Compositional Reasoning**
  - Addresses the lack of a systematic mechanism to describe concern influences
  - Provides a way to assess the impact of concerns (note though that evaluation mechanisms for goal models offer some support for that for high level goals which are often nonfunctional requirements (NFRs))

Source: A. Rashid, A. Moreira, Domain Models are NOT Aspect Free, Proceedings of MoDELS Conference 2006, Springer.

Aspect-Oriented Requirements Engineering

- Leverage the benefits of aspect-orientation in terms of abstraction, modularity, and composability
- Improved support for separation of crosscutting functional and non-functional properties during requirements engineering
  - A better means to identify and manage conflicts arising due to tangled representations
- Identify the mapping and influence of requirements-level aspects on artefacts at later development stages
  - Establish critical trade-offs even before the architecture is derived
- Trace aspectual requirements and their trade-offs to architecture and subsequently all the way to implementation

Improved understanding of the problem and ability to reason about it

Managing Crosscutting Concerns in Requirements (1)

- To find crosscutting in requirements, look for behavioural terms or concepts that are mentioned in more than one location

1. Pay interest of a certain percent on each account making sure that the transaction is fully completed and an audit history is kept.
2. Allow customers to withdraw from their accounts, making sure that the transaction is fully completed and an audit history is kept.

Managing Crosscutting Concerns in Requirements (2)

- Separate each of those concepts into a section of their own

1. *Pay interest* of a certain percent on each account

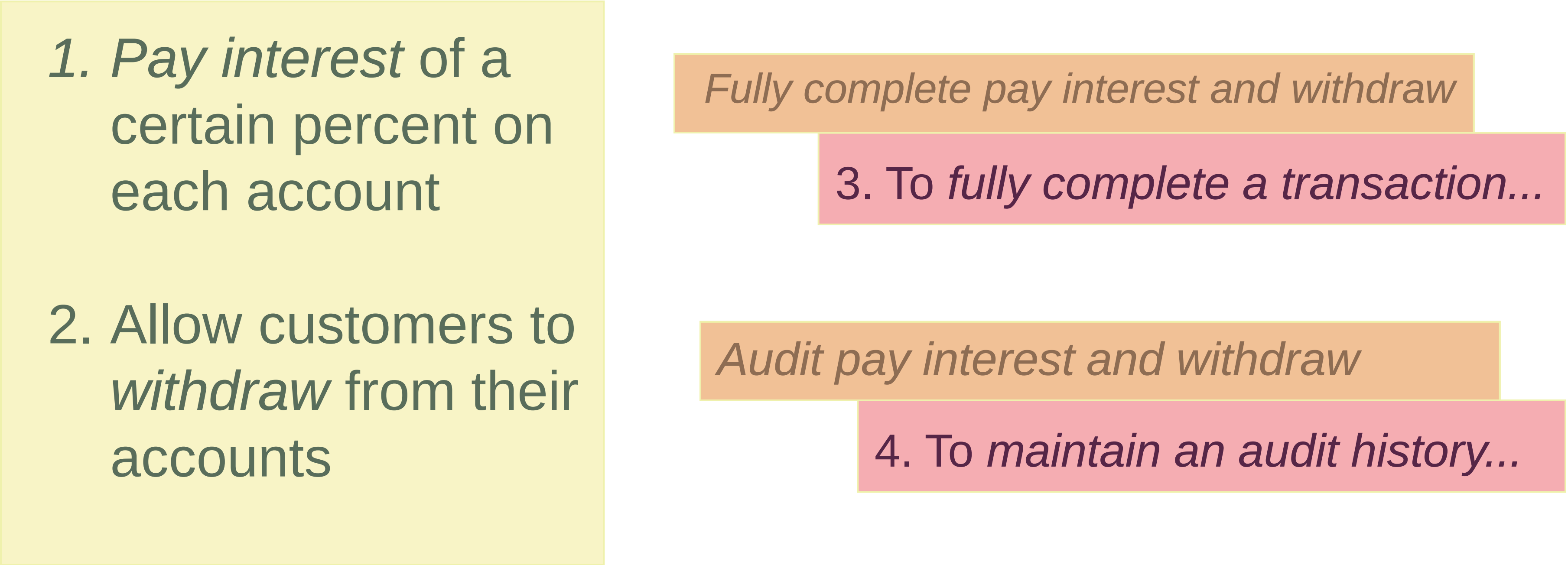
2. Allow customers to *withdraw* from their accounts

3. To *fully complete a transaction...*

4. To *maintain an audit history...*

Managing Crosscutting Concerns in Requirements (3)

- Composition specifies the crosscutting relationship, showing how crosscutting concerns affect base concerns



1. Pay interest of a certain percent on each account

2. Allow customers to *withdraw* from their accounts

Fully complete pay interest and withdraw

3. To *fully complete a transaction...*

Audit pay interest and withdraw

4. To *maintain an audit history...*

Carleton UNIVERSITY

McGill

uOttawa
L'Université canadienne
Canada's university

The Aspect-oriented User Requirements Notation (AoURN): Learning by Example

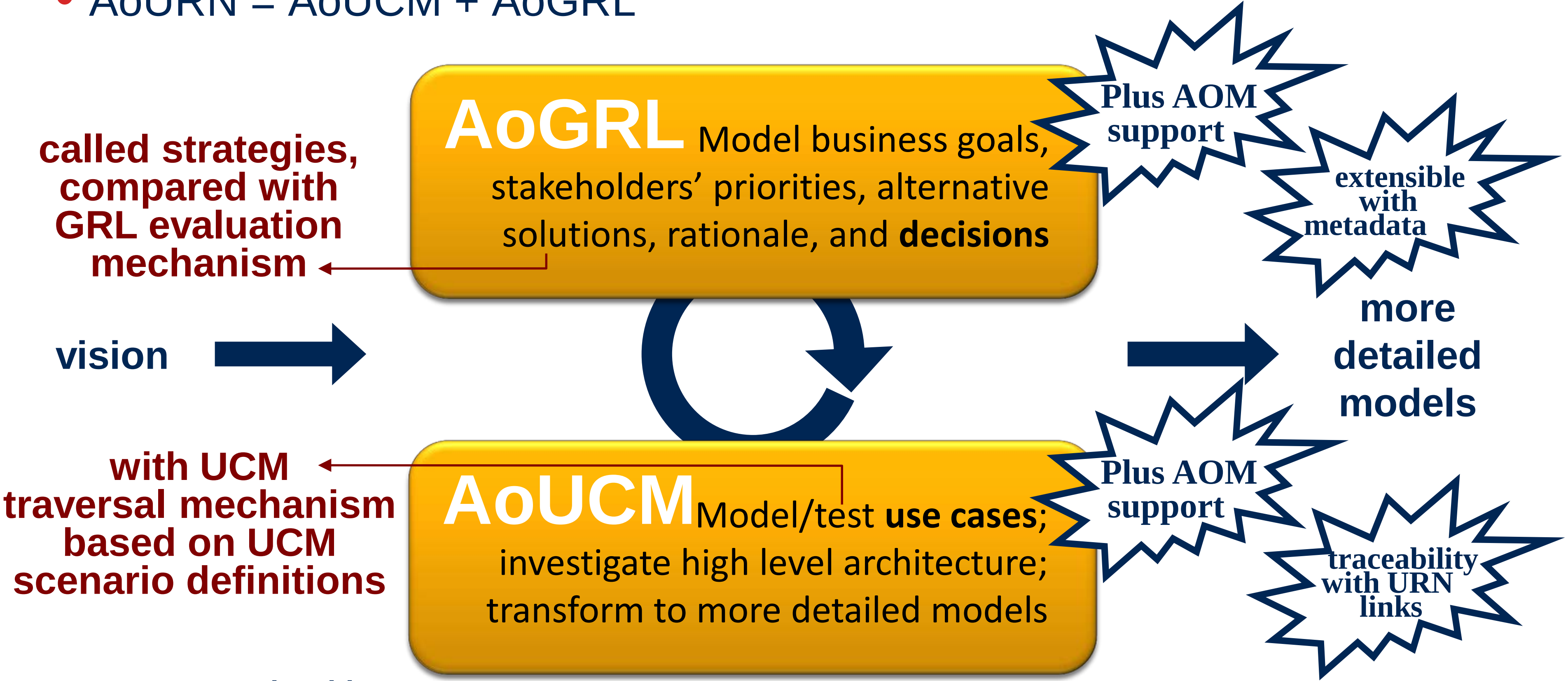
Motivation

- Aspects have the potential to significantly influence the future of software development like no other software engineering concept since the emergence of object-oriented techniques
- The User Requirements Notation (URN) is the first and currently only standard (ITU-T Z.151) which explicitly combines goals (non-functional requirements) and scenarios (functional requirements)

- Aspects can improve the modularity, reusability, scalability, maintainability and other properties of URN models
- Aspects can help bridge the gap between goals and operational scenarios
- Aspects can benefit from a standardized way of modeling NFRs and use cases with URN, considering the strong overlap between …
  - NFRs and crosscutting concerns
  - Stakeholder goals and concerns
  - Use cases and concerns

Objective

- The objective is to deliver a notation and process which unify URN concepts and aspect concepts in one framework in order to encapsulate concerns from the early requirements stage in the software development life cycle
- AoURN = AoUCM + AoGRL



AoGRL Model business goals, stakeholders' priorities, alternative solutions, rationale, and **decisions**

Plus AOM support

extensible with metadata

called strategies, compared with GRL evaluation mechanism

vision

more detailed models

AoUCM Model/test **use cases**; investigate high level architecture; transform to more detailed models

Plus AOM support

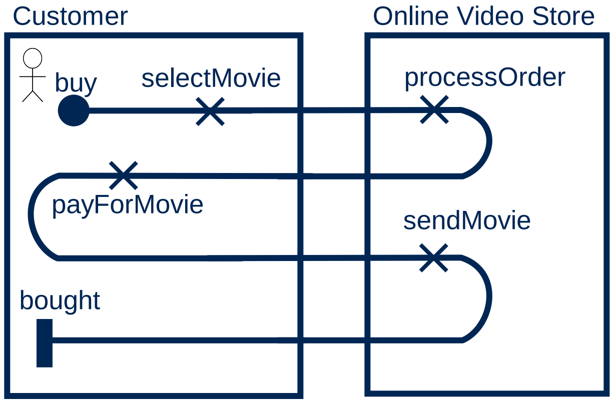
traceability with URN links

with UCM traversal mechanism based on UCM scenario definitions

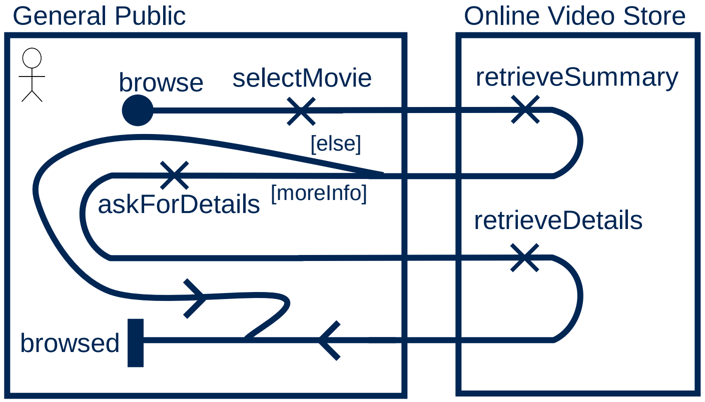
AOM … Aspect-oriented Modeling

AoURN: Learning by Example (1)

