
# Model-Driven Development with MagicDraw UML

**Dr. Darius Silingas**
Principal MagicDraw UML Trainer/Consultant

# Lecturer


Dr. **Darius Šilingas**, darius.silingas@nomagic.com
Principal MagicDraw UML Trainer/Consultant

Trained & consulted MagicDraw customers in 17 countries

- USA, UK, Germany, France, South Africa, Russia, …

Received Ph.D. in Informatics from **Vytautas Magnus University** in 2005

Spoke at Architecture & Design World, OOP, JAX

Holds the following professional certificates:

- Microsoft Certified Professional
- Sun Certified Programmer for the Java 2 Platform 1.4
- OMG-Certified UML Professional Advanced

# Contents

- UML and Software Development Process
- Introduction to MagicDraw UML
- UML Case Study
- Demo of Selected MagicDraw Features

# What Is UML?

A language (with graphical notation) for modeling object-oriented systems

A standard maintained by the Object Management Group (OMG)

Provides 249 modeling concepts (meta-classes) and defines 13 diagram types

A means for visualizing, specifying, constructing, and documenting systems

% of software developers using UML

- Almost every software development company

# of UML tools available

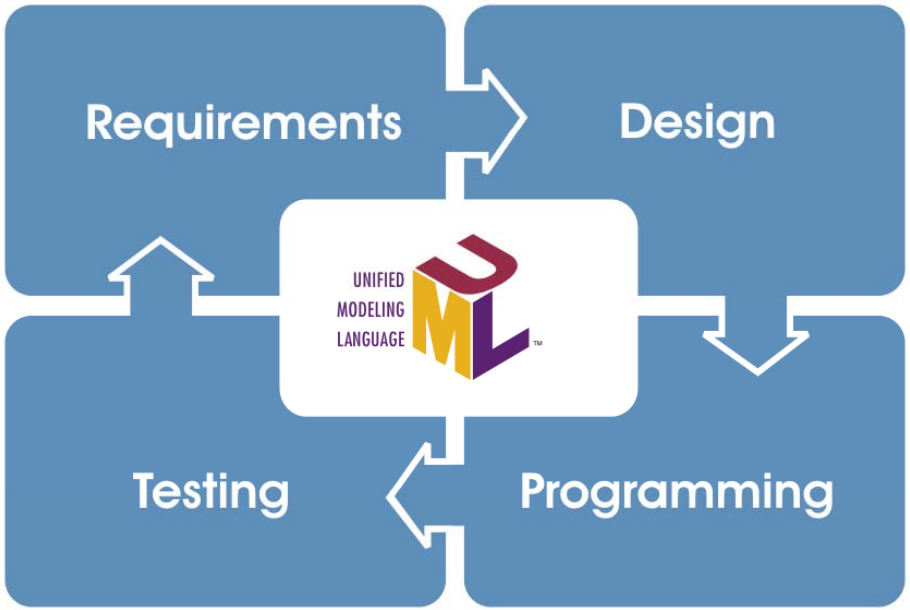
- ~10 really mature UML tools
  - MagicDraw, Rational Architect, Enterprise Architect, Artisan Studio, Together/J, Objecteering, …
- ~100 less mature UML tools, focusing on fragments of UML

→ http://www.uml.org

# Using UML in Software Workflows

Domain concepts and relations
Domain object lifecycle
Business processes
Actors and use cases
Use cases scenarios

Package/component structure
Interaction scenarios
Data structure
Service API
GUI navigation schemas




Test case action flows
Test data object structures
Interactions for test scenarios

Code generation from UML
Visualization of code structure
Model transformations

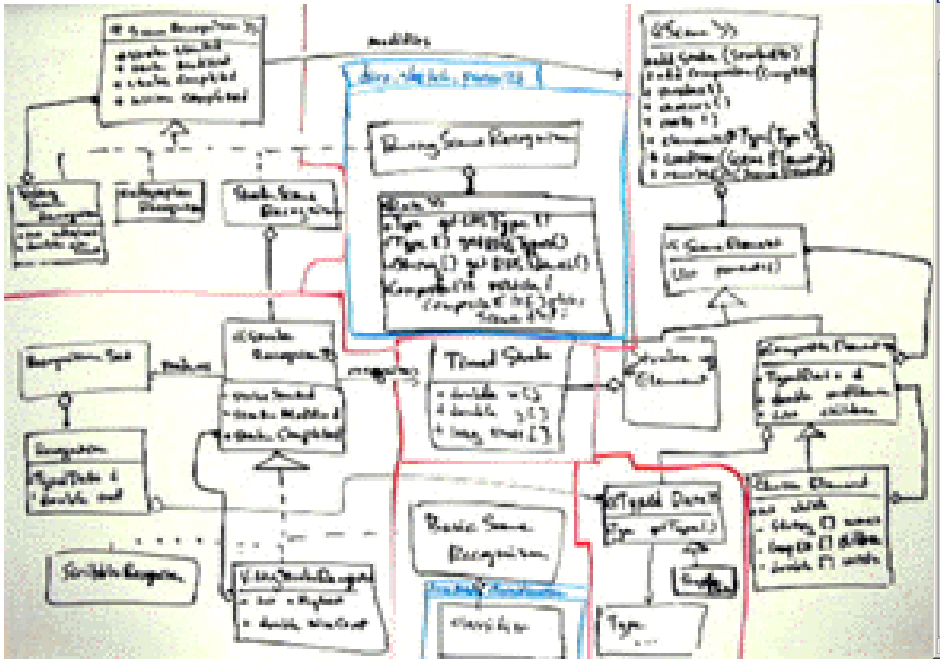
# Choice of Tools

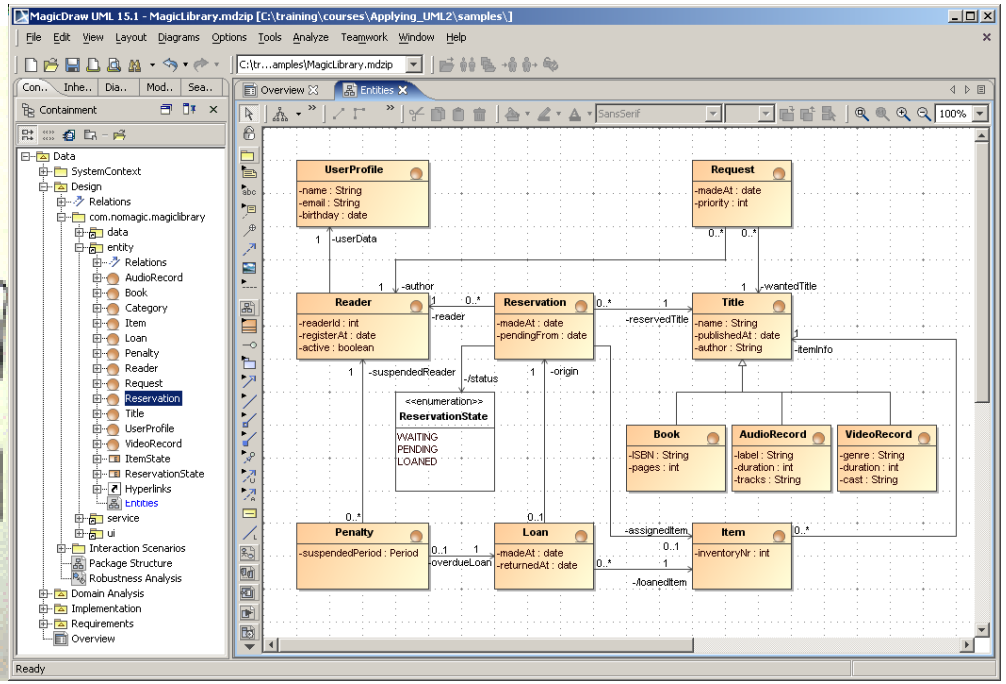
?

