From Use-Cases to Test-Cases via Meta-Model-based Reasoning

Stefan Gruner
University of Pretoria
sg@cs.up.ac.za

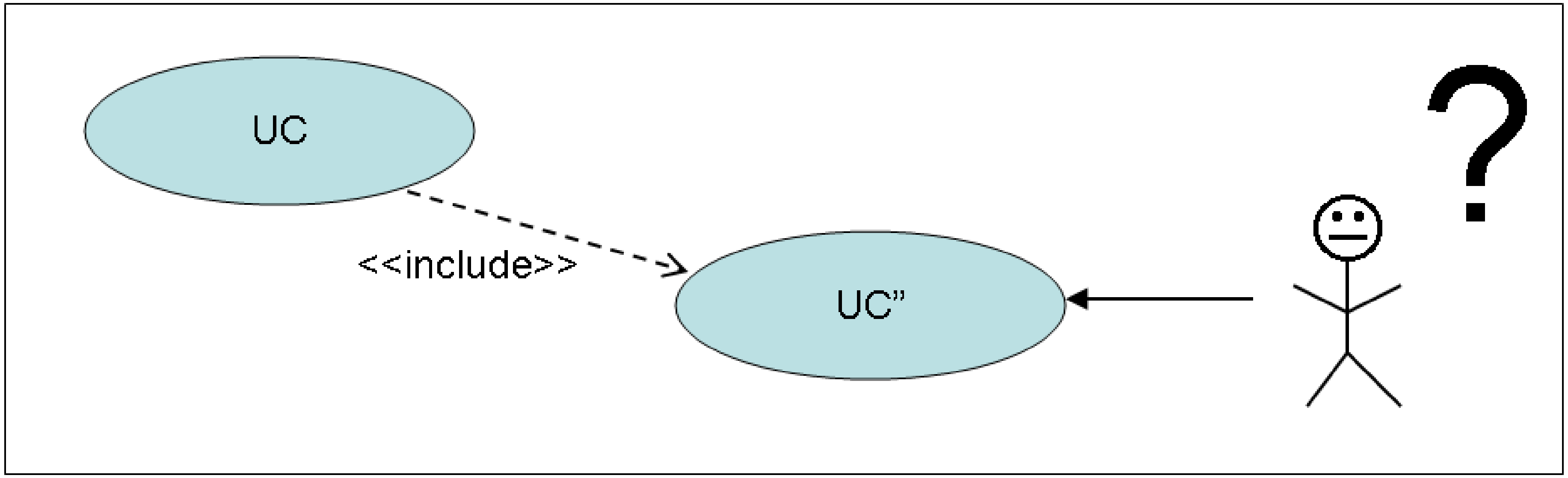
Sponsorship by Formal Methods Europe, and the University of Pretoria, is acknowledged

Abstract

- In *Use Cases Considered Harmful*, **Simons** analysed the logical weaknesses of the UML Use Case notation and has recommended to **“fix the faulty notion of dependency”**. The project sketched in this position paper is inspired by Simons' critique
- The main contribution of this paper is a detailed meta model of possible relations between Use Cases. Later in the project this meta model is then to be formalized in a natural deduction calculus which can be implemented in *Prolog*
- As a result of formalization, Use Case specifications can be queried for **inconsistencies** as well as for **test cases** which must be observable after a software system is implemented based on such a Use Case specification
- Software tool support for this idea is under development

Introduction: What does this mean?