- Buy Movie Concern

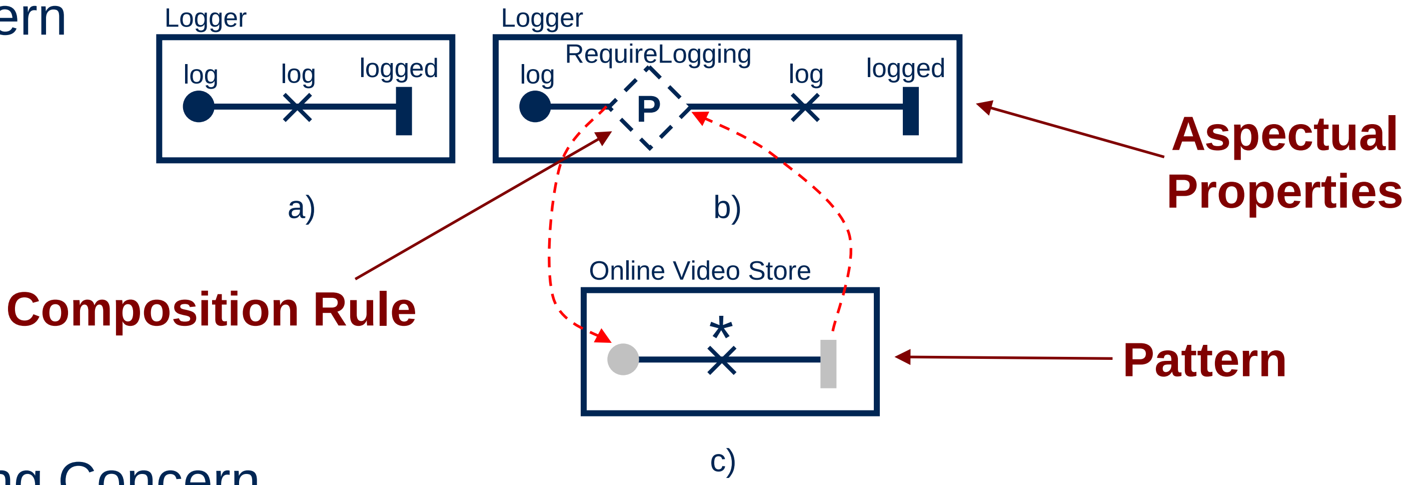


- Browse Movie Concern



AoURN: Learning by Example (2)

- Logging Concern (b + c)

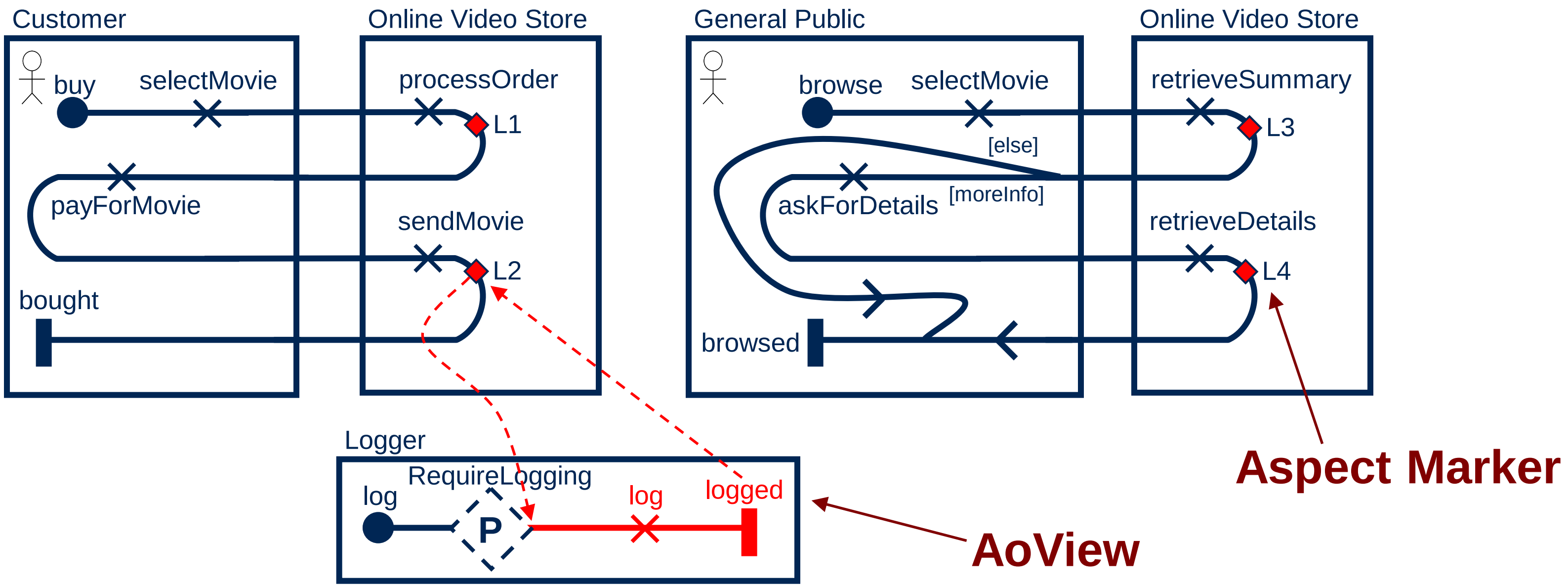


Aspectual Properties

Composition Rule

Pattern

- Applied Logging Concern

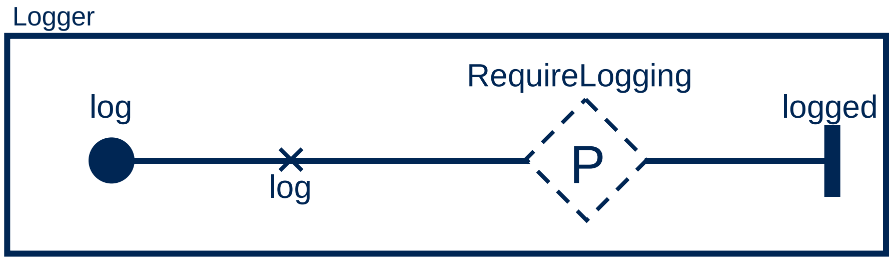


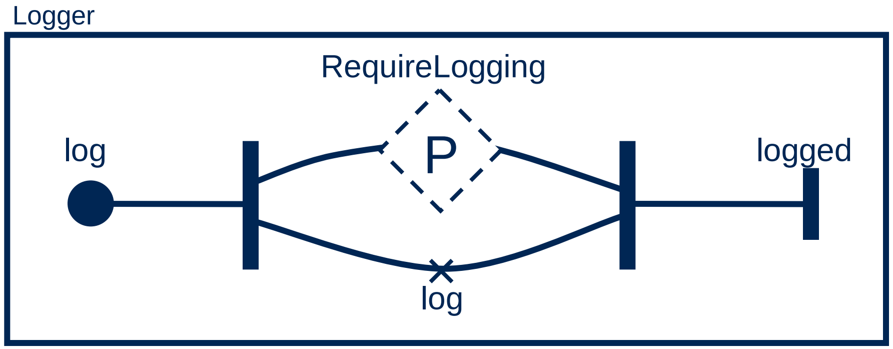
Aspect Marker

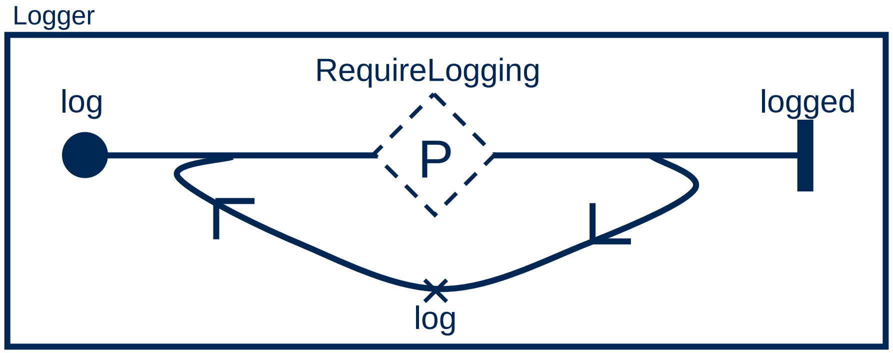
AoView

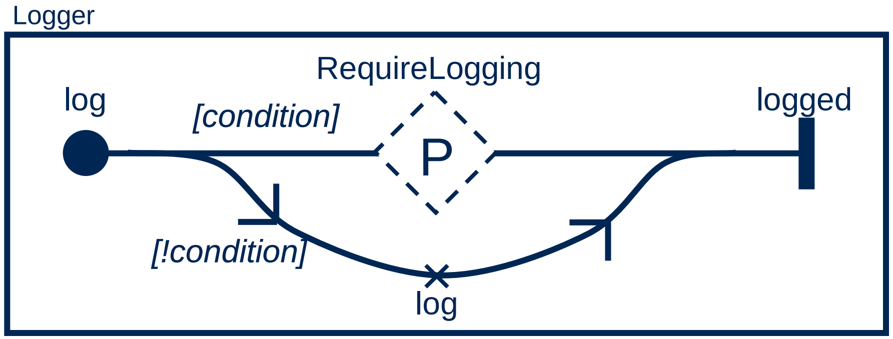
AoURN: Learning by Example (3)

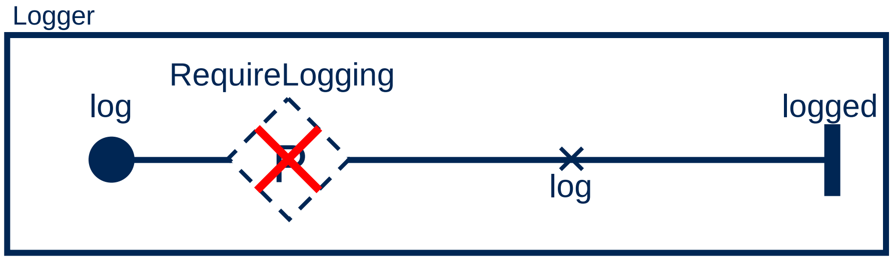
- Composition
  - AoURN can use any composition rule that can be described with the URN notation, not just simple before and after composition rules