# Informal Modeling vs. Using UML Tool

- **Informal Modeling**
  - Easy to start
  - Very useful for brainstorming
  - No need to buy tools
  - Difficult to maintain
  - Allows deviations from standard
  - Reuse is not possible

- **Using UML Tool**
  - Reusable model repository
  - Model analysis tools
  - Model 2 code transformations
  - Teamwork possibility
  - Learning to use tool
  - Need to invest into buying the tool





# What is MagicDraw UML?

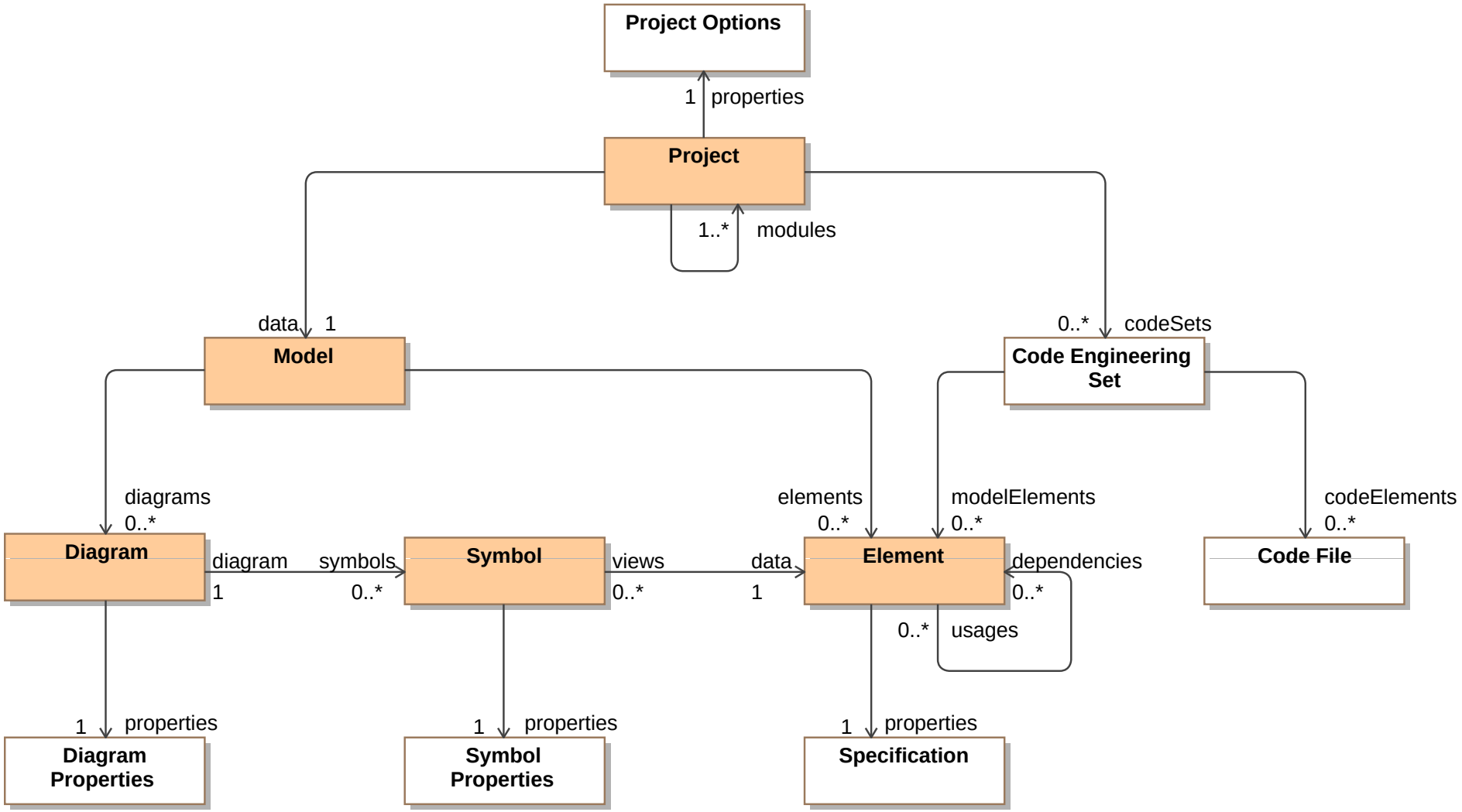




- A visual UML modeling editor
- Domain-Specific Language (DSL) engine
- A roundtrip code engineering tool
- Business process modeling tool
- A UML model teamwork control system (using Teamwork Server)
- System documentation tool (using the built-in and custom reports)
- A modeling environment and model repository for *Enterprise Architecture* and *Model-Driven Architecture* paradigms

- 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-complient UML tool

# Major MagicDraw UML Concepts



# MagicDraw Differentiation from Other UML Tool

Most compliant to the standard

High modeling productivity

Highly configurable and extendible

Free technical support

Best value for the price

# 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).

**CHECK OUT *MagicTest.mdzip* MODEL *for* ARTEFACTS**

# Current MagicDraw UML Functionalities

## Model Repository

## Graphical Diagramming

## Model Analysis Tools

- Search/Replace
- Find/Display Related Elements
- Calculate Metrics
- Define Dependency Matrices