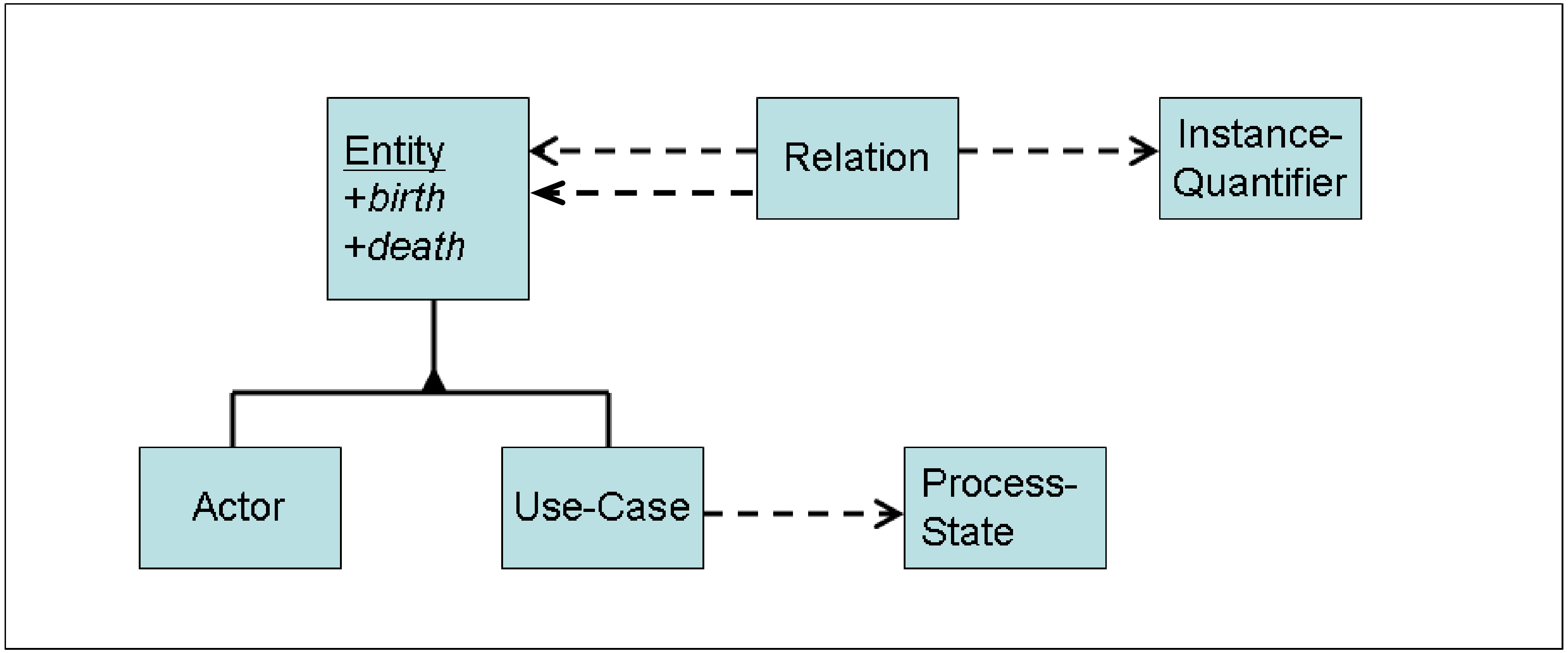
- This is a syntactically legal UML Use Case Specification √
- But how is a System supposed to behave, according to this Specification?



Meta Model (Taxonomy)

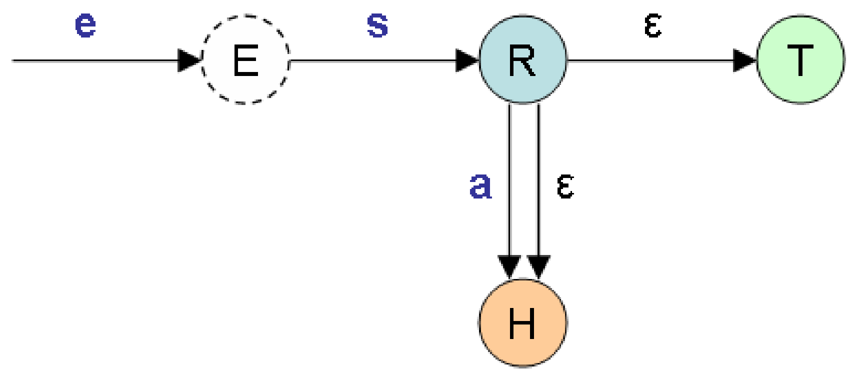
- Formal Reasoning about relations in a UC specification will be based on a Taxonomy (Meta Model)
- A **Meta Model for Use Cases**, which is considerably more comprehensive than the “official” UC description, is the main contribution of this paper.
- The associated reasoning-engine is under development (ongoing stud.-project work)

Top Layer of the Meta Model



We will clearly distinguish Use Cases from their Instances, in analogy to the distinction between Classes and Objects in OOP. Use Cases Instances, at runtime, are Processes!

