Applying OCL for Model Validation in MagicDraw UML

Dr. **Darius Šilingas**

Principal MagicDraw UML Trainer @ No Magic Europe

About MagicDraw UML

- A visual UML modeling editor
- Domain-specific language (DSL) engine
- Roundtrip code engineering for Java, C++, C#, …
- Modeling teamwork control system
- Model documentation and report generation tool
- A modeling environment and model repository for *Enterprise Architecture* and *Model-Driven Architecture* paradigm
- Model validation engine using Dresden OCL Toolkit

- Developed since 1998 – second oldest UML tool in the market!
- Sold in >70 countries, used in different business domains
- Widely regarded as the most standard-compliant UML tool

Disclaimer

I use term model validation because it is used in MagicDraw user interface and documentation manuals

In precise computer science terminology, I will discuss model verification since it is based on precisely defined rules

Validate → Verify

Model Validation: Correctness and Completeness

Modeling like programming is an error-prone process

A user model can be either incorrect (*it breaks some rules*) or incomplete (*it lacks some required information*)

Rules defined in UML specification are automated in MagicDraw

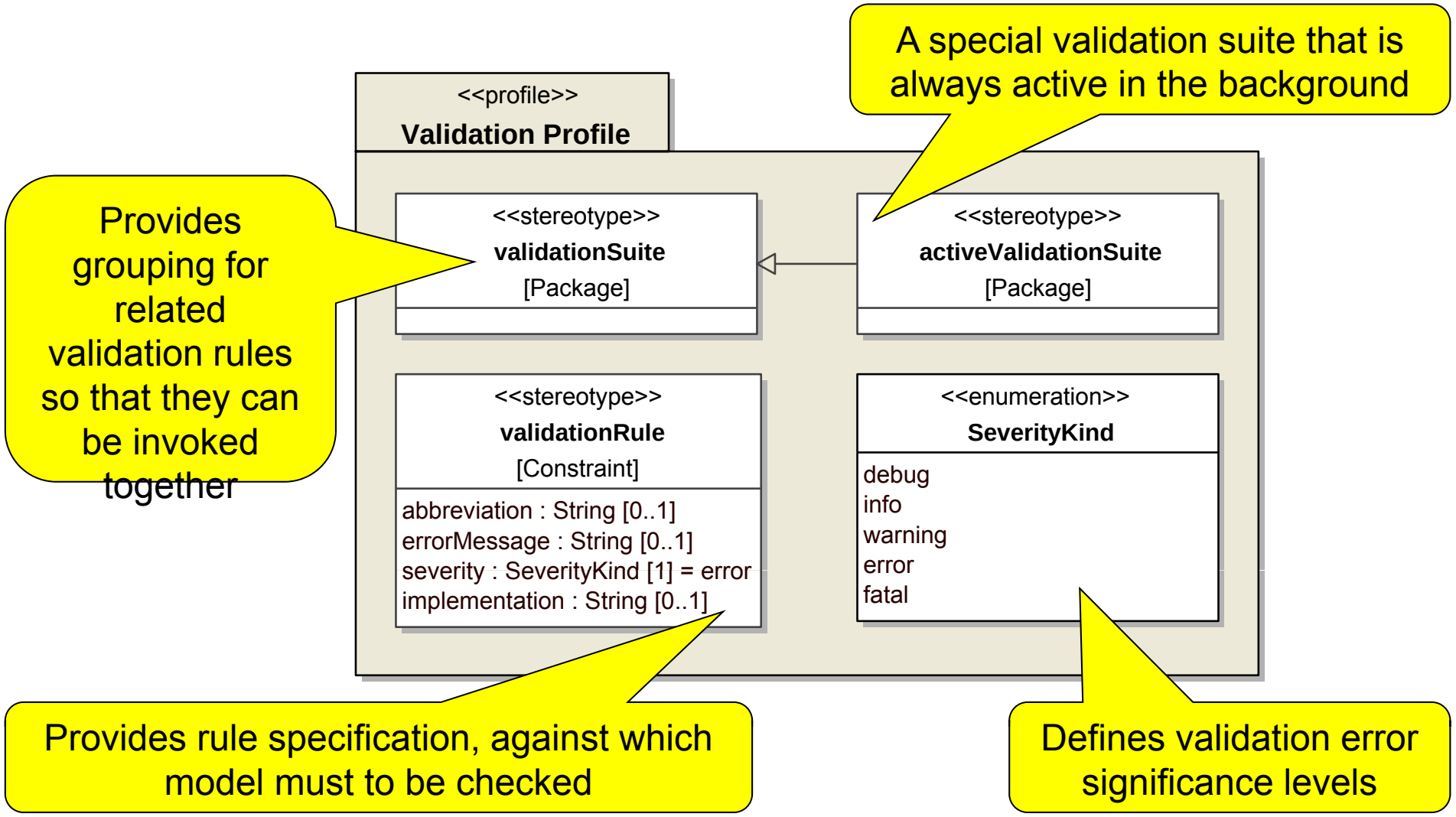
However, a specific modeling method typically implies additional rules