## Configuration

- User Perspective
- Environment Options
- Project Options

## Environment Extendibility

- Open API
- Fragments of AMI
- Custom Reports

## Project Decomposition

- Projects & Modules

## Model Validation

- Using OCL or Java

## Teamwork Server

- Modeling in Teams

## DSL Engine

- Define custom modeling languages

## Project Version Management

- Branching, comparison, merge

## Model Reports

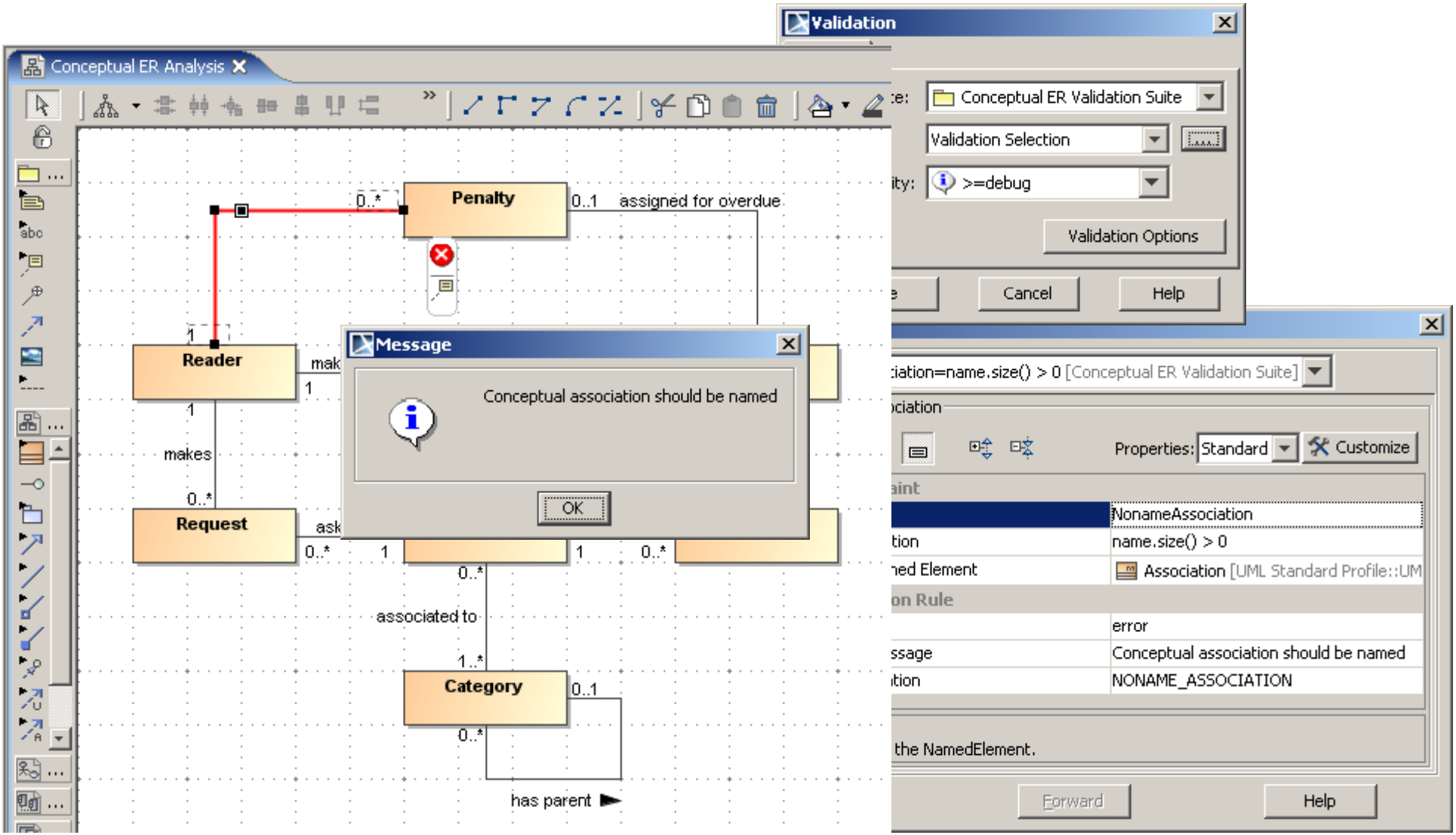
- Based on user-defined templates

## Code Engineering

- Generate code or reserve model

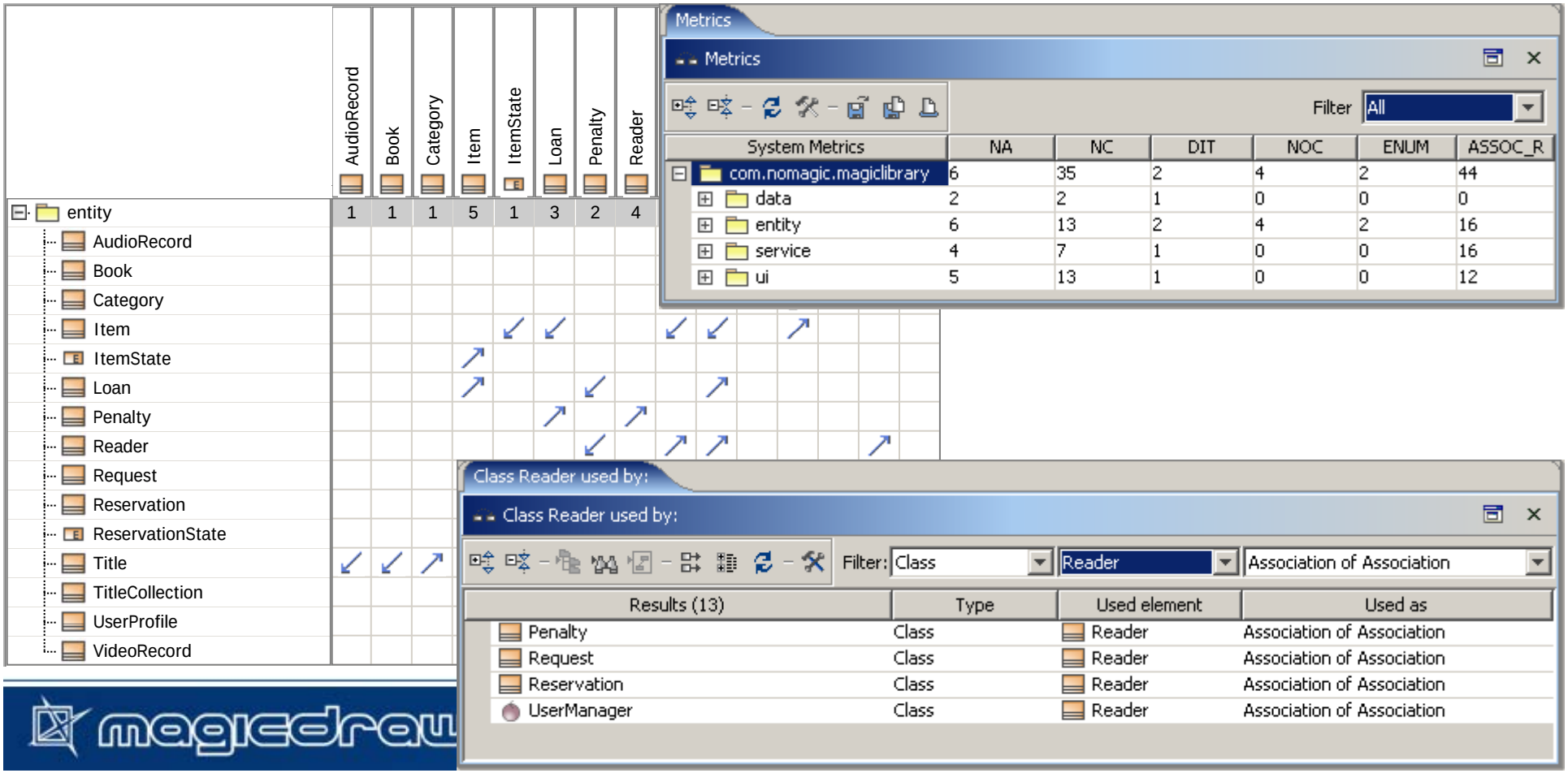
# Model Validation

- Created models may be incorrect or incomplete
- You can define validation rules in **OCL** and validate model against them