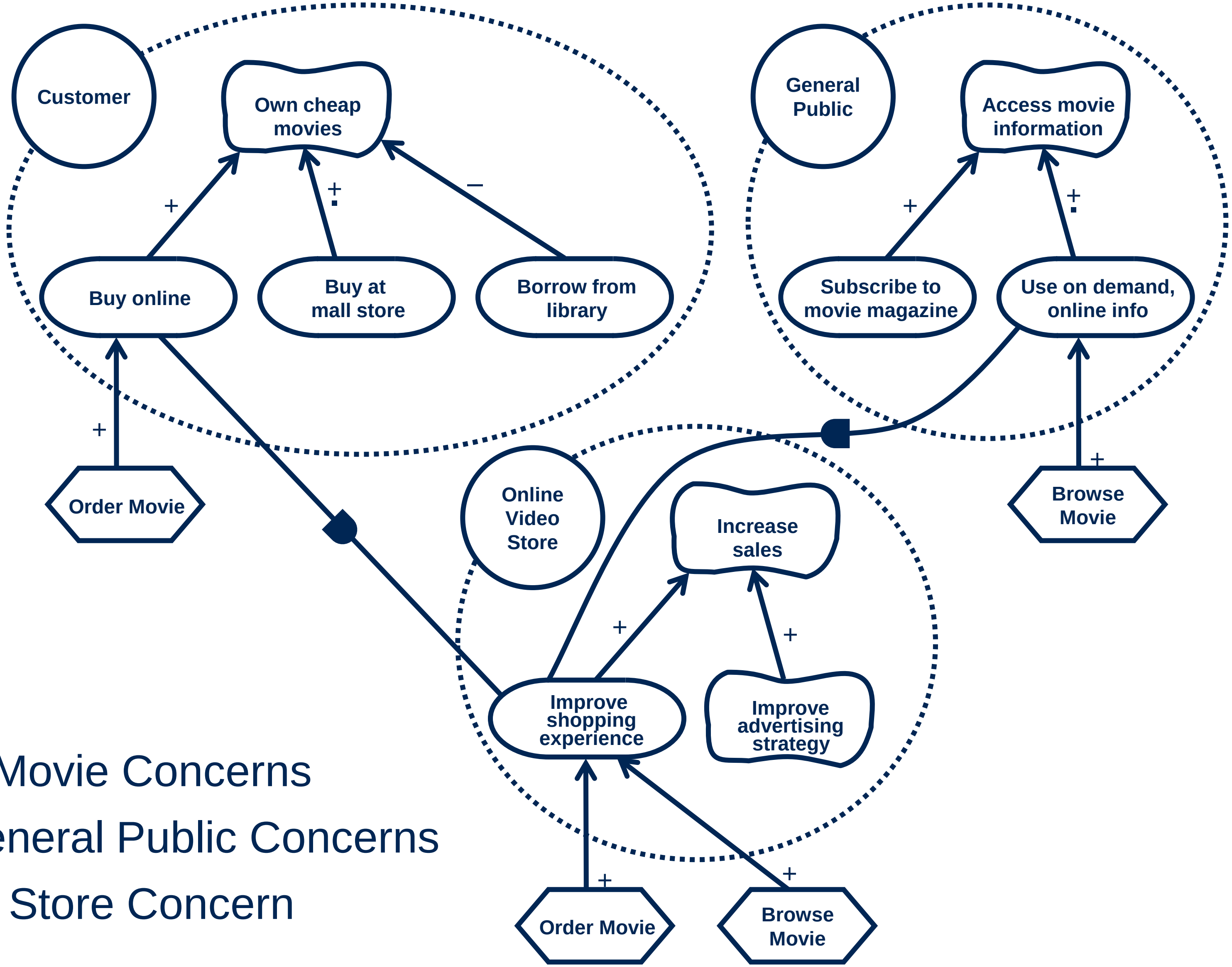








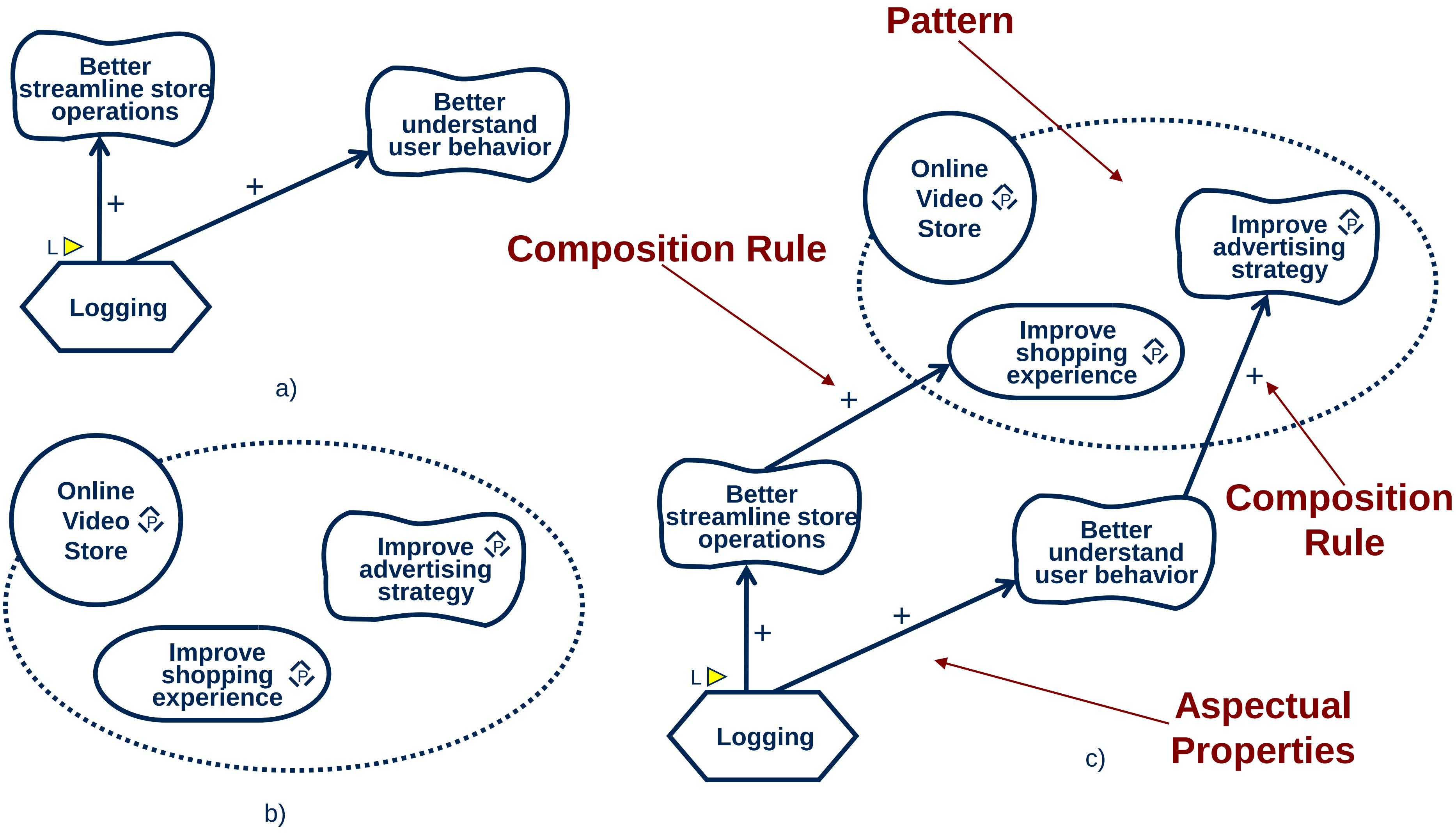
AoURN: Learning by Example (4)



- Buy/Browse Movie Concerns
- Customer/General Public Concerns
- Online Video Store Concern

AoURN: Learning by Example (5)

- Logging Concern (c)



Better streamline store operations

Better understand user behavior

+

+

L

Logging

a)

Online Video Store

Improve advertising strategy

Improve shopping experience

b)