- Restriction to single generalization for classes
- Compulsory role names on navigable association ends
- Each use case must be documented with owned comment

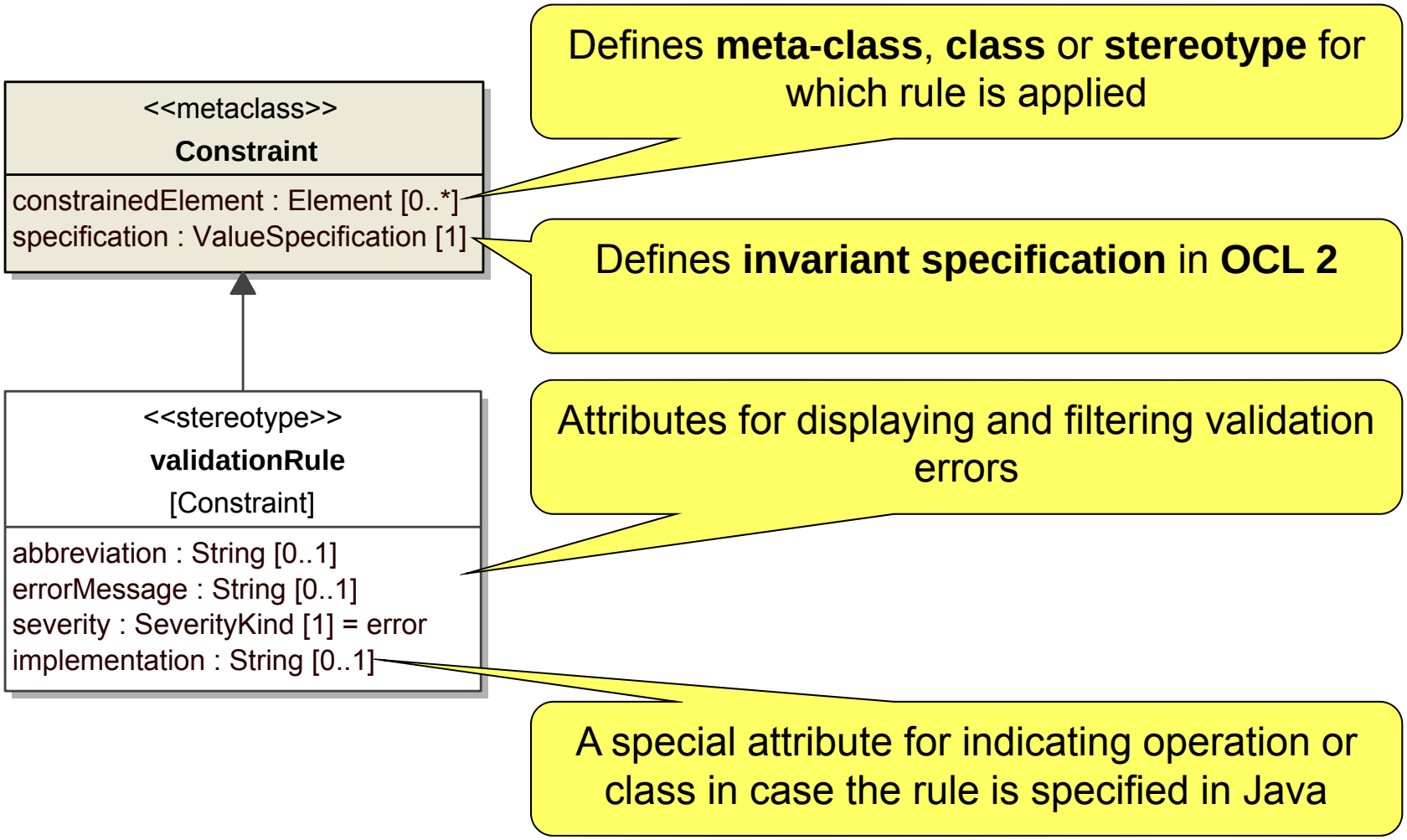
MagicDraw provides a way to define validation rules and validate models