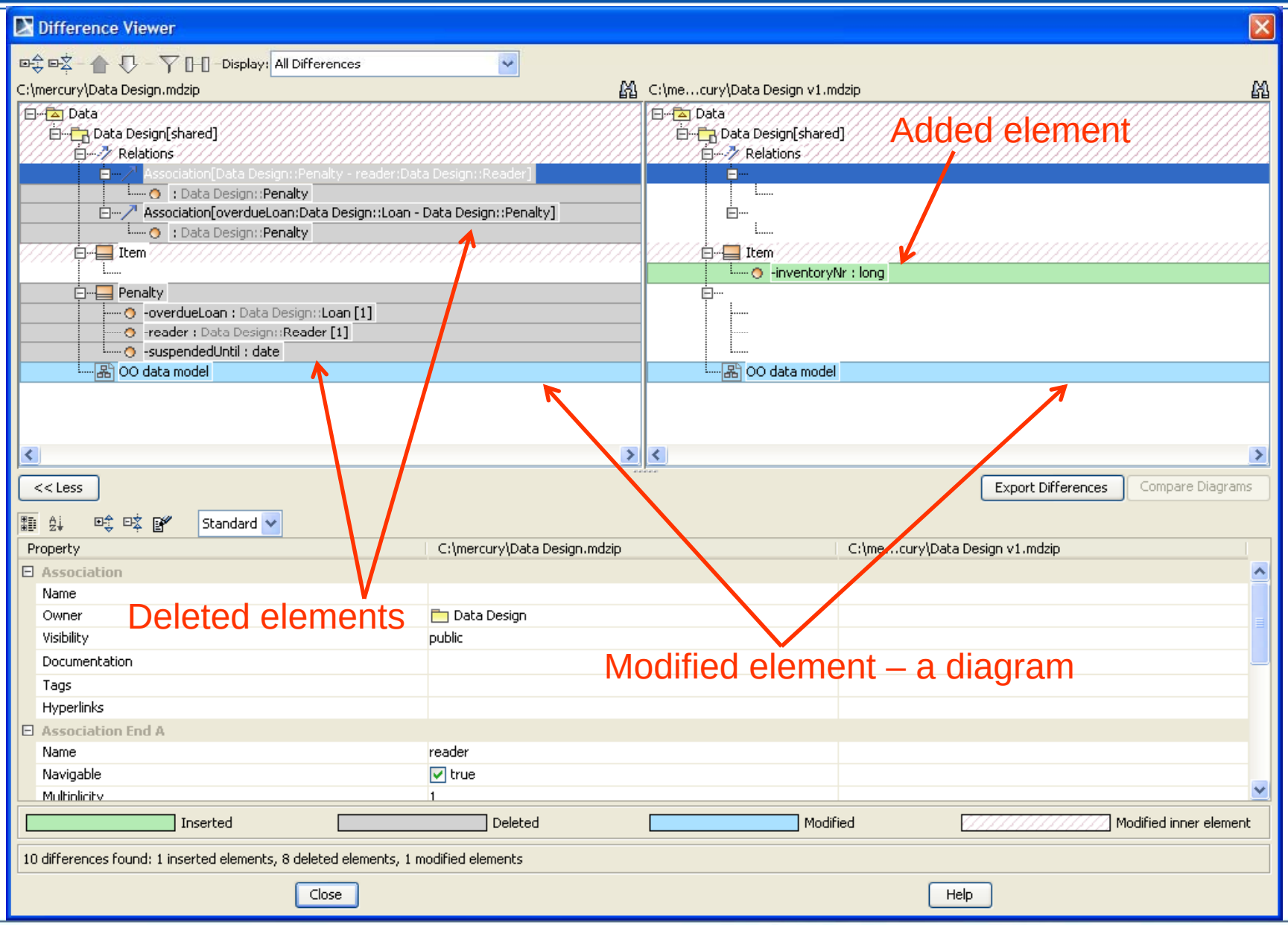


# Model Analysis Tools

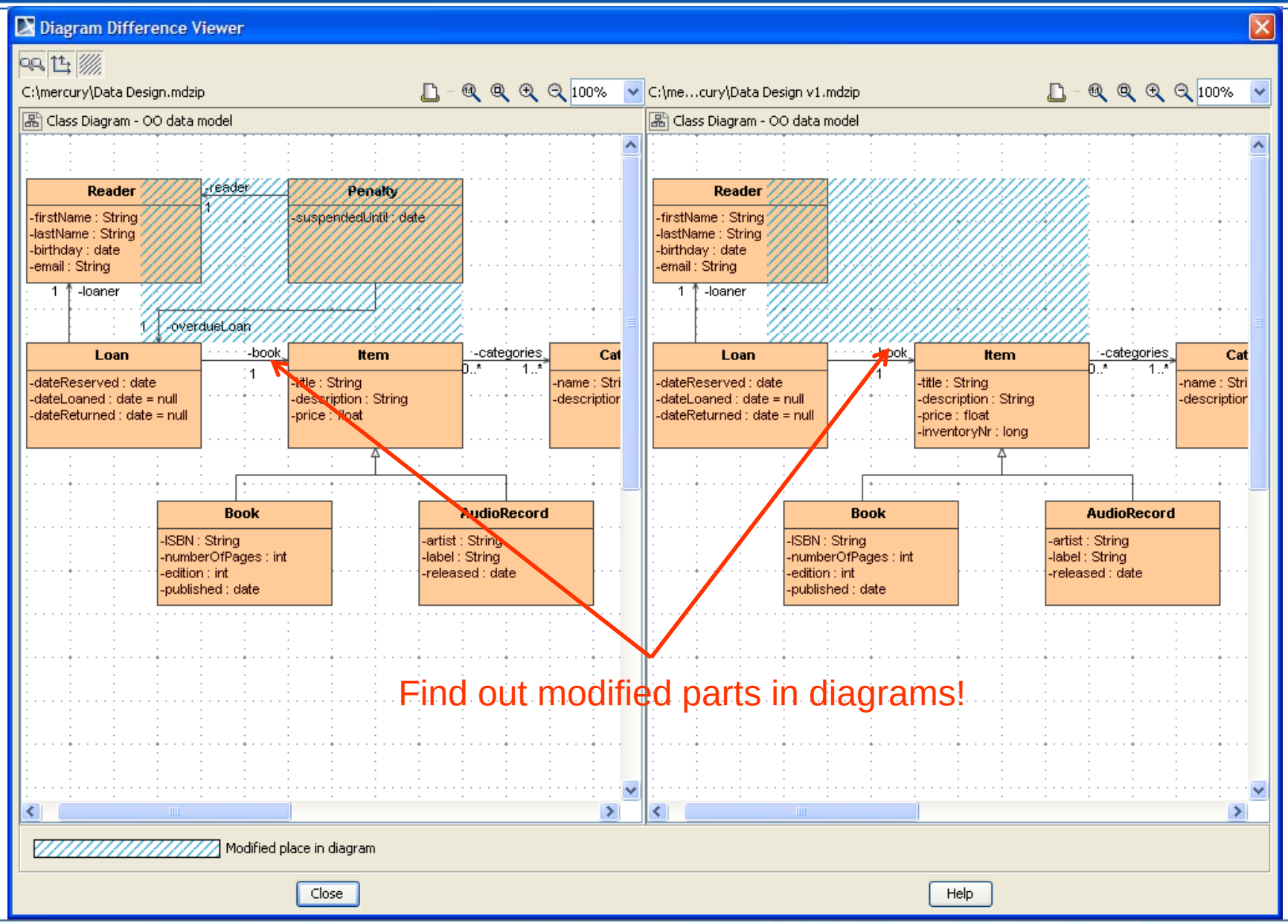
- Generating Relationship Matrix
- Finding model element Dependencies/Usages
- Calculating model metrics
- Comparing two versions of model data structures and diagrams