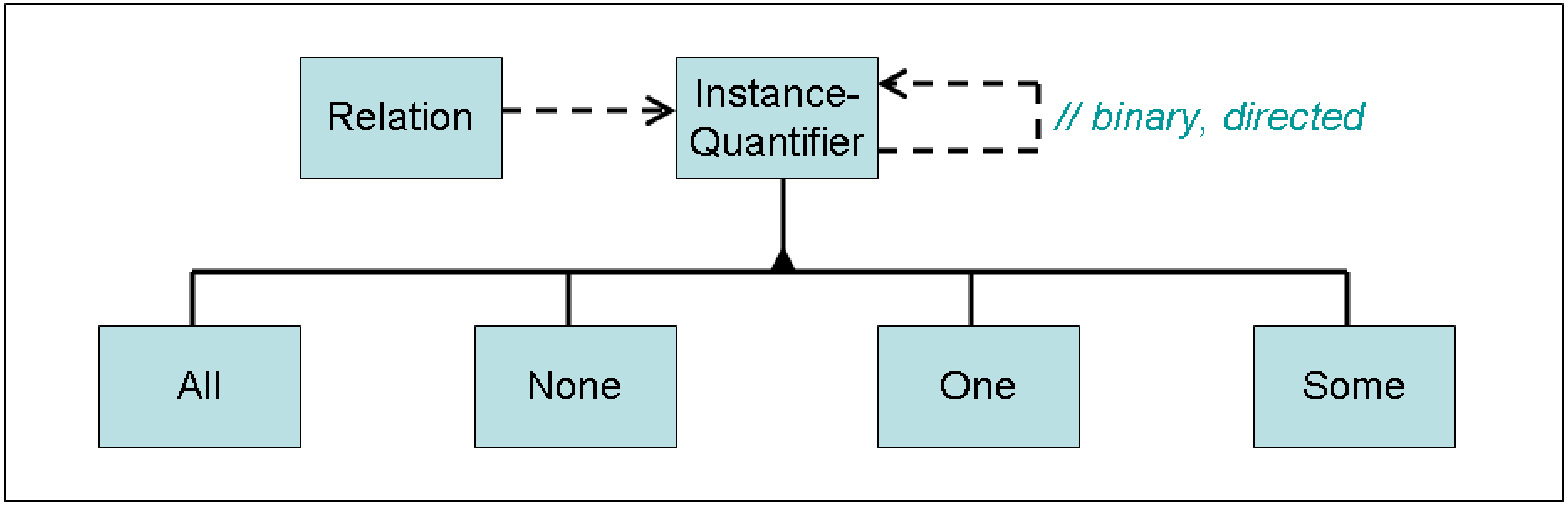
States of a Use Case Instance


$$\begin{tikzcd} \arrow[r, "e"] & E \arrow[r, "s"] & R \arrow[r, "\varepsilon"] \arrow[d, shift right, "a"'] \arrow[d, shift left, "\varepsilon"] & T \\ & & H & \end{tikzcd}$$

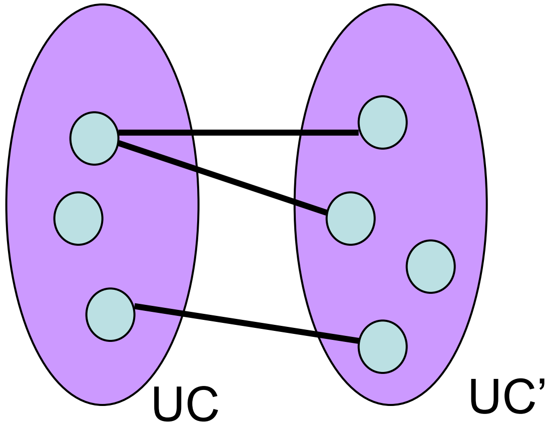
E: state of virtual existence
R: running
T: terminated successfully
H: halted un-successfully
e: enabled by pre-condition
s: started by actor
a: aborted by actor
ε: internal epsilon-transition

We assume eventual termination of each individual UC instance. Infinite life of a reactive system is still possible, namely though a non-limited chain of creation (of limited instances) during the life-time of the system as a whole.