- Validation rules can be specified on **OCL 2.0** or Java

MagicDraw Validation Profile



Validation Rule Properties



Validation Suite vs. Active Validation Suite

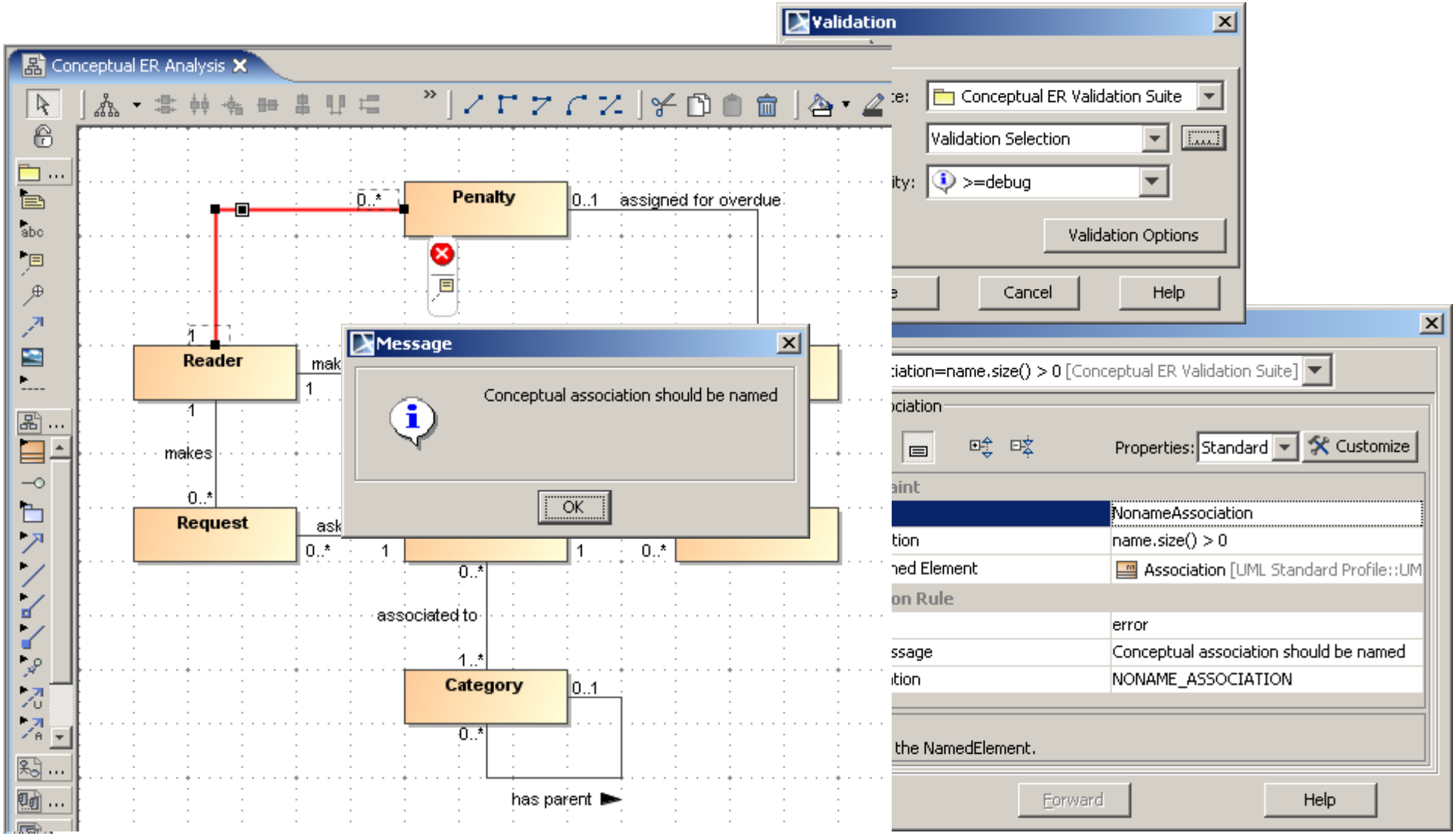
Validation rules that are collected in a validation suite either by nesting or element import relationship are invoked by user