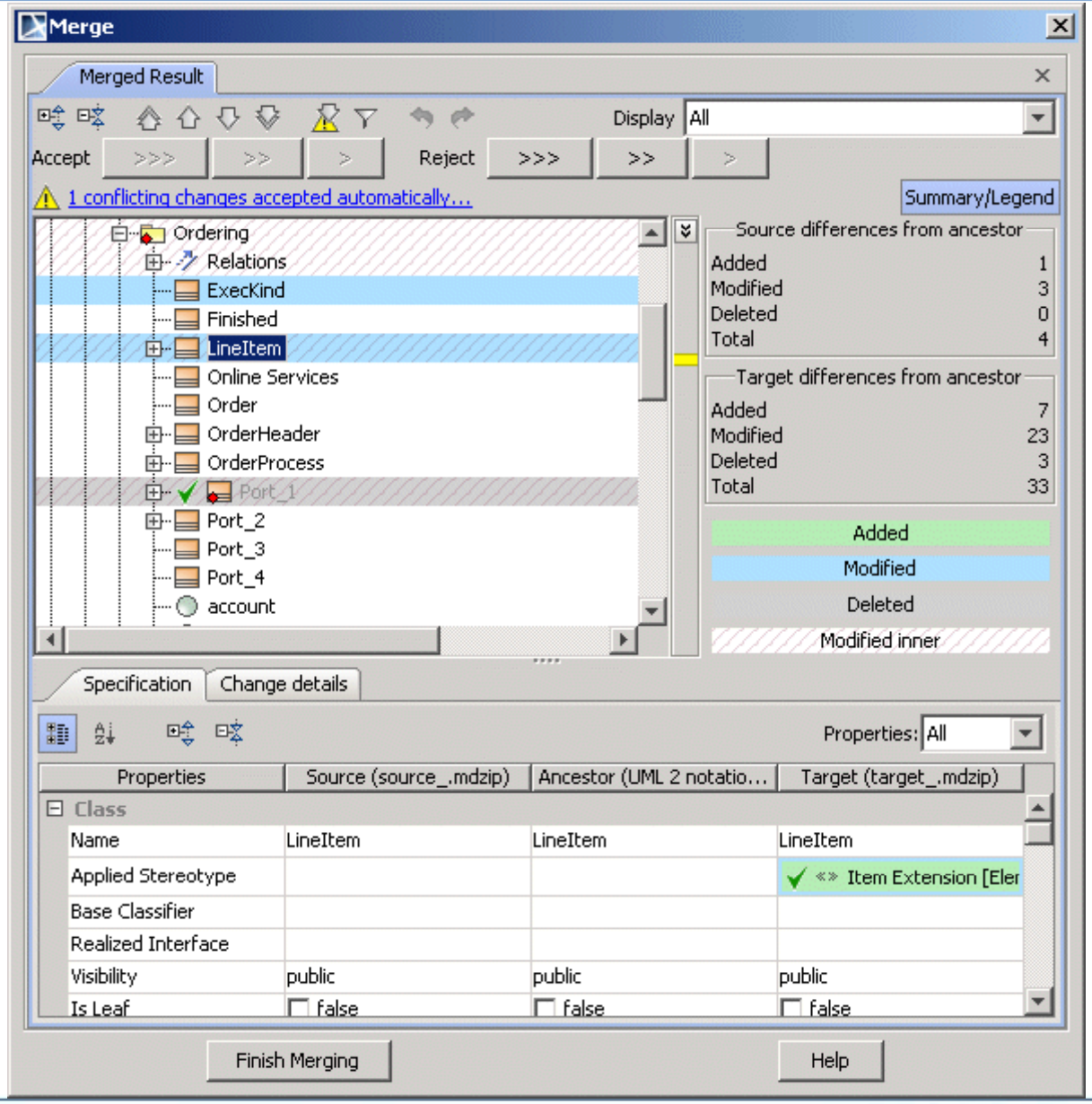
# Comparison of Model Data Structure Versions