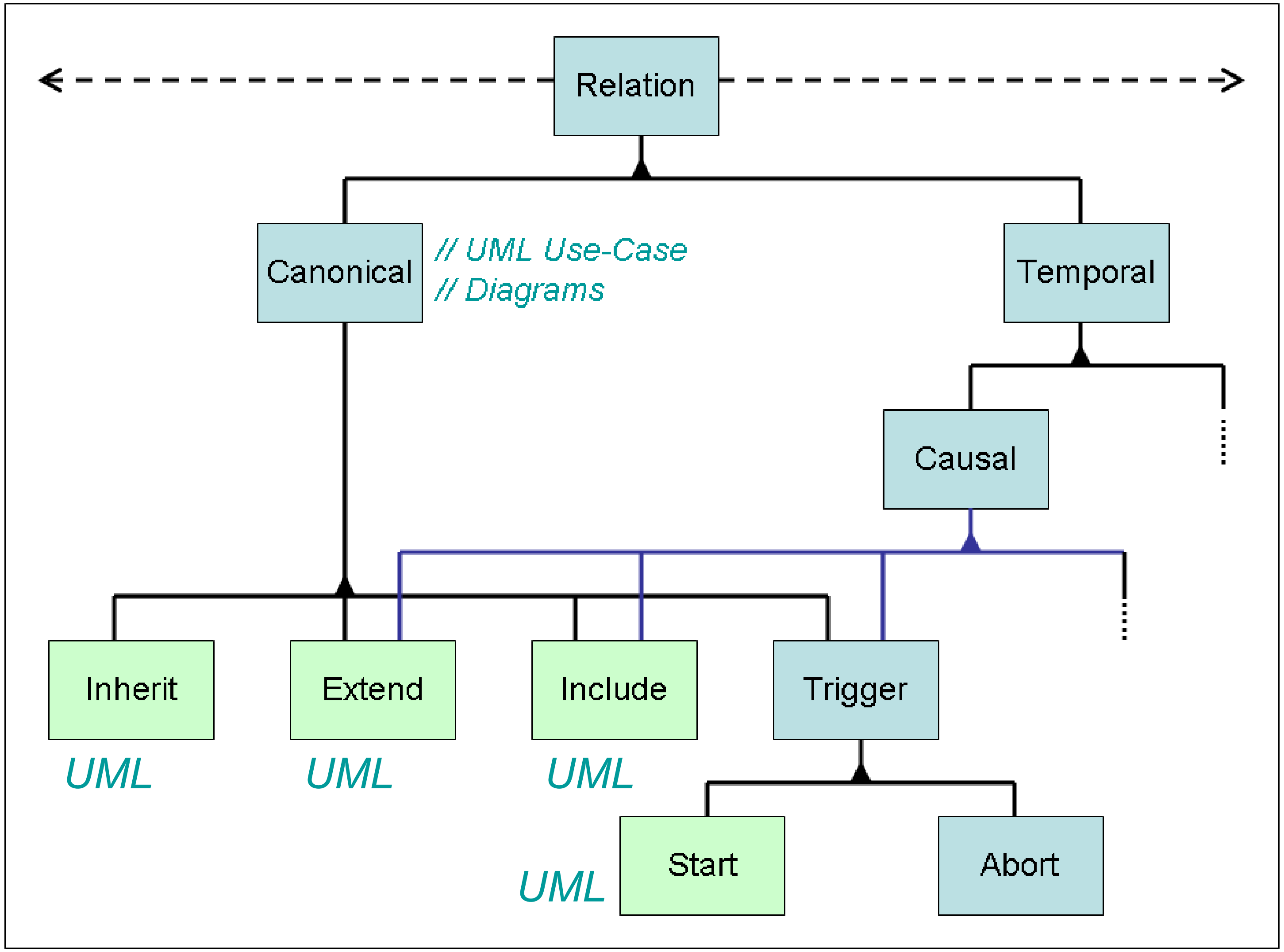
Lower Level of Meta Model



As soon as a conceptual distinction between Use Cases and their runtime instances is made, one relationship of two UC, at specification level, is multiplied at the level of instances.
This motivates the Introduction of the usual Quantifiers to further specify a relationship type at Instance Levels