- This is a typical case for model completeness validation
- Validation suite is re-initialized on every invocation

Validation rules that are collected in an active validation suite are invoked automatically when user changes the model

- This is a typical case for model correctness validation
- It is implemented in an intelligent way so that they fire only when relevant changes in model are performed
- For better performance, it is recommended to specify them in Java
- Active validation suite is initialized at project load time

Simple Model Validation Demo



A Case Study for Various Validation Rule Examples

1. An university needs a system MagicTest, which automates test assessments.
2. Teachers specify and maintain questions.
3. Each question must be applicable for 1 or more courses.
4. A question can be closed or open.
5. A closed question defines an ordered set of answer options, where at least one answer option is correct and at least one is incorrect.
6. An open question defines an expected correct answer.
7. Teachers define tests for particular classes that they are running.
8. A test collects a number of questions.
9. A test author also specifies test title, instructions, and allowed time.
10. Students participate in test assessments by providing answers to test questions.
11. MagicTest calculates test assessment evaluations.
12. Data about teachers, students, courses and classes are provided by University Data System (UDS).

Validation Rules with Constrained Meta-Classes

COMPLETENESS

- Each **use case** must be documented with an **owned comment**

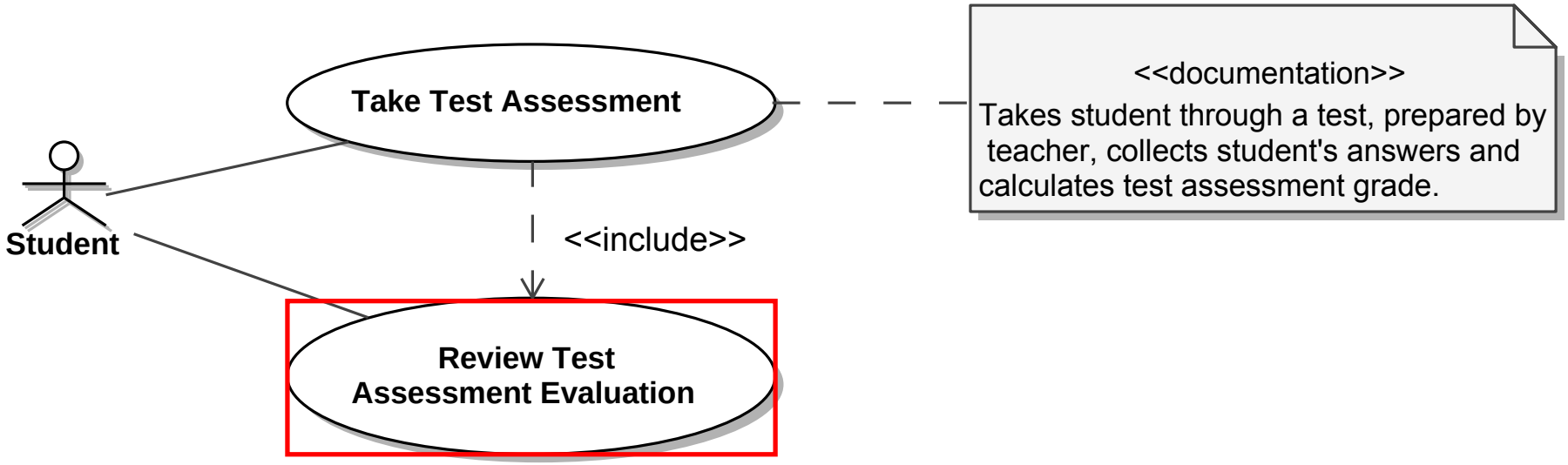
```
context UseCase inv UseCaseDocumentation
ownedComment->size()>0
```