# Comparison of Diagram Versions



# Merging Two Model Versions



# Modeling Teamwork

## Global project repository

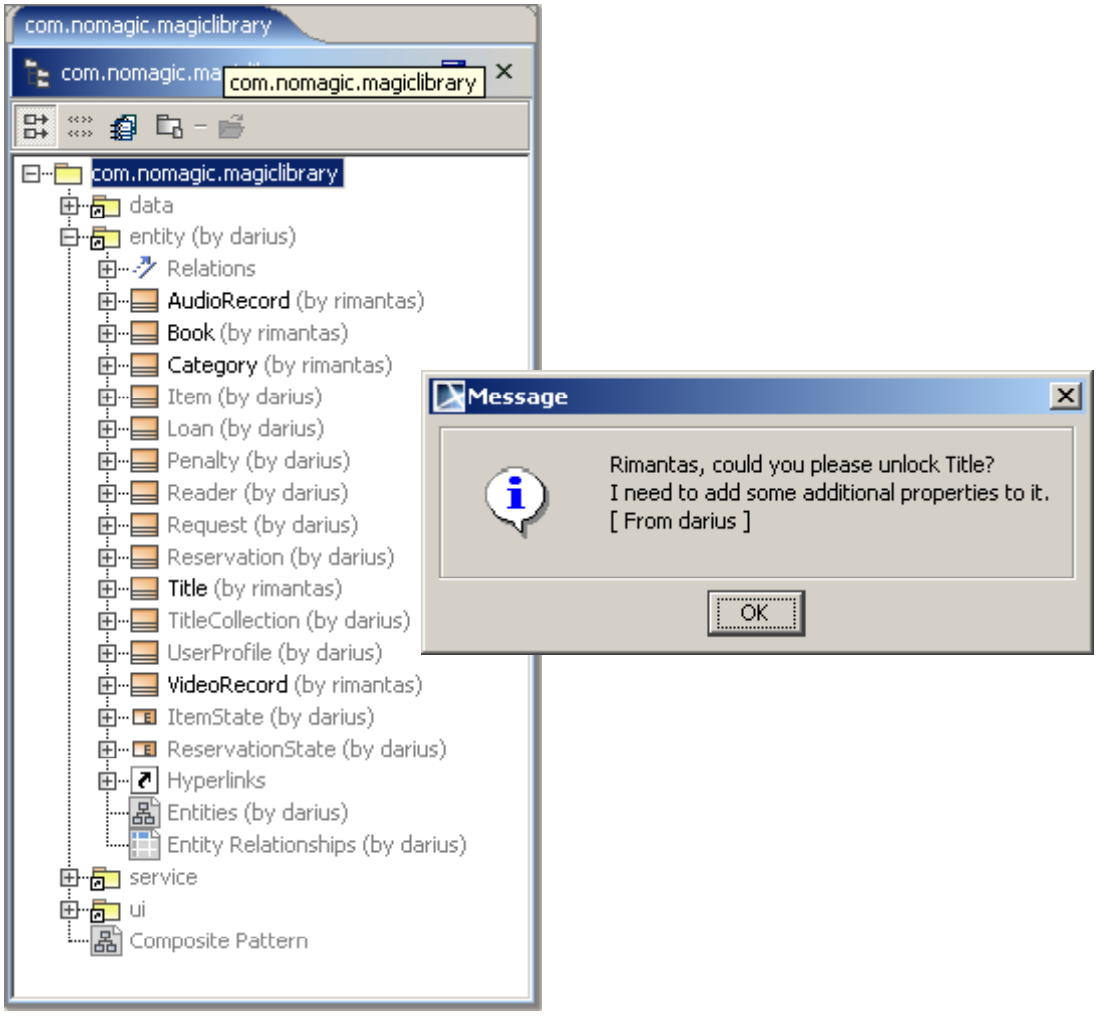
- Projects
- Users
- Permissions

## Collaboration inside a project

- Locking/unlocking model elements
- Seeing who has locked which elements
- Submitting changes

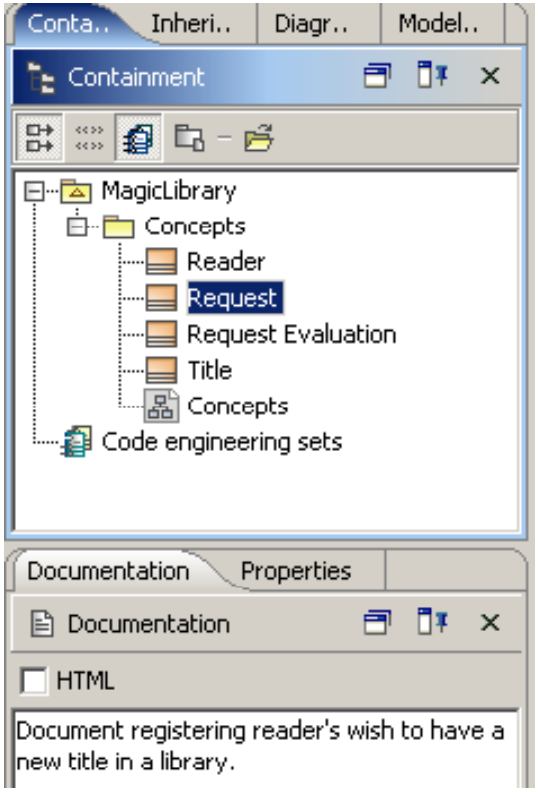
## Change management

- Versioning
- Branching
- Comparing
- Merging



# Model Transformations

- Model 2 Code using Code Engineering Sets
- Model 2 Model using plug-in Transformations
- Model 2 Document using Report Wizard with Templates

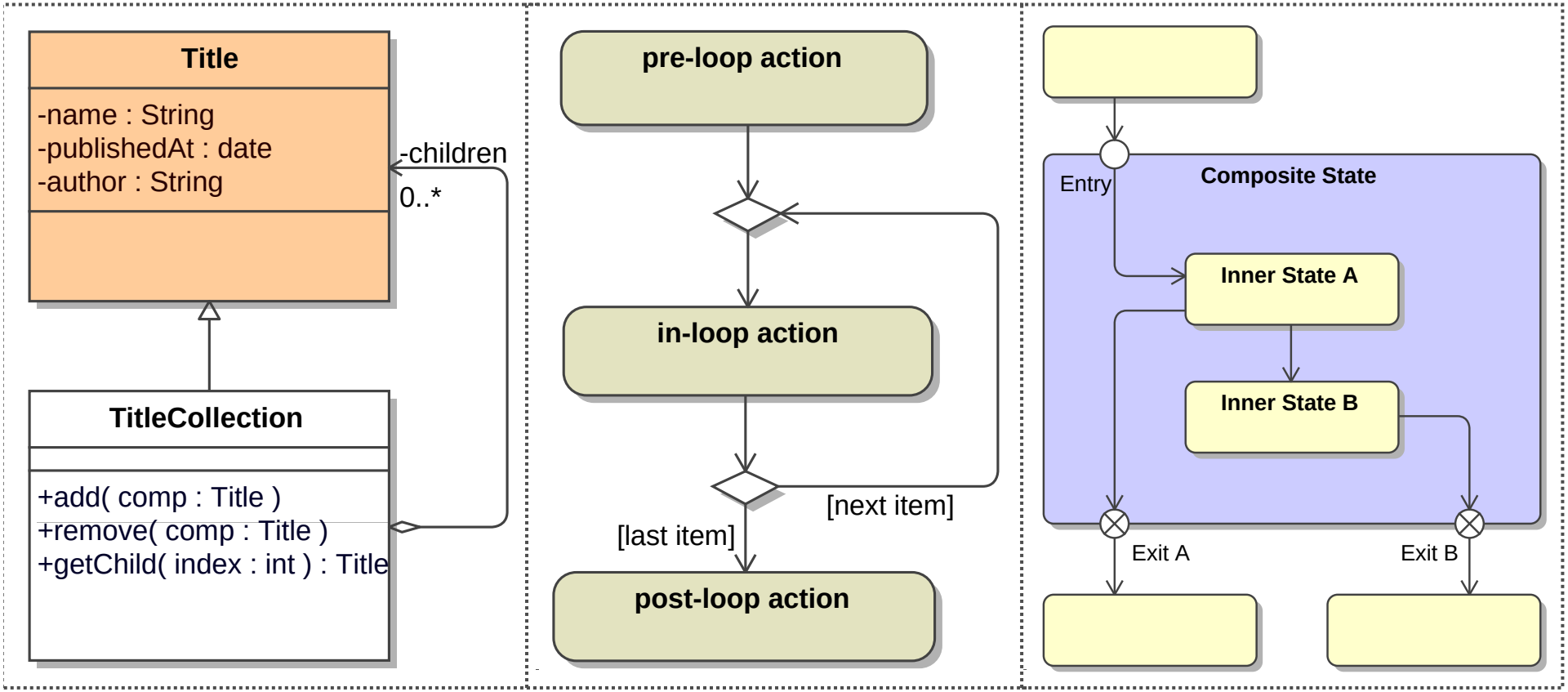



| Concept | Description |
|---|---|
| #forrow ($class in $sorter.sort($Class, “name”))<br>$report.getIconFor($class)<br>**$class.name** | *$report.getComment($class)*<br>#endrow |

| Concept | Description |
|---|---|
| **Reader** | *Information about library customer.* |
| **Request** | *Document registering reader's wish to have a new title in a library.* |
| **Request Evaluation** | *Librarian's decision whether to approve or deny reader's request.* |
| **Title** | *Information about a book, journal or another kind of library inventory item. Library may contain multiple copies of the same title.* |