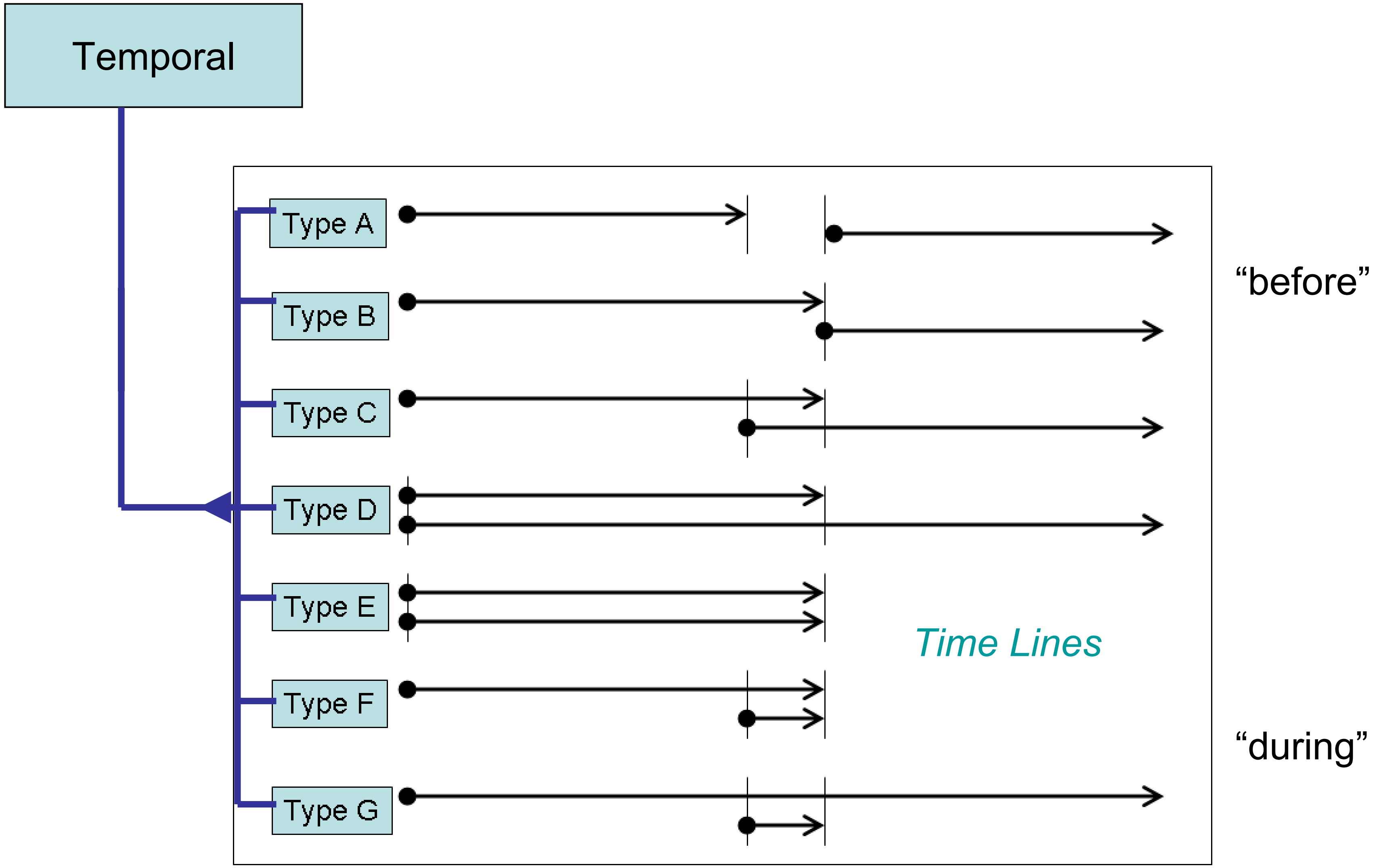


Lower Level of Meta Model

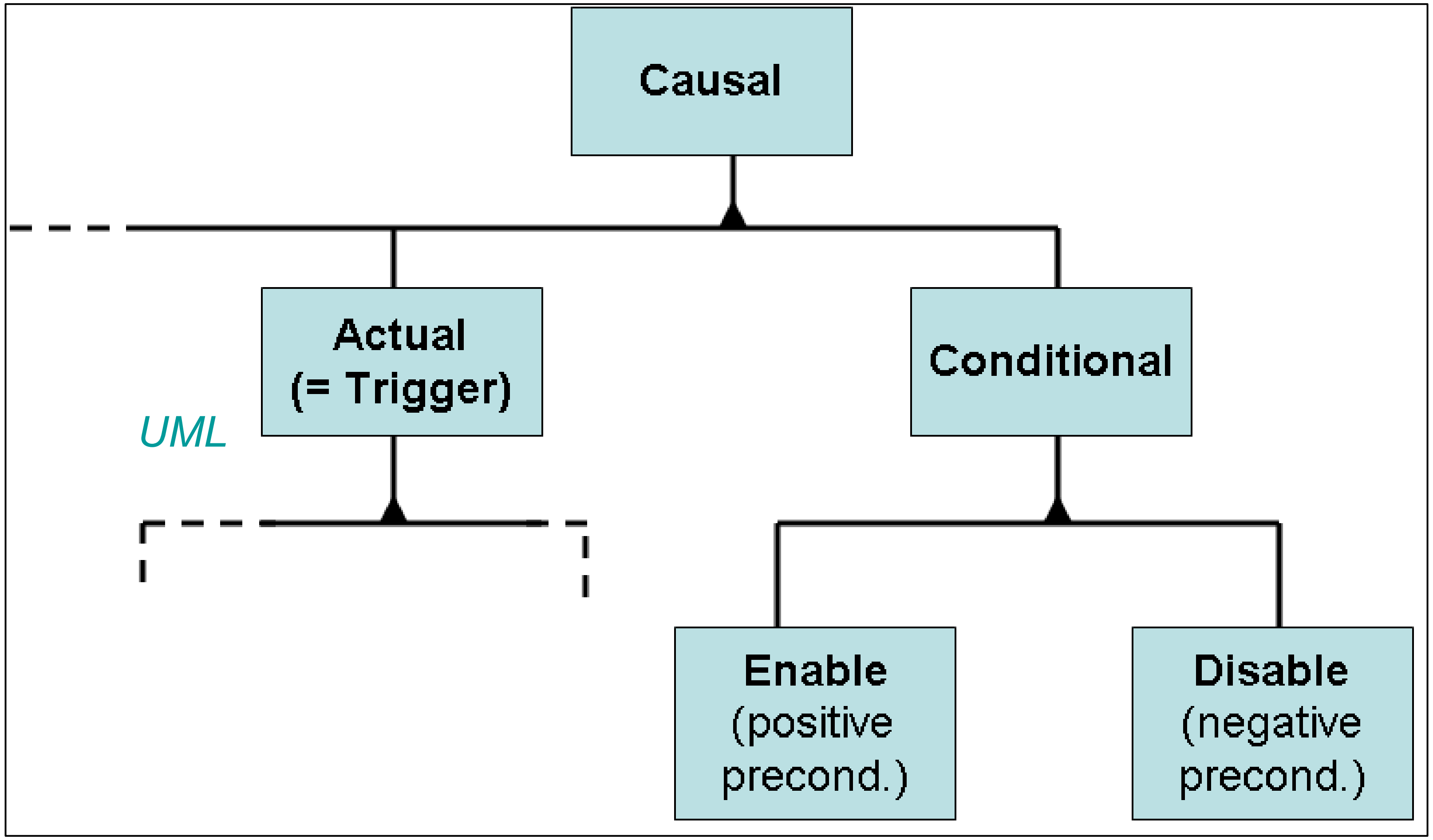


Note the additions to the canonical features of UML

Types of Temporal Relations