Pattern

Composition Rule

Online Video Store

Improve advertising strategy

Improve shopping experience

+

+

Better streamline store operations

Better understand user behavior

Composition Rule

+

+

L

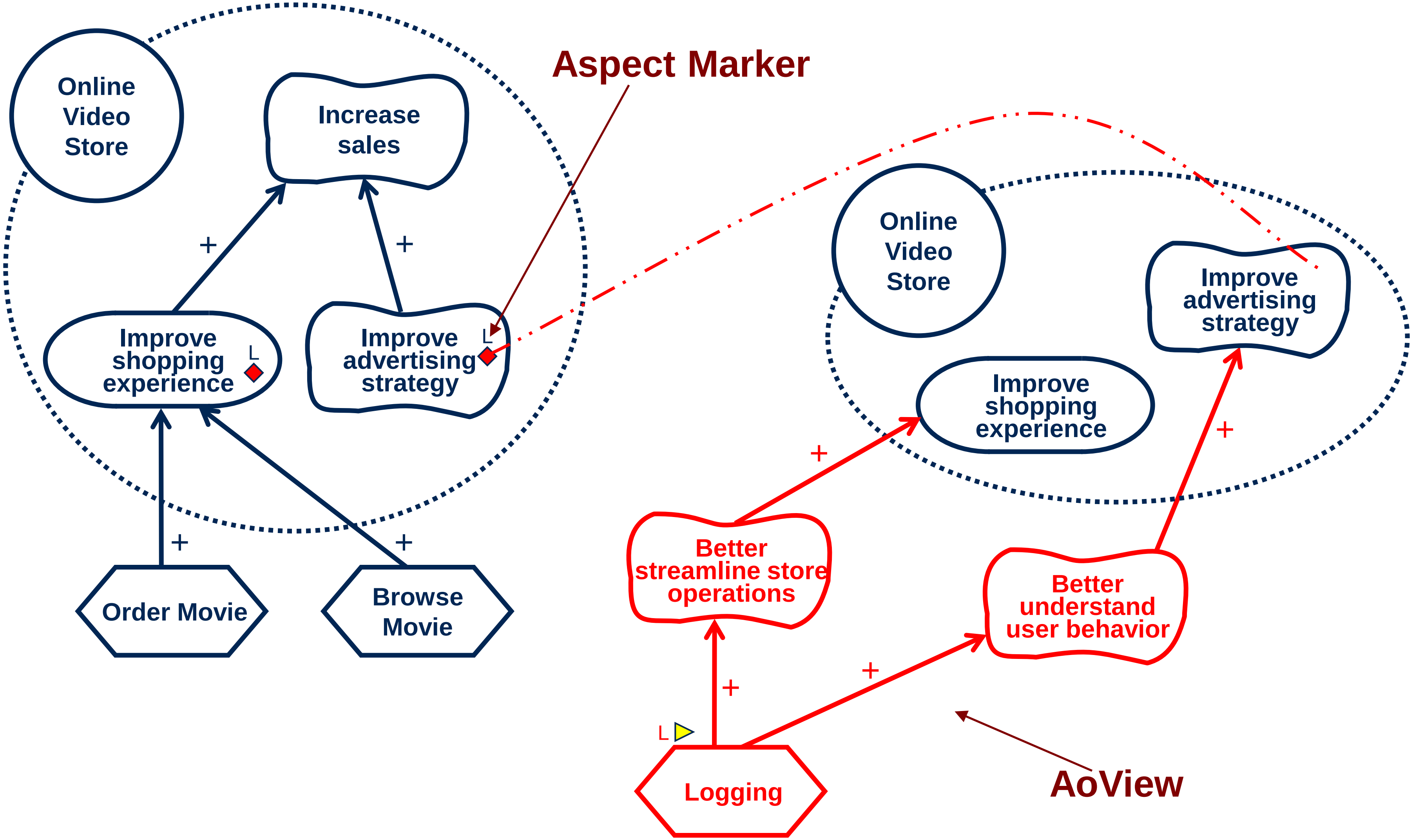
Logging

Aspectual Properties

c)

AoURN: Learning by Example (6)

- Applied Logging Concern

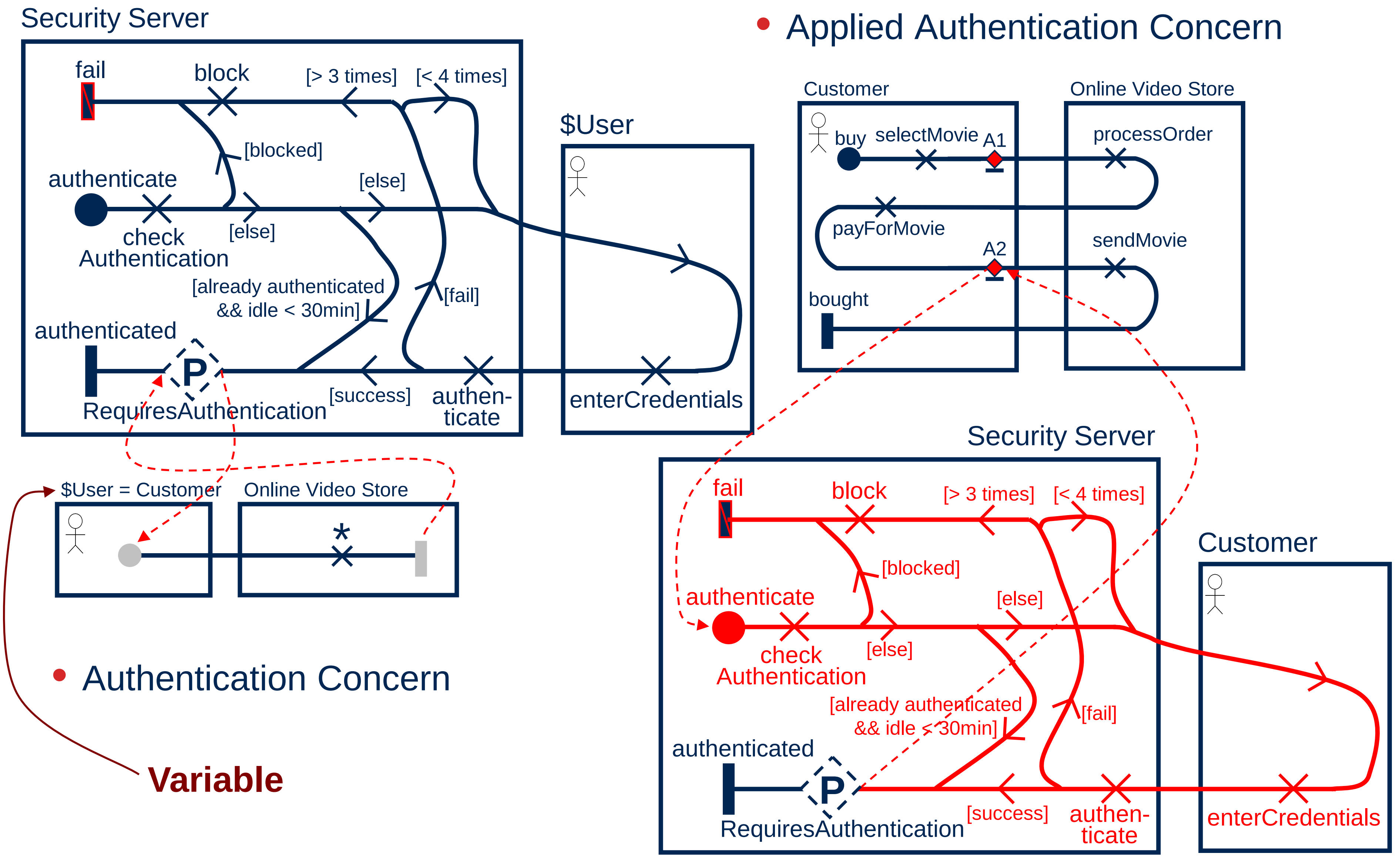

Online Video Store
Increase sales
+
+
Improve shopping experience
L
Improve advertising strategy
L
+
+
Order Movie
Browse Movie
Aspect Marker
Online Video Store
Improve advertising strategy
Improve shopping experience
+
+
Better streamline store operations
Better understand user behavior
+
+
L
Logging
AoView

AoURN: Learning by Example (7)

- Pointcut Map
  - Completely separate from aspect map – contains only pointcut expression
  - Allows for reuse of pointcut expression and aspectual properties
- Pointcut Graph
  - Contains pointcut expressions and some aspectual properties to specify the composition rule
  - Harder to reuse pointcut expression and aspectual properties
  - More inspired by graph transformation-based approaches such as MATA
  - Due to the nature of GRL models – highly interconnected

AoURN: Learning by Example (8)