CORRECTNESS

- Each **action** cannot have more than one incoming **control flow**

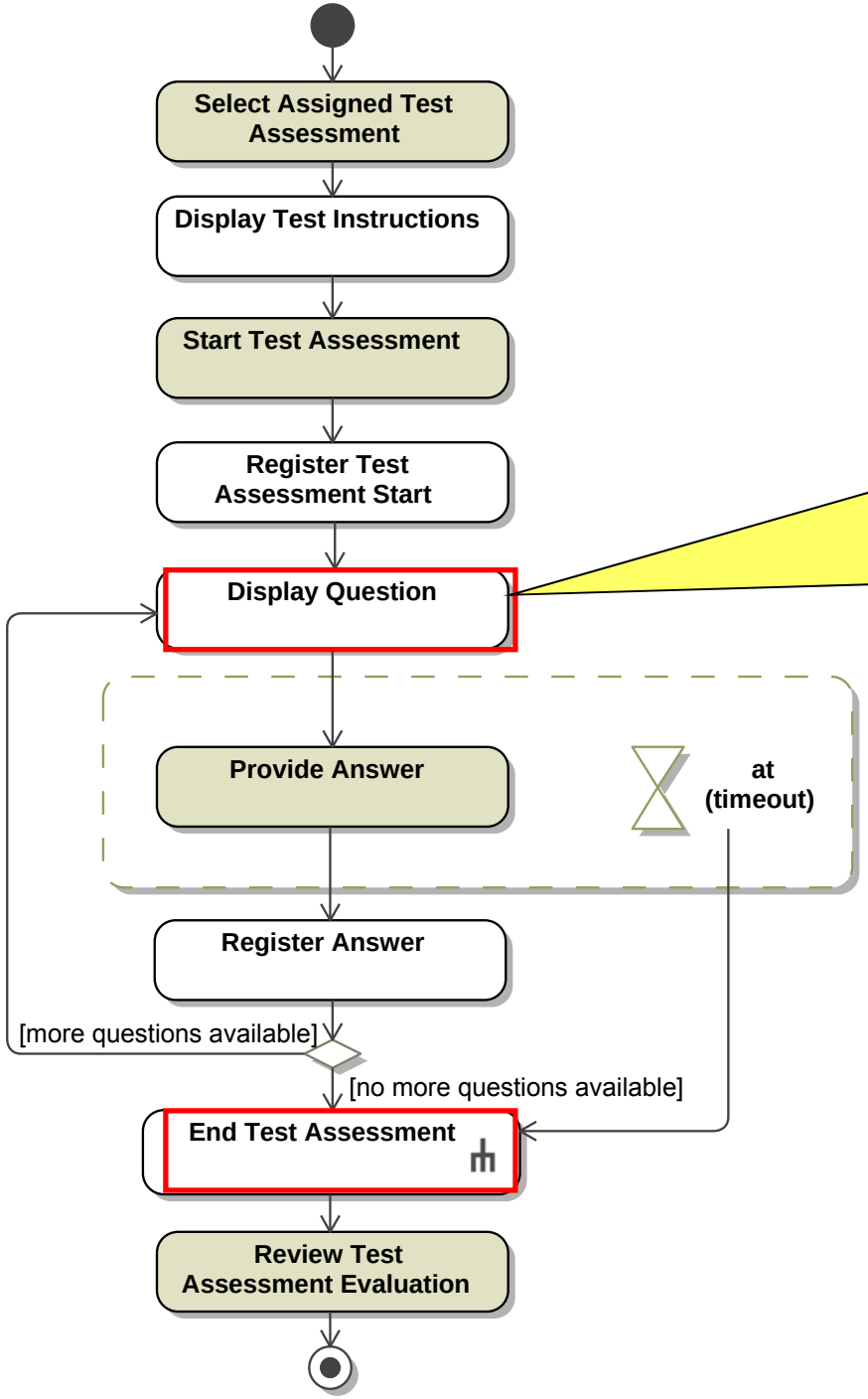
```
context Action inv SingleIncomingControlFlow
incoming->size()<2
```

Missing Use Case Documentation



In MagicDraw UML the first owned comment is treated as “official” model element documentation.