# Model Patterns

- Apply classic GoF patterns
- Apply behavioral patterns
- Reuse predefined model fragments from libraries



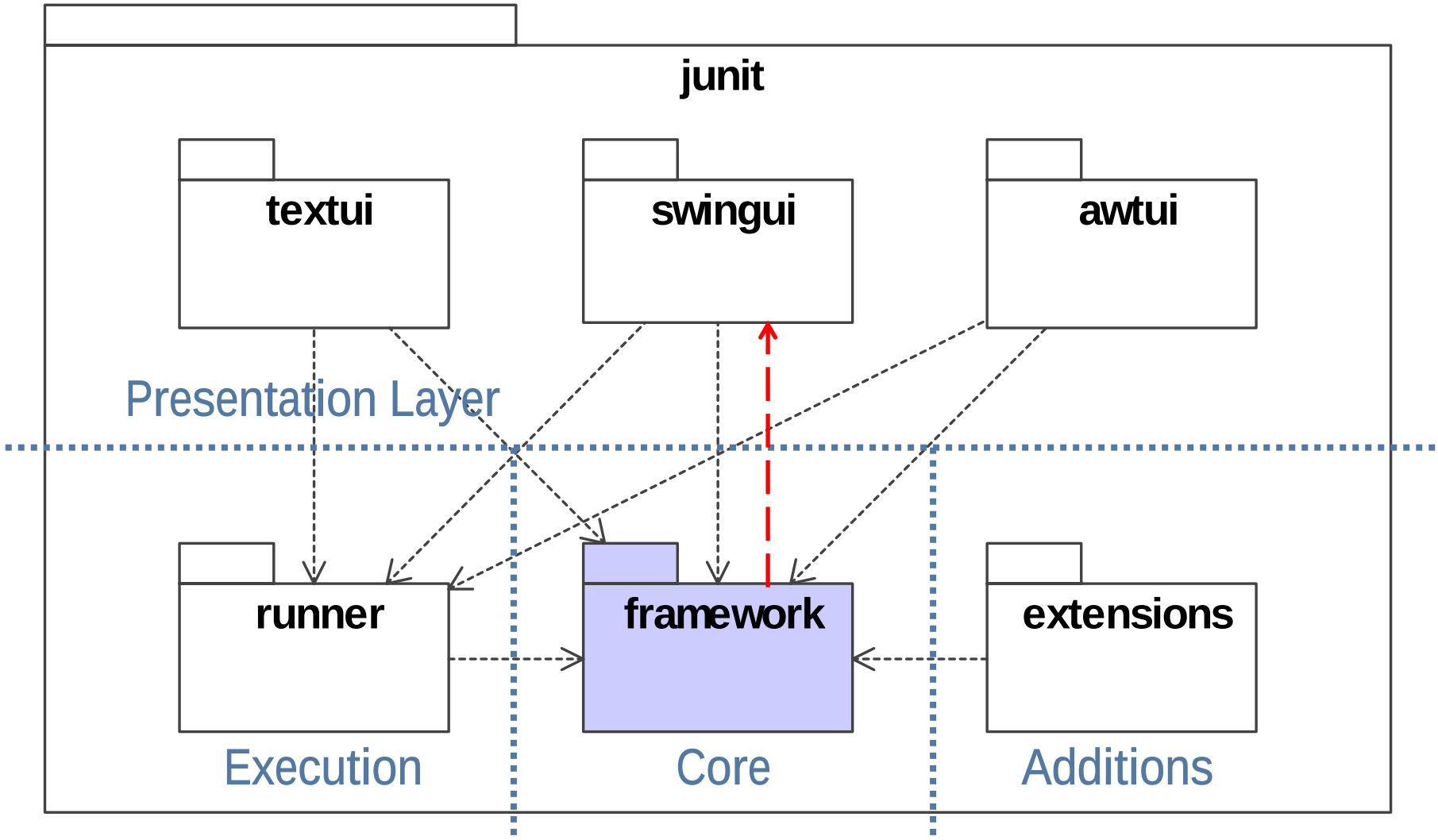
# Source Code Analysis Using UML

## You can analyze source code structure using UML as visualization

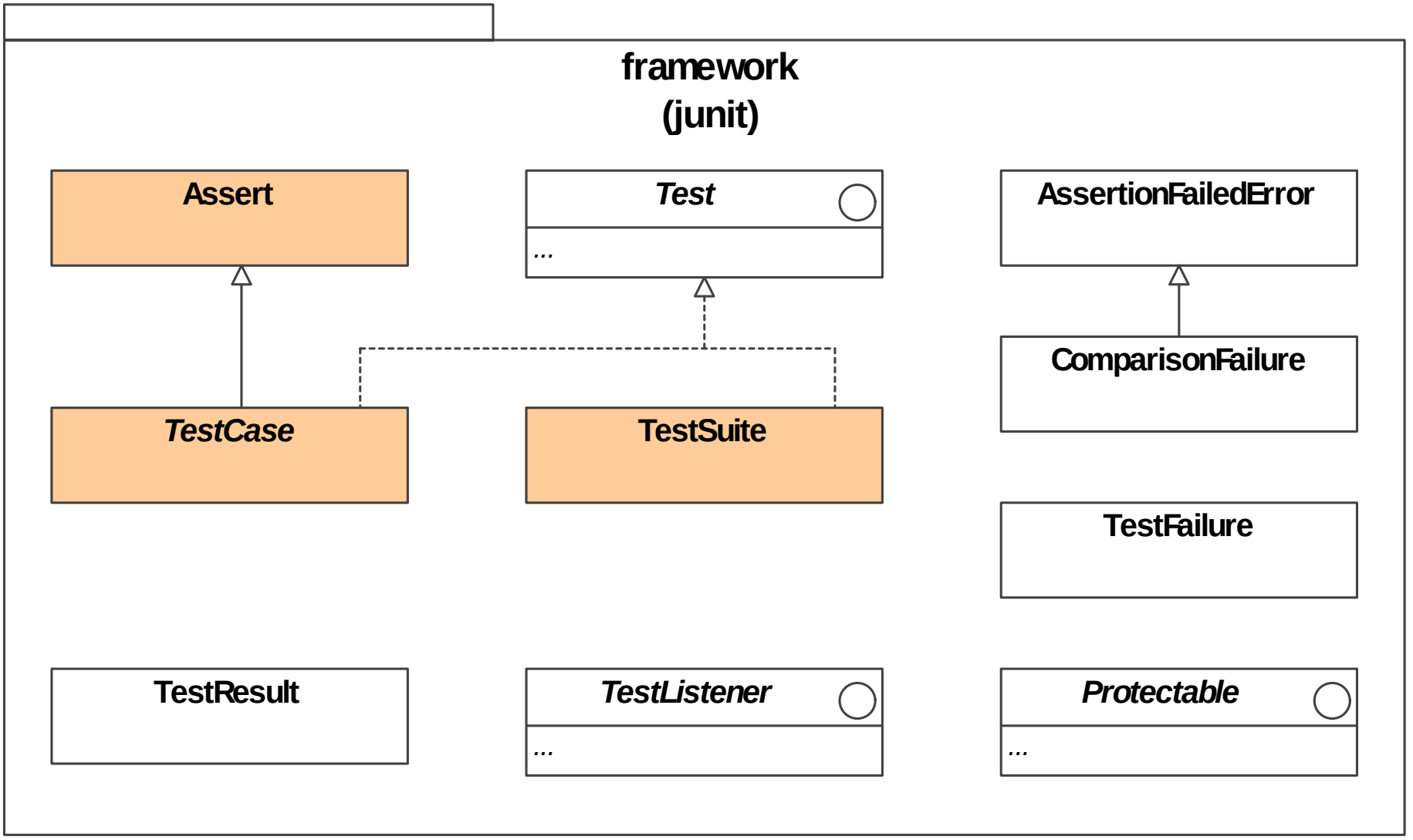
- Model can be reversed from source code using **reverse engineering** tools
- UML tools include **model visualization** tools that help to create diagrams from reversed model quickly
- For model analysis, it is common to use dependency diagrams
  - Possibility to track related elements
  - Calculation of various code analysis metrics