- Applied Authentication Concern

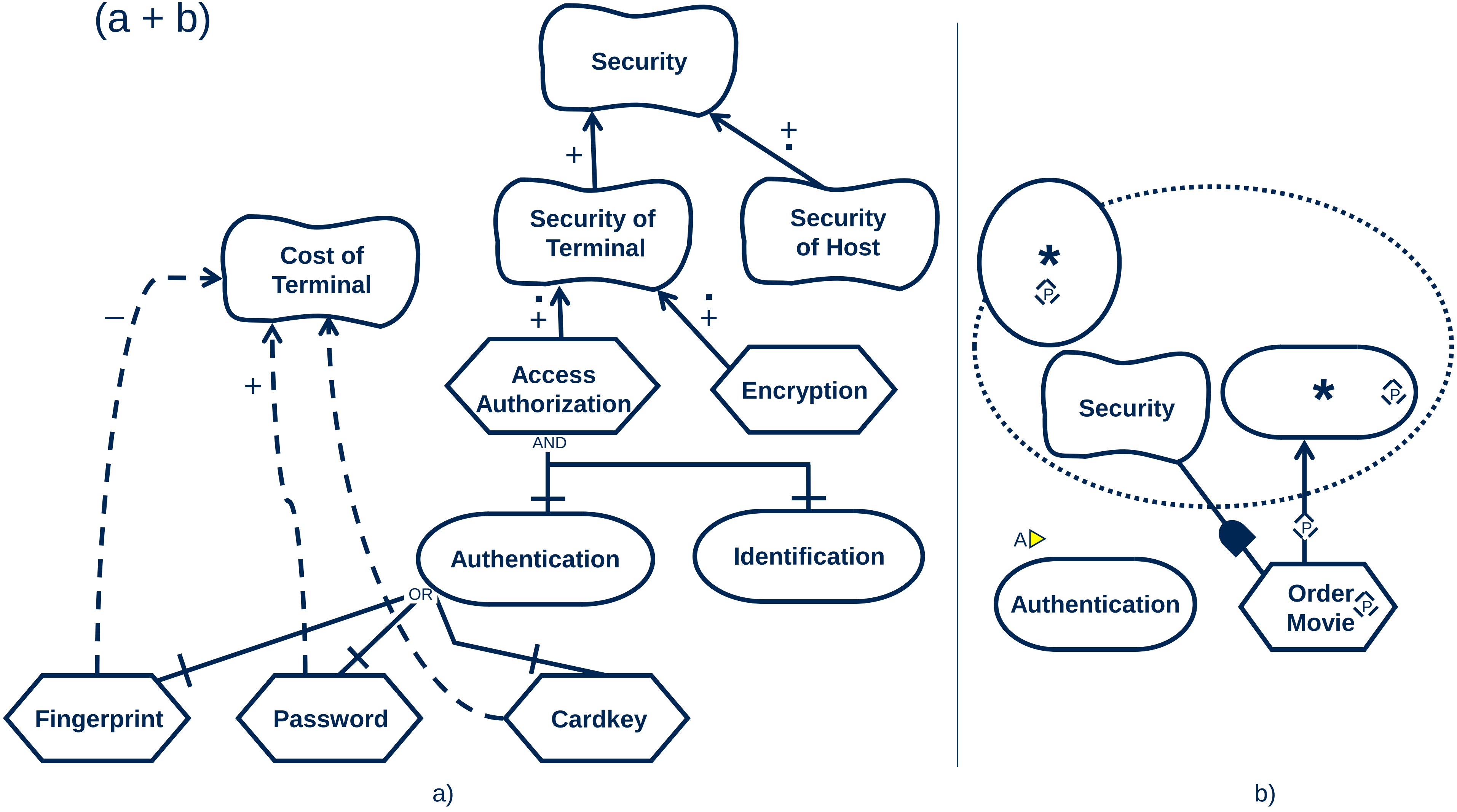


- Authentication Concern

Variable

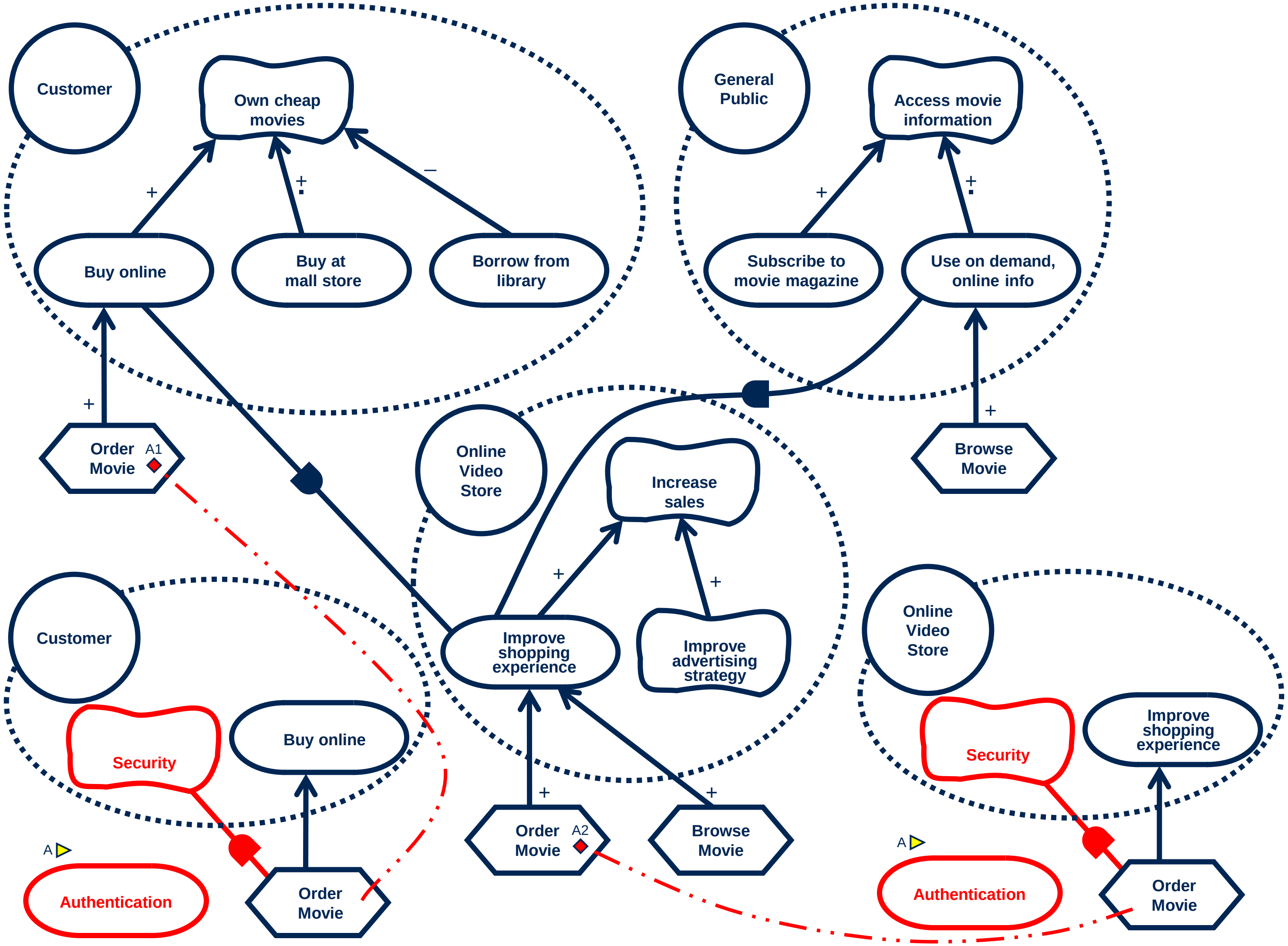
AoURN: Learning by Example (9)

- Authentication Concern (a + b)



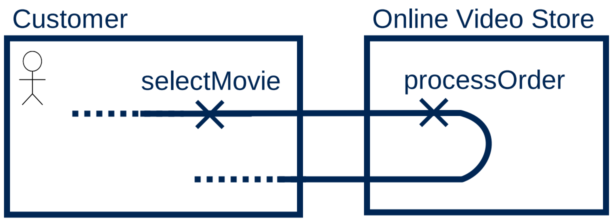
AoURN: Learning by Example (10)

- Applied Authentication Concern

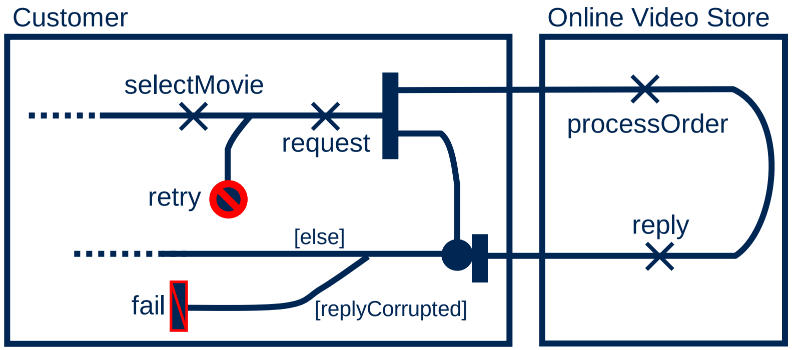


AoURN: Learning by Example (11)

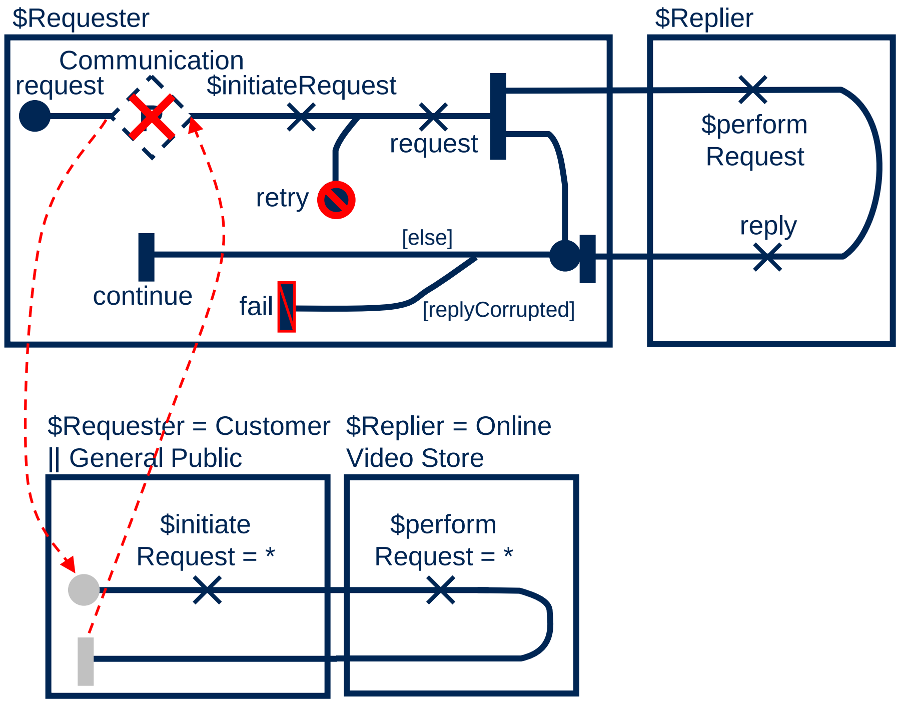
- Communication Concern (c)



a)

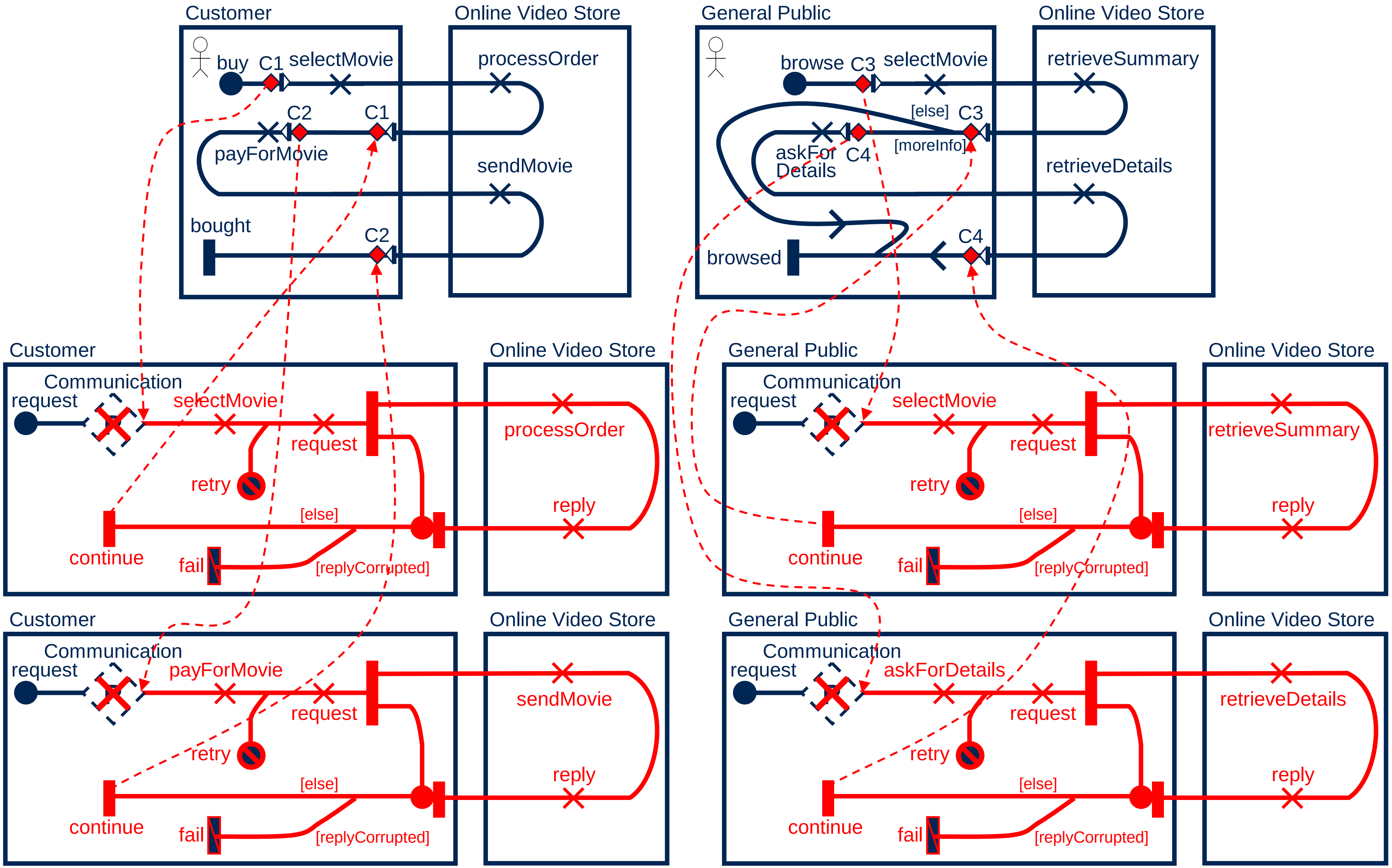


b)