Types of Causal Relations

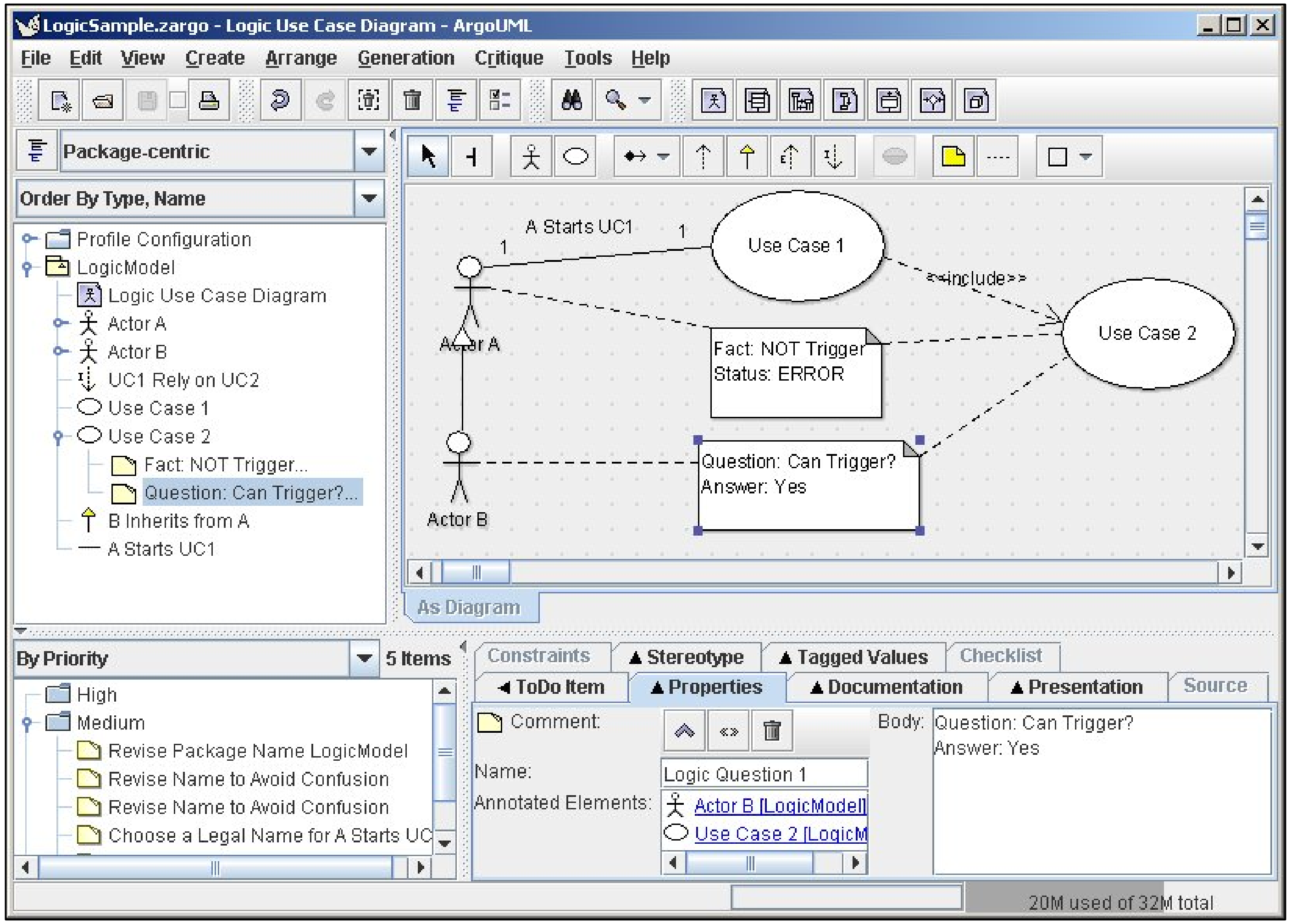


Conditional Relations were so far *informally* annotated to UC Specifications

Ongoing Work (Method)