**EXERCISE:** Try out reverse engineering and analyzing **JUnit** framework

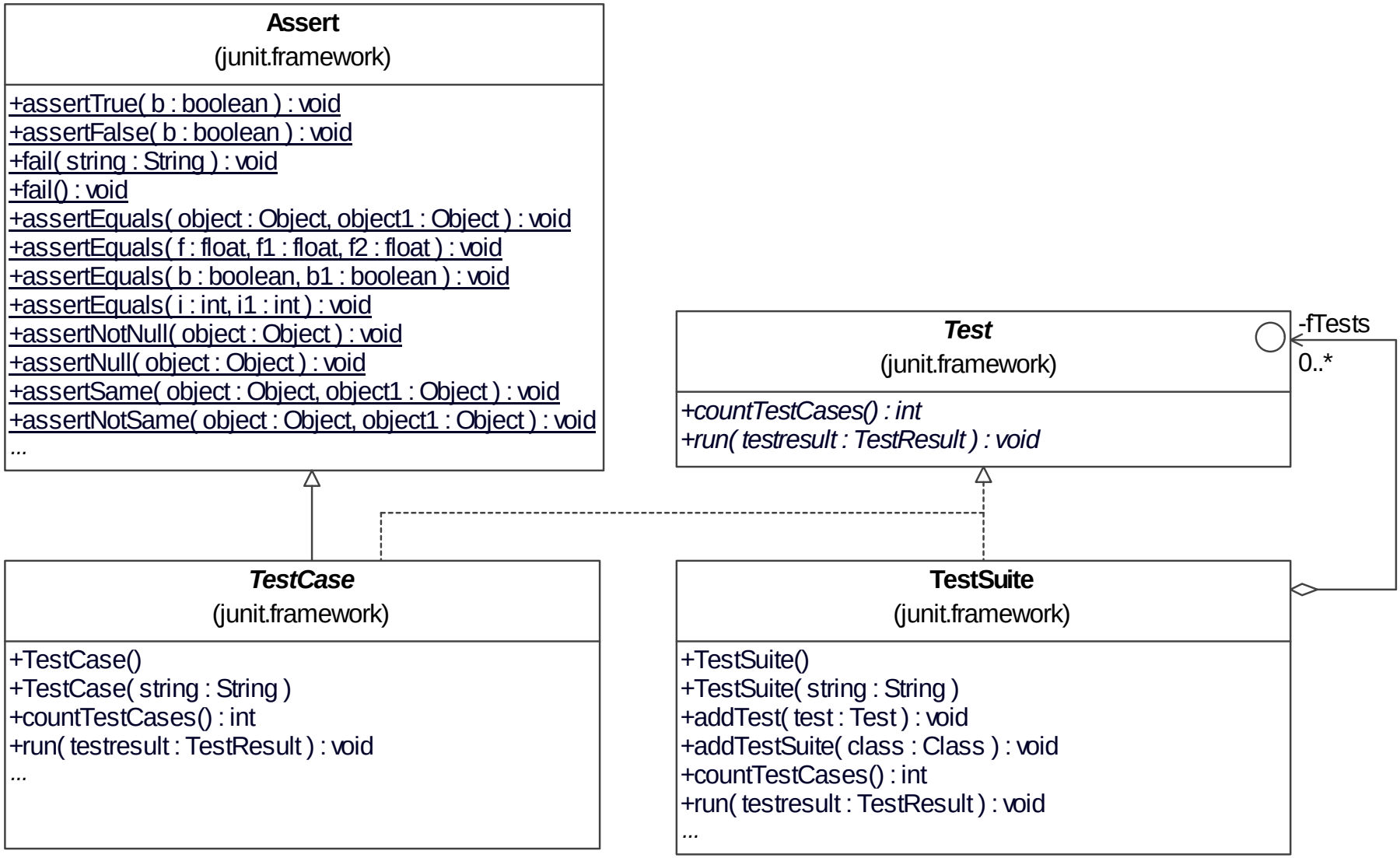
# Top-Level Package Dependency Diagram for JUnit Framework



# Overview of JUnit Package *junit.framework*

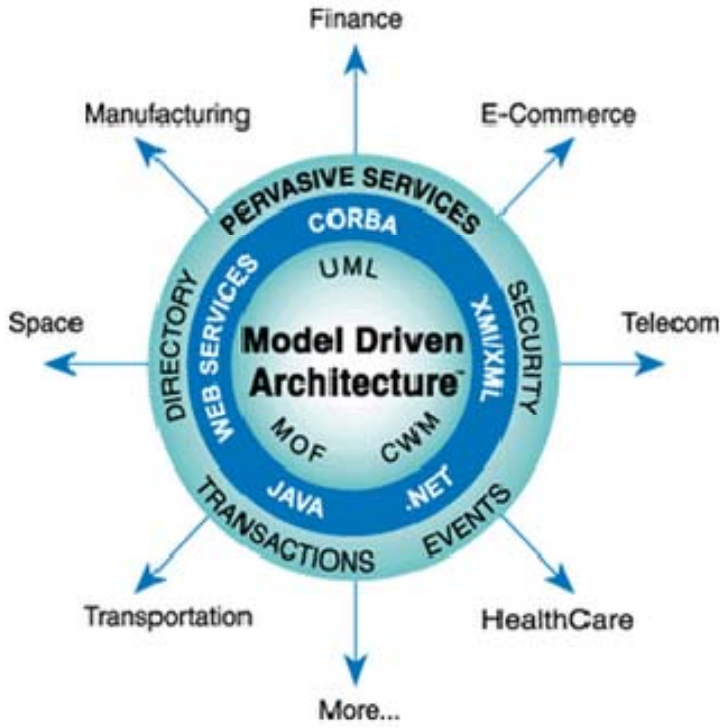


# Core JUnit Classes



# Model-Driven Architecture

OBJECT MANAGEMENT GROUP




→ http://www.omg.org/mda

Model-driven architecture (MDA) is a software design approach for the development of software systems.