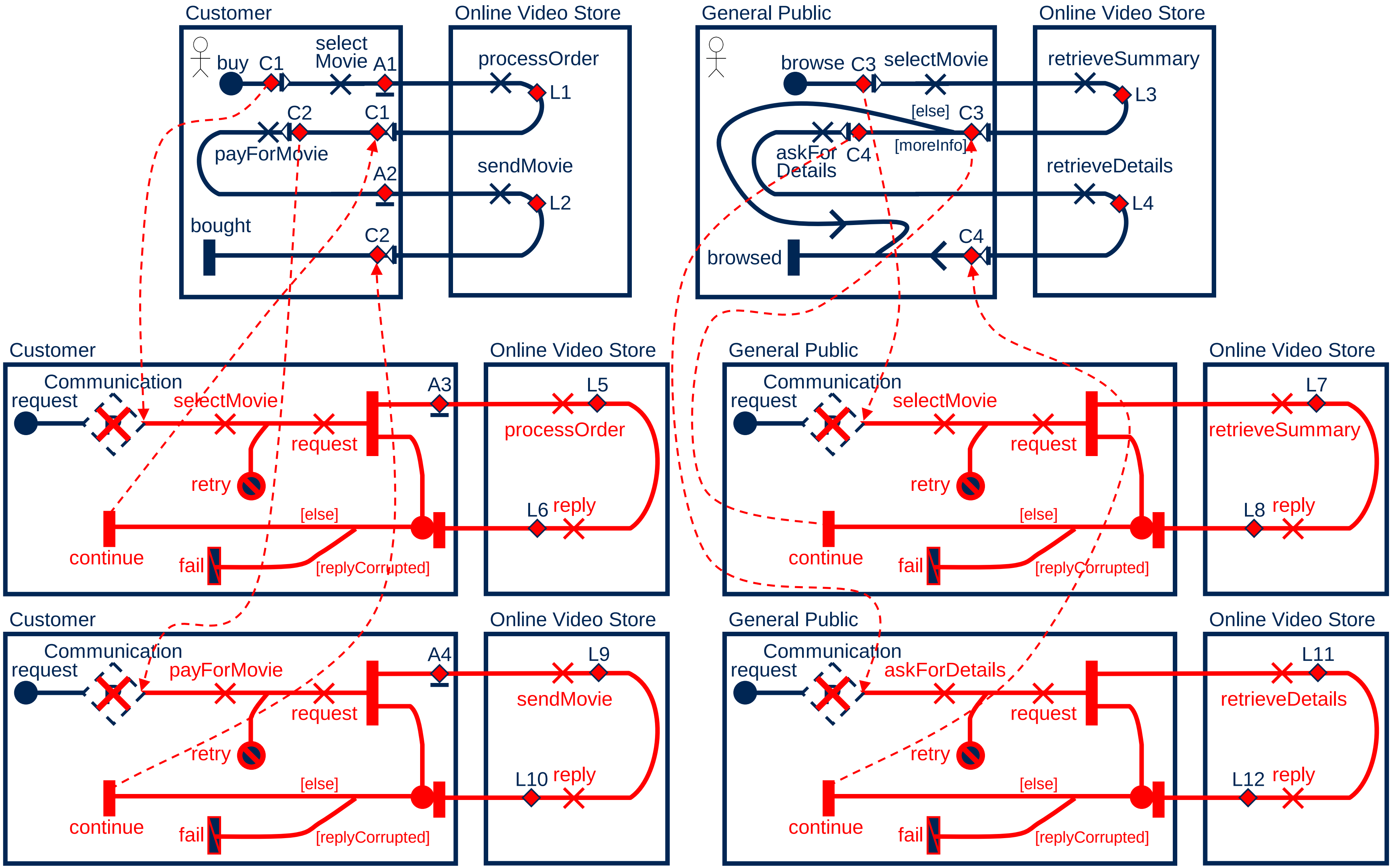


c)

AoURN: Learning by Example (12)

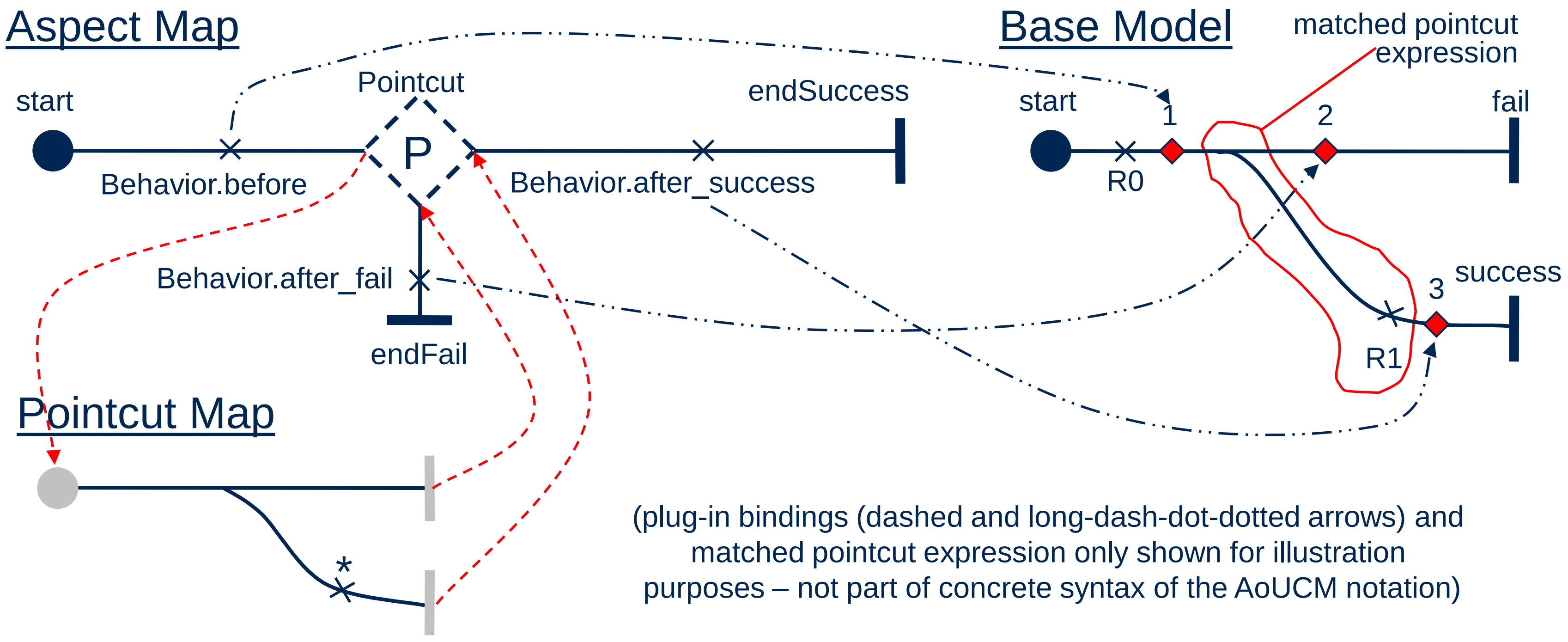


AoURN: Learning by Example (13)



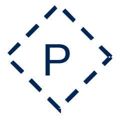
Aspect-oriented Use Case Maps: In a Nutshell (1)

- An aspect defines its structure/behavior (= aspect map) and a pattern called pointcut expression (= pointcut map) for its composition rule stating where the aspect is to be applied in a model



(plug-in bindings (dashed and long-dash-dot-dotted arrows) and matched pointcut expression only shown for illustration purposes – not part of concrete syntax of the AoUCM notation)

Pointcut stub:

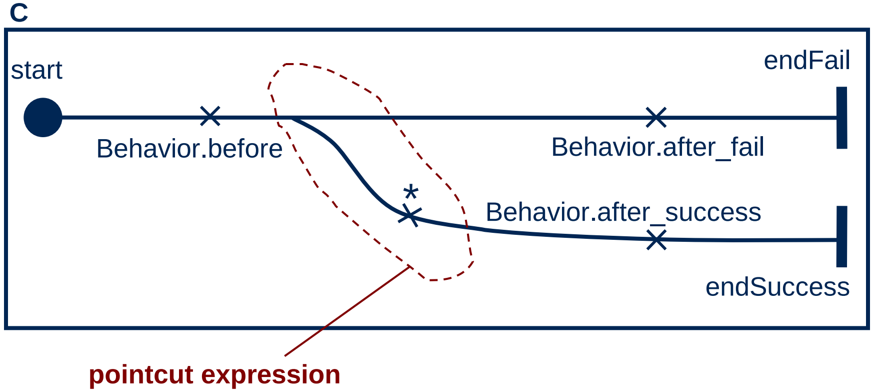


Aspect marker: ◆

Aspect-oriented Use Case Maps: In a Nutshell (2)

- All-in-One style not used because it hinders separate reuse of aspectual properties and pointcut expressions
- All-in-One style leads to duplication if the same aspect needs to be applied at various locations that cannot be described with one pattern