- Re-Use existing UML Editor to create Logic-enhanced UC specifications: ➔
- Editor creates textual representation in an XML-like notation: ➔
- Translation into PROLOG-Facts, either via some Description-Logic, or directly (e.g. http://www.fraber.de/sitec/dl.html)
- Specification of PROLOG-Rules which represent the consistency conditions to which any UC specification must obey.

UC Editor (based on *ArgoUML*)



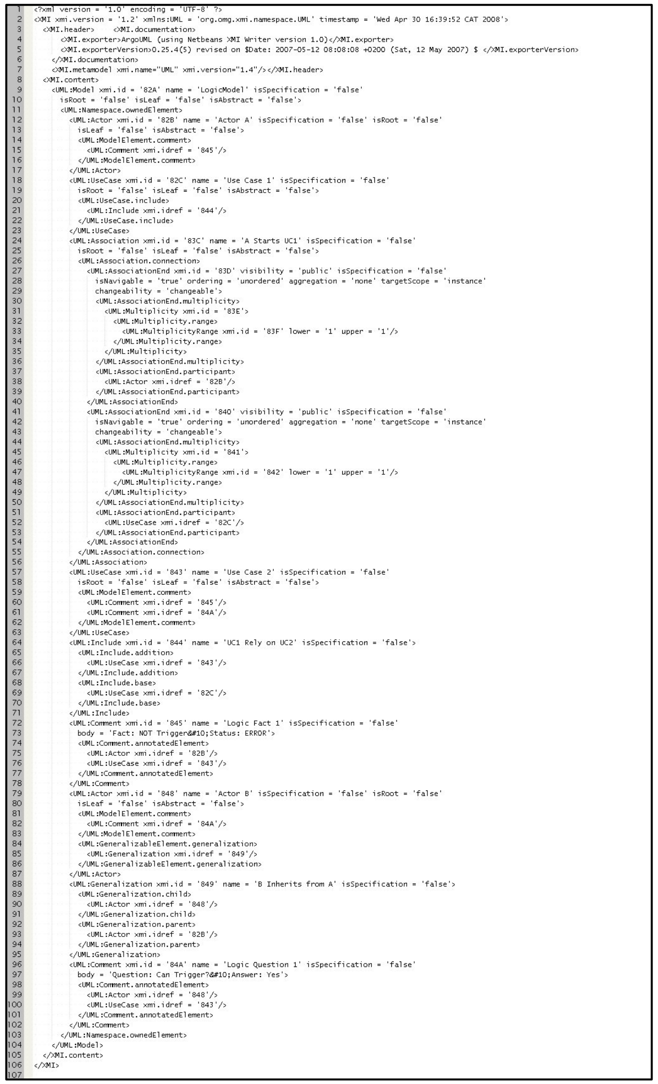
Name: Logic Question 1

Annotated Elements: Actor B [LogicModel], Use Case 2 [LogicM

Body: Question: Can Trigger?
Answer: Yes

User can specify also the non-UML relations from our new Meta Model

XML-like Representation of UC Specification



```xml
<?xml version = '1.0' encoding = 'UTF-8' ?>
<XMI xmi.version = '1.2' xmlns:UML = 'org.omg.xmi.namespace.UML' timestamp = 'Wed Apr 30 16:39:52 CAT 2008'>
  <XMI.header>    <XMI.documentation>
      <XMI.exporter>ArgoUML (using Netbeans XMI Writer version 1.0)</XMI.exporter>
      <XMI.exporterVersion>0.25.4(5) revised on $Date: 2007-05-12 08:08:08 +0200 (Sat, 12 May 2007) $ </XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name="UML" xmi.version="1.4"/></XMI.header>
  <XMI.content>
    <UML:Model xmi.id = '82A' name = 'LogicModel' isSpecification = 'false'
      isRoot = 'false' isLeaf = 'false' isAbstract = 'false'>
      <UML:Namespace.ownedElement>
        <UML:Actor xmi.id = '82B' name = 'Actor A' isSpecification = 'false' isRoot = 'false'
          isLeaf = 'false' isAbstract = 'false'>
          <UML:ModelElement.comment>
            <UML:Comment xmi.idref = '845'/>
          </UML:ModelElement.comment>
        </UML:Actor>
        <UML:UseCase xmi.id = '82C' name = 'Use Case 1' isSpecification = 'false'
          isRoot = 'false' isLeaf = 'false' isAbstract = 'false'>
          <UML:UseCase.include>
            <UML:Include xmi.idref = '844'/>
          </UML:UseCase.include>
        </UML:UseCase>
        <UML:Association xmi.id = '83C' name = 'A Starts UC1' isSpecification = 'false'
          isRoot = 'false' isLeaf = 'false' isAbstract = 'false'>
          <UML:Association.connection>
            <UML:AssociationEnd xmi.id = '83D' visibility = 'public' isSpecification = 'false'
              isNavigable = 'true' ordering = 'unordered' aggregation = 'none' targetScope = 'instance'
              changeability = 'changeable'>
              <UML:AssociationEnd.multiplicity>
                <UML:Multiplicity xmi.id = '83E'>
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange xmi.id = '83F' lower = '1' upper = '1'/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:AssociationEnd.multiplicity>
              <UML:AssociationEnd.participant>
                <UML:Actor xmi.idref = '82B'/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
            <UML:AssociationEnd xmi.id = '840' visibility = 'public' isSpecification = 'false'
              isNavigable = 'true' ordering = 'unordered' aggregation = 'none' targetScope = 'instance'
              changeability = 'changeable'>
              <UML:AssociationEnd.multiplicity>
                <UML:Multiplicity xmi.id = '841'>