Multiple Incoming Control Flows



I will demonstrate later how to implement an auto-resolution for this validation rule violations.
However, this must be done in Java as OCL is effect-free language and we need to fix the model…

Validation Rules with Constrained Stereotypes

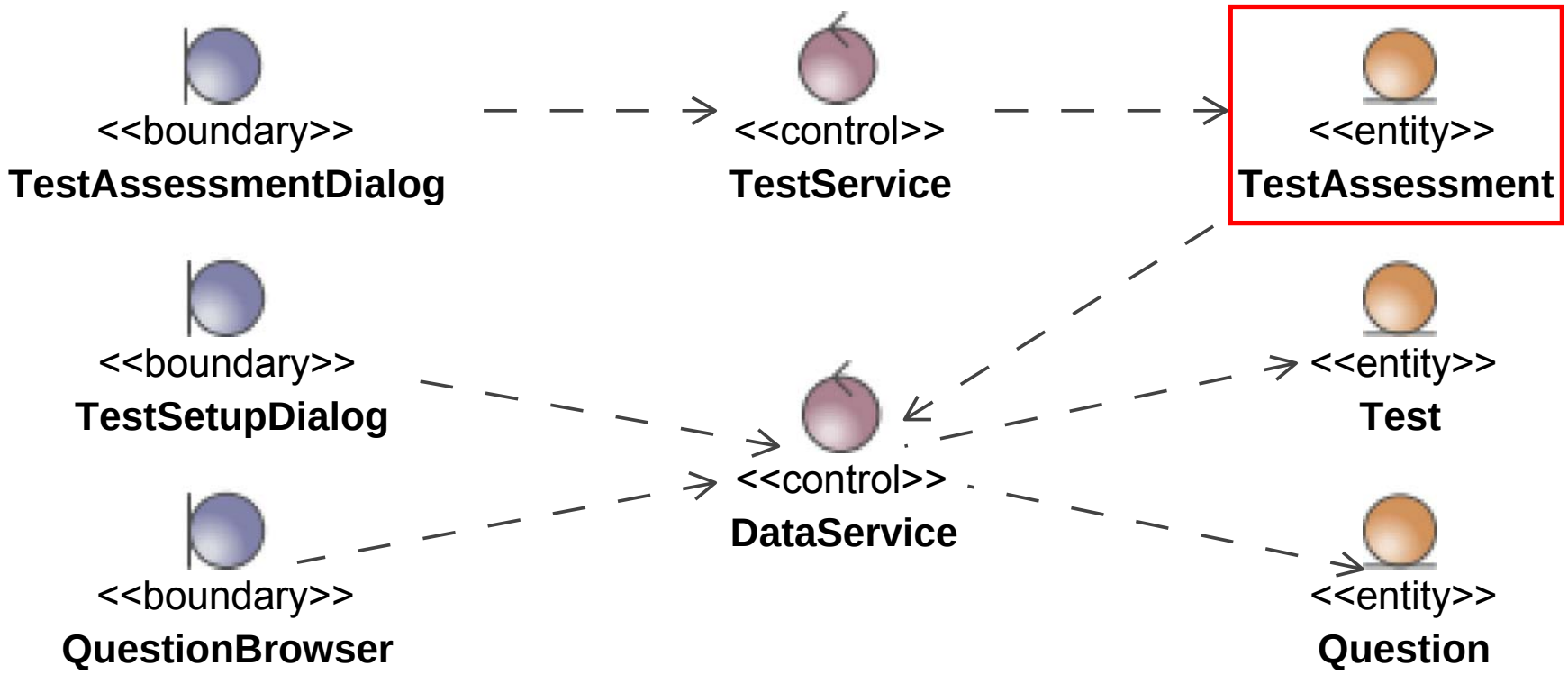
We will be using robustness analysis method, which requires a profile with stereotypes «boundary», «control», «entity»

CORRECTNESS

- Boundary components can use only boundary and control components
- Control components can use only control and entity components
- **Entity components can use only entity components**

```
context entity inv UseOnlyEntities
clientDependency->forAll(cd | cd.supplier->forAll(s | s.oclIsKindOf(entity)))
```