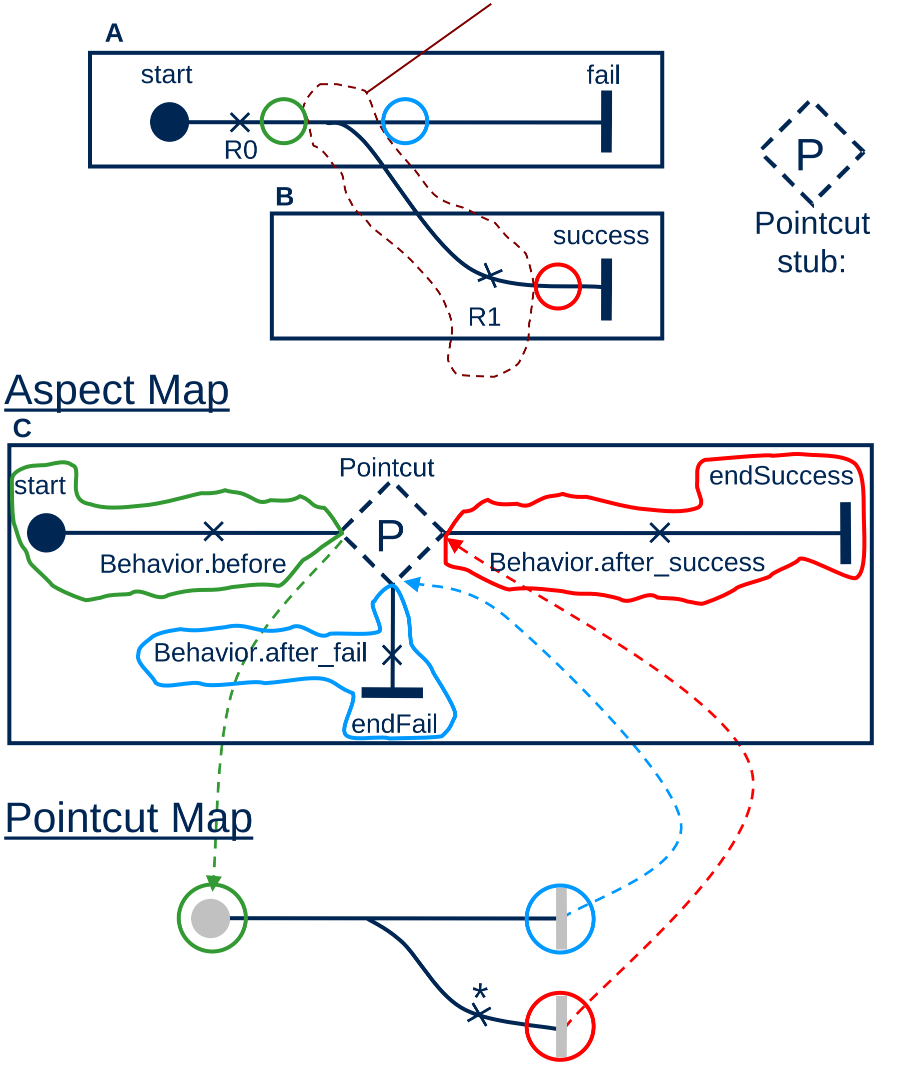
All-in-One Style - Aspect/Pointcut Map



Base Model

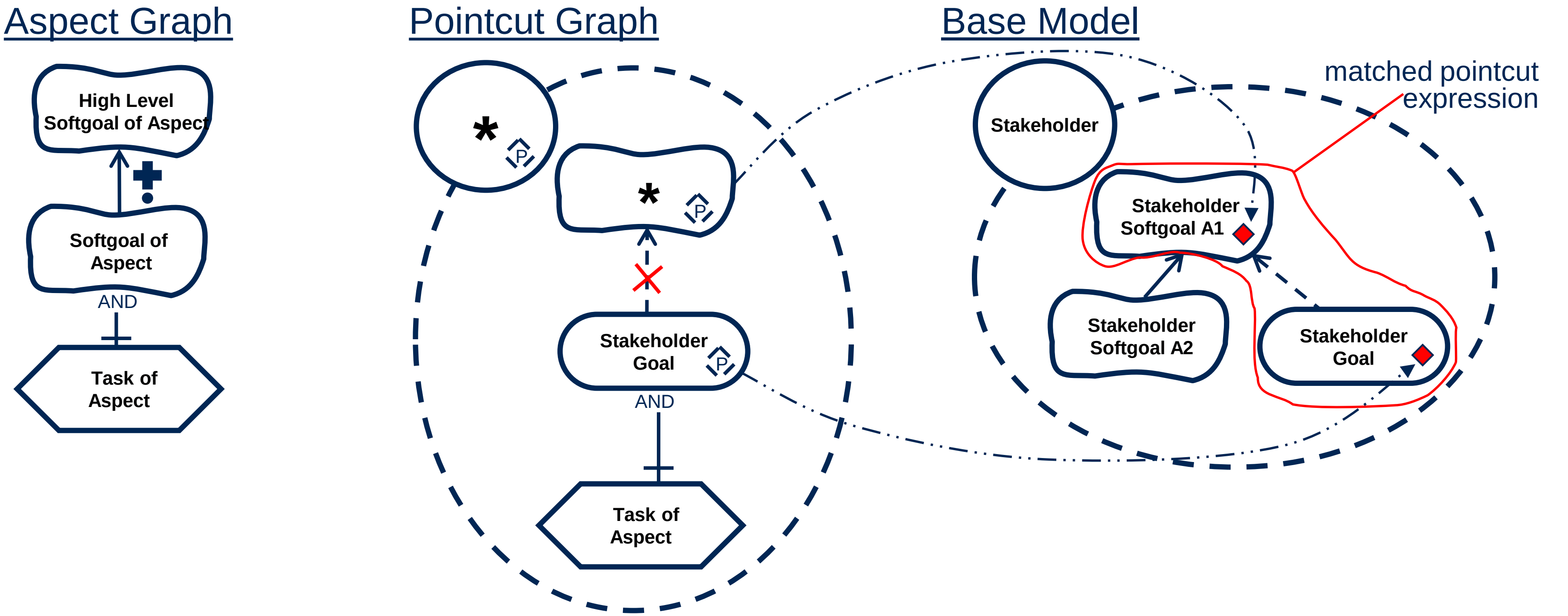
Aspect Map

Pointcut Map



Aspect-oriented GRL: In a Nutshell

- An aspect defines its structure/behavior (= aspect graph + non-marked pointcut graph) and a pattern called pointcut expression (= marked pointcut graph) for its composition rule stating where the aspect is to be applied in a model



Pointcut marker: P

Pointcut deletion marker: ✕

Aspect marker: ◆

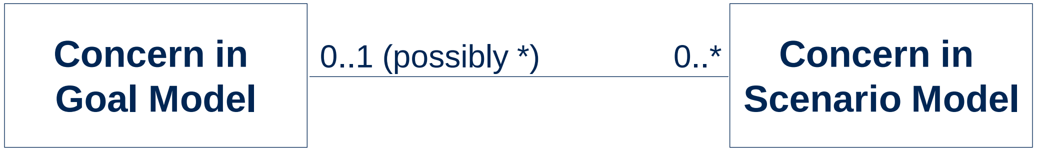
(mapping of pointcut expression to base model (long-dash-dot-dotted arrows) and matched pointcut expression only shown for illustration purposes – not part of AoGRL notation)

Aspect-oriented URN: Overview (1)

- AoURN (Aspect-oriented User Requirements Notation)
  - Unifies goal-oriented, scenario-based, and aspect-oriented concepts in a scalable framework
  - Extends the abstract syntax, the concrete syntax, and the semantics of URN with aspect-oriented concepts
  - Requires almost no changes to the familiar URN notation (syntax remains virtually the same but the existing semantics are clarified and extended)
- AoURN models each use case, each stakeholder’s goals, and each NFR as a concern
  - NFRs fundamentally are aspectual (i.e., crosscutting) in nature
  - Most use cases are not aspectual but are peers
  - Most stakeholder goals are not aspectual but are peers
- AoURN does not differentiate between concerns and aspects and hence follows a more symmetrical, multi-dimensional approach to aspect-oriented modeling

Aspect-oriented URN: Overview (2)

- Features of AoURN
  - Crosscutting concerns (including pointcut expressions) are fully described in a graphical way
  - Exhaustive composition of crosscutting concerns is only limited by the expressive power of URN itself (as opposed to a particular pointcut or composition language)
  - Aspectual properties and pointcut expressions are defined separately
- AoURN defines for each concern at least a goal model or a scenario model (or both)
  - Behavioral/structural dimensions of a concern are modeled with AoUCM
  - Reasons for a concern are modeled with AoGRL
  - Concerns in GRL models can be traced to concerns in UCM models


Concern in Goal Model — 0..1 (possibly *) — 0..* — **Concern in Scenario Model**

Carleton UNIVERSITY

McGill

uOttawa
L'Université canadienne
Canada's university

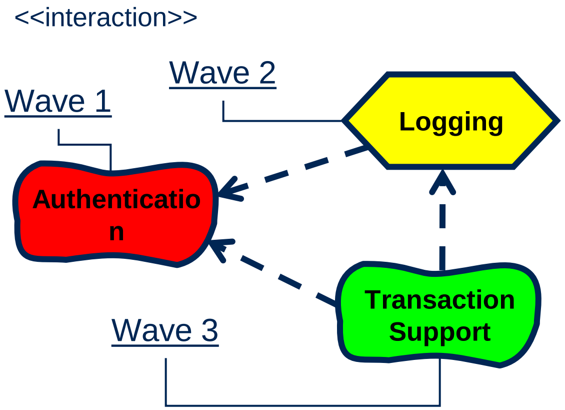
Advanced Features of AoURN

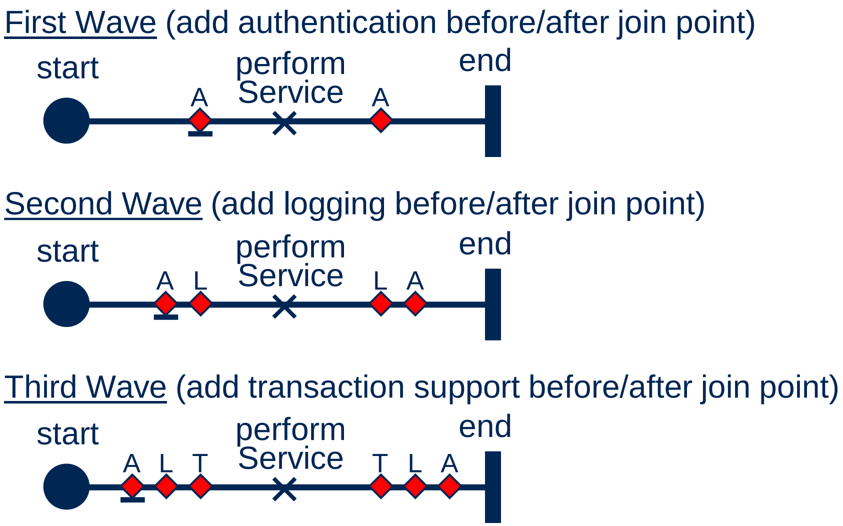
Anything Pointcut Element

- Indicates in the pattern that a series of model elements may be matched

Concern Interactions

- Concerns may interact with each other in undesired ways
- In AoURN, a Concern Interaction Graph (CIG) specifies precedence rules that decide which concerns are applied first
- A CIG is a spezialized goal model tagged with <<interaction>>
- Waves