                  <UML:Multiplicity.range>
                    <UML:MultiplicityRange xmi.id = '842' lower = '1' upper = '1'/>
                  </UML:Multiplicity.range>
                </UML:Multiplicity>
              </UML:AssociationEnd.multiplicity>
              <UML:AssociationEnd.participant>
                <UML:UseCase xmi.idref = '82C'/>
              </UML:AssociationEnd.participant>
            </UML:AssociationEnd>
          </UML:Association.connection>
        </UML:Association>
        <UML:UseCase xmi.id = '843' name = 'Use Case 2' isSpecification = 'false'
          isRoot = 'false' isLeaf = 'false' isAbstract = 'false'>
          <UML:ModelElement.comment>
            <UML:Comment xmi.idref = '845'/>
            <UML:Comment xmi.idref = '84A'/>
          </UML:ModelElement.comment>
        </UML:UseCase>
        <UML:Include xmi.id = '844' name = 'UC1 Rely on UC2' isSpecification = 'false'>
          <UML:Include.addition>
            <UML:UseCase xmi.idref = '843'/>
          </UML:Include.addition>
          <UML:Include.base>
            <UML:UseCase xmi.idref = '82C'/>
          </UML:Include.base>
        </UML:Include>
        <UML:Comment xmi.id = '845' name = 'Logic Fact 1' isSpecification = 'false'
          body = 'Fact: NOT Trigger&#10;Status: ERROR'>
          <UML:Comment.annotatedElement>
            <UML:Actor xmi.idref = '82B'/>
            <UML:UseCase xmi.idref = '843'/>
          </UML:Comment.annotatedElement>
        </UML:Comment>
        <UML:Actor xmi.id = '848' name = 'Actor B' isSpecification = 'false' isRoot = 'false'
          isLeaf = 'false' isAbstract = 'false'>
          <UML:ModelElement.comment>
            <UML:Comment xmi.idref = '84A'/>
          </UML:ModelElement.comment>
          <UML:GeneralizableElement.generalization>
            <UML:Generalization xmi.idref = '849'/>
          </UML:GeneralizableElement.generalization>
        </UML:Actor>
        <UML:Generalization xmi.id = '849' name = 'B Inherits from A' isSpecification = 'false'>
          <UML:Generalization.child>
            <UML:Actor xmi.idref = '848'/>
          </UML:Generalization.child>
          <UML:Generalization.parent>
            <UML:Actor xmi.idref = '82B'/>
          </UML:Generalization.parent>
        </UML:Generalization>
        <UML:Comment xmi.id = '84A' name = 'Logic Question 1' isSpecification = 'false'
          body = 'Question: Can Trigger?&#10;Answer: Yes'>
          <UML:Comment.annotatedElement>
            <UML:Actor xmi.idref = '848'/>
            <UML:UseCase xmi.idref = '843'/>
          </UML:Comment.annotatedElement>
        </UML:Comment>
      </UML:Namespace.ownedElement>
    </UML:Model>
  </XMI.content>
</XMI>
```

Massive amount of data even for small UC diagrams because representational Information (e.g. colours, positions and shapes) are stored as well.

Filtering out the logically relevant parts will lead to considerably more compact files for the logic-analysis engine.

Future Work: Back-Propagation of the automatically detected defects into the visual user-interface via the rich XML representation.

Thanks for your Attention

Questions**?**