Incorrect Dependency from Entity to Control



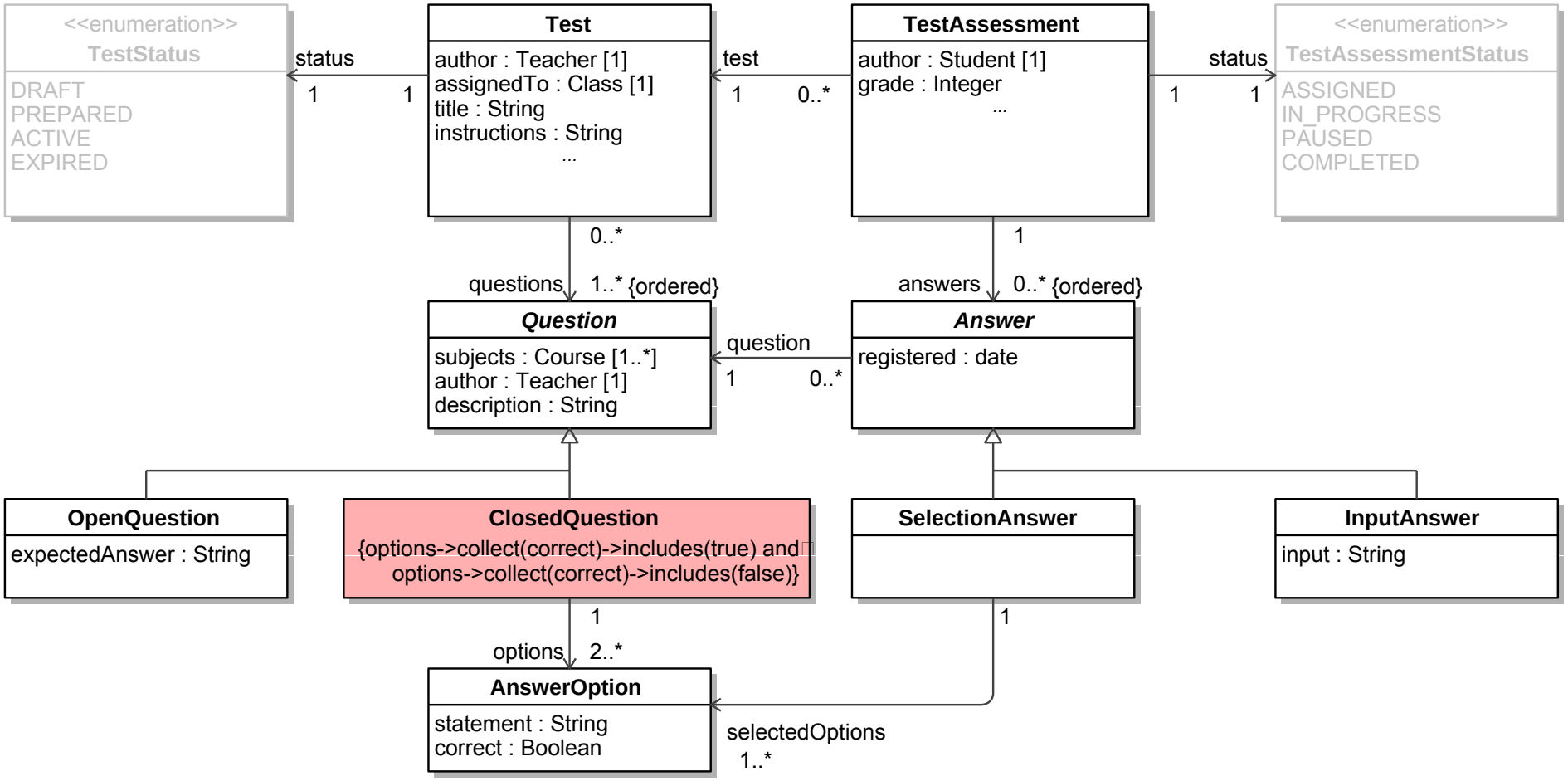
MagicDraw provides DSL engine, which enables modelers to enforce such stereotype-specific rules without OCL