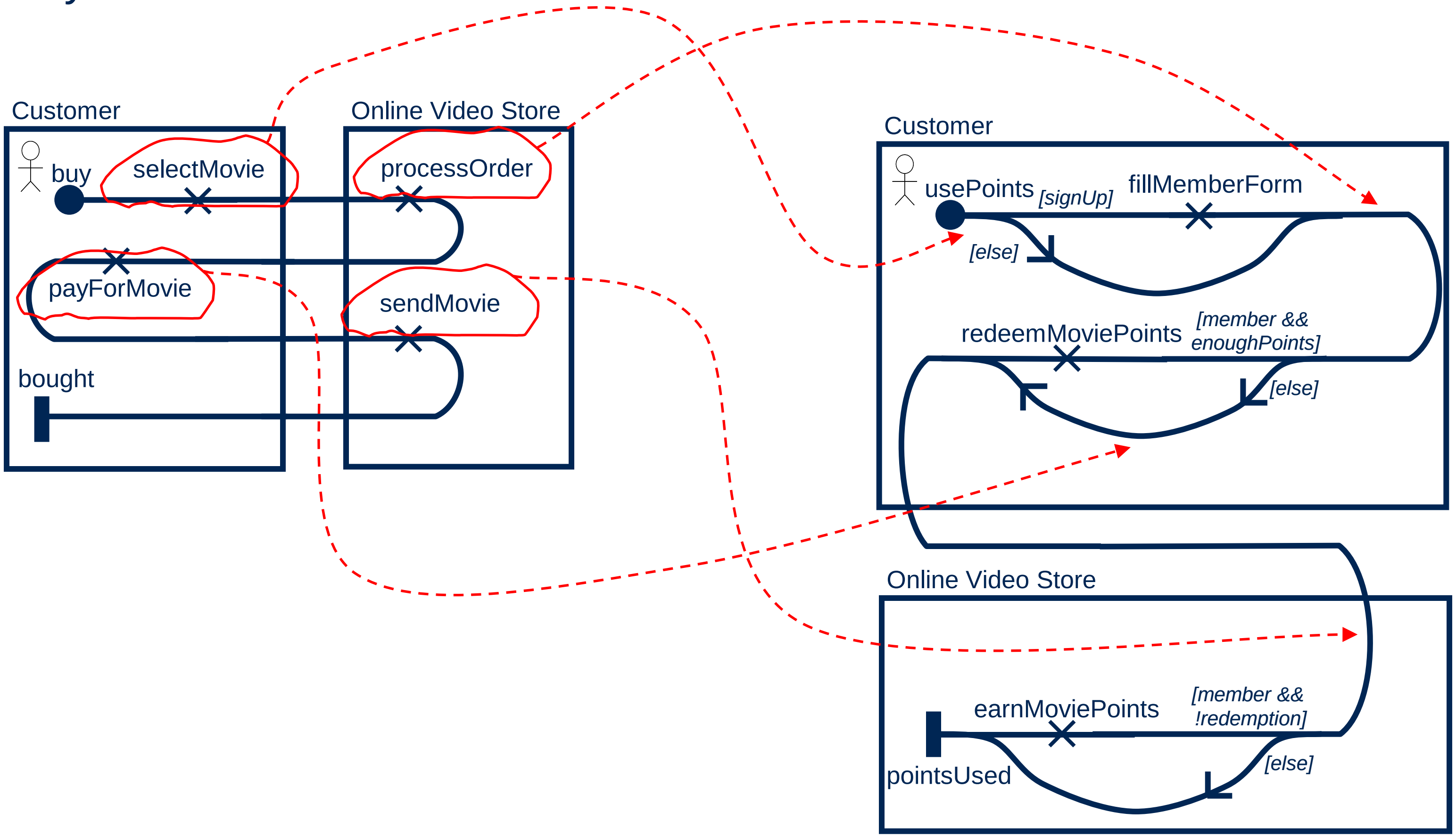

First Wave (add authentication before/after join point)

Second Wave (add logging before/after join point)

Third Wave (add transaction support before/after join point)

Interleaved Composition

- Buy Movie Concern
- Movie Points Concern

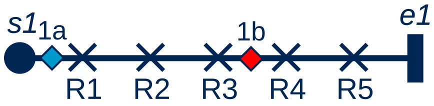


Semantics-Based Composition

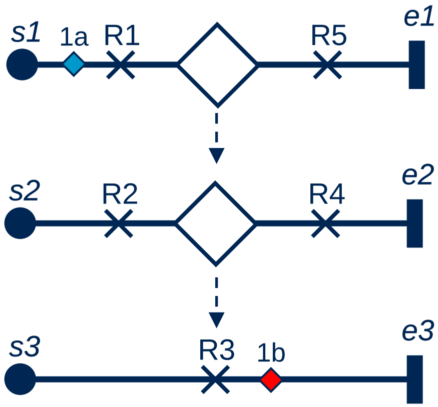
- ”Whitespace” in the base model is not matched
  - E.g., direction arrow, OR-join…

- Stubs are interpreted as their flattened equivalent

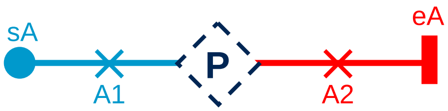
Scenario One



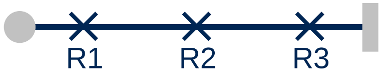
Scenario Two



Aspect Map / AoView



Pointcut Map



Conclusion and References

Conclusion (1)

- The User Requirements Notation (URN) is an ITU-T standard
- Modeling with the Goal-oriented Requirement Language (GRL)
  - Focuses on answering "why" questions
  - Intentions, functional / non-functional requirements, rationales
- Modeling with Use Case Maps (UCM)
  - Focuses on answering "what" questions
  - Scenarios, services, architectures

- While modeling with URN as a whole, goals are operationalized into tasks, and tasks are elaborated in (mapped to) UCM path elements and scenarios
  - Moving towards answering "how" questions
  - Can guide the selection of an architecture or scenarios

- Enables the elicitation/specification of systems, standards, and products, their analysis from various angles, and transformations

Conclusion (2)

- The Aspect-oriented User Requirements Notation (AoURN) combines goal-modeling, scenario-modeling, and aspect-oriented modeling at the requirements level in one framework

- Graphical and familiar
- Better encapsulation of concerns in URN models (goals and scenarios)
- URN models are more easily maintained and reused
- A standardized way of modeling non-functional requirements and use cases
- Enhanced matching mechanism based on semantics

- Tool support for AoURN is available in the jUCMNav tool

References

General

- URN Virtual Library (~350 entries), http://www.UseCaseMaps.org/pub/
- URN tool: jUCMNav, University of Ottawa, http://jucmnav.softwareengineering.ca/jucmnav/
- ITU-T, Recommendation Z.151 (11/08): User Requirements Notation (URN) - Language Definition, Geneva, Switzerland, approved November 2008.
- URN website: http://www.usecasemaps.org/urn

Overview of URN

- Mussbacher, G.: “Aspect-oriented User Requirements Notation”. PhD thesis, SITE, University of Ottawa, Canada, 2010.
- Mussbacher, G., Amyot, D., Araújo, J., and Moreira, A.: “Requirements Modeling with the Aspect-oriented User Requirements Notation (AoURN): A Case Study”. Katz, S., Mezini, M., and Kienzle, J. (Editors), TAOSD VII, Springer, LNCS 6210, pp. 23–68 (2010)
- Mussbacher, G., Amyot, D., and Whittle, J.: “Refactoring-Safe Modeling of Aspect-Oriented Scenarios”. ACM/IEEE 12th International Conference on Model Driven Engineering Languages and Systems (MODELS 2009), Denver, USA, Springer, LNCS 5795, pp. 286–300 (October 2009)

Appendix: Notation Overview – AoURN

Concern Interaction Graph:
models conflicts and dependencies between concerns

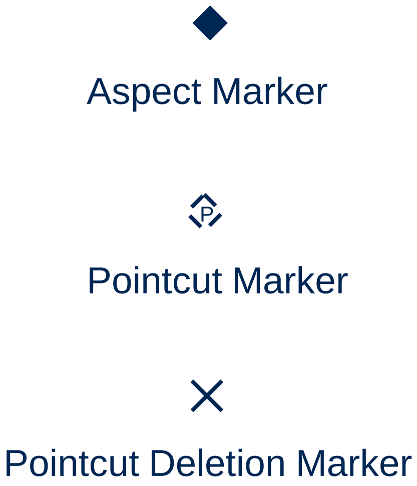
Aspect Graph:
models aspectual properties

Pointcut Graph:
models pointcut expression and composition rule

Aspect Map:
models aspectual properties and composition rule

Pointcut Map:
models pointcut expression

(a) AoURN Diagrams

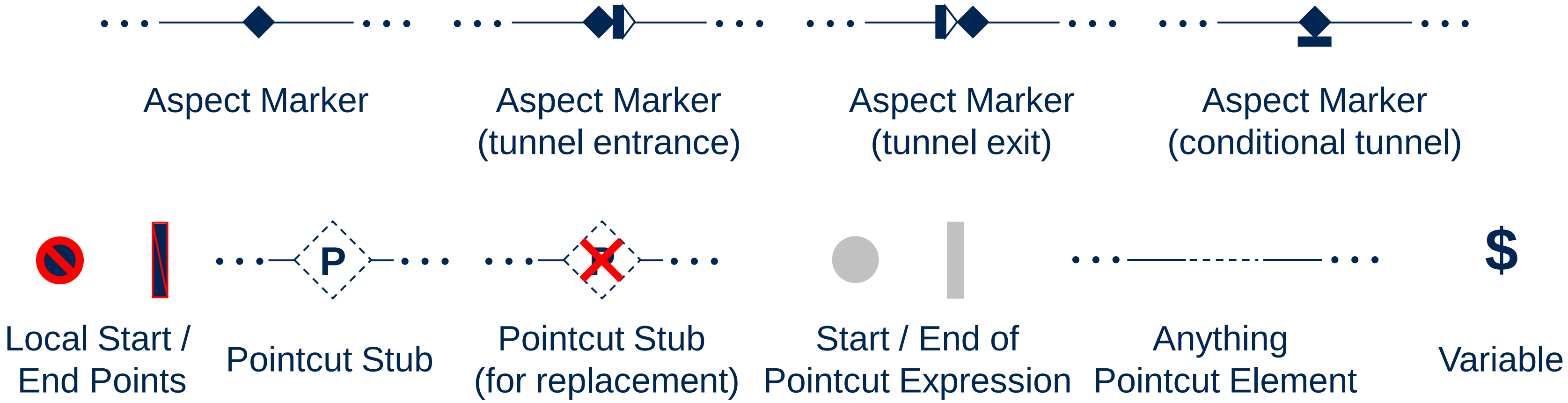


Aspect Marker

Pointcut Marker

Pointcut Deletion Marker

(b) AoGRL Elements



Aspect Marker

Aspect Marker (tunnel entrance)

Aspect Marker (tunnel exit)

Aspect Marker (conditional tunnel)

Local Start / End Points

Pointcut Stub

Pointcut Stub (for replacement)

Start / End of Pointcut Expression

Anything Pointcut Element

Variable

(c) AoUCM Elements