Validation Rules with Constraints on Classes

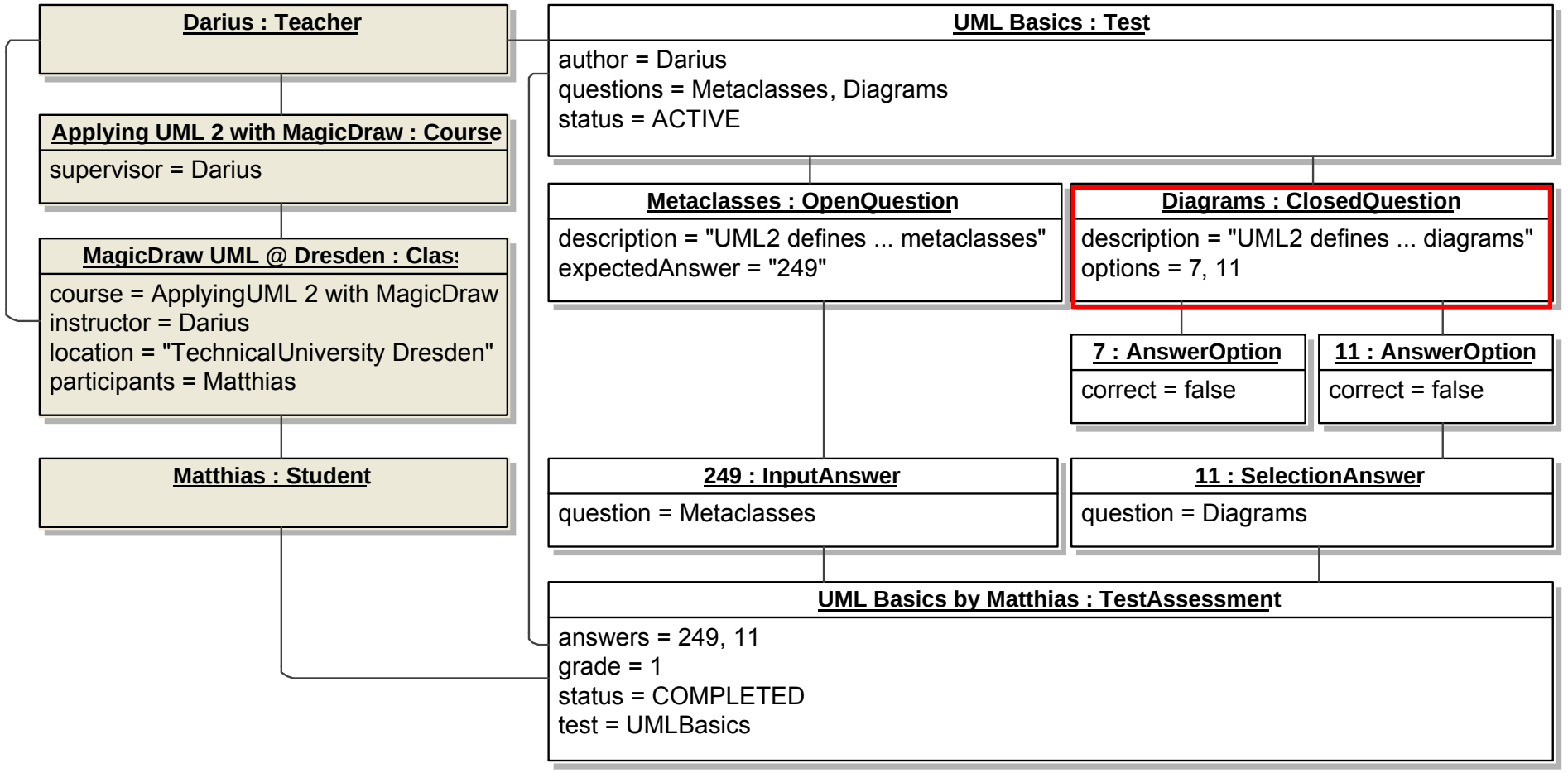
COMPLETENESS | CORRECTNESS

- A **closed question** defines an ordered set of **answer options**, where at least one **answer option** is correct and at least one is incorrect

Data Structure Design



Data Sample: Instance Specifications



Ideas for Future OCL-Driven Features in MagicDraw

Update to the most recent version of Dresden OCL toolkit ☺

- Needs a bridge from MOF-based to EMF-based model

Integrate intelligent OCL editor into constraint specification

- Easier specification and debugging of OCL 2.0 expressions

Support implicit model element relationship analysis

- Specify criteria in OCL for dependency matrix and related elements finder

Generate code for operation bodies from OCL expressions

- Applicable for query operations, test cases

Thank You for Attention!

Questions ? ? ?

Let's Keep in Touch:

Dr. **Darius Šilingas**

- E-mail: **darius.silingas@nomagic.com**
- Skype: **darius.silingas**
- Phone: **+370 37 705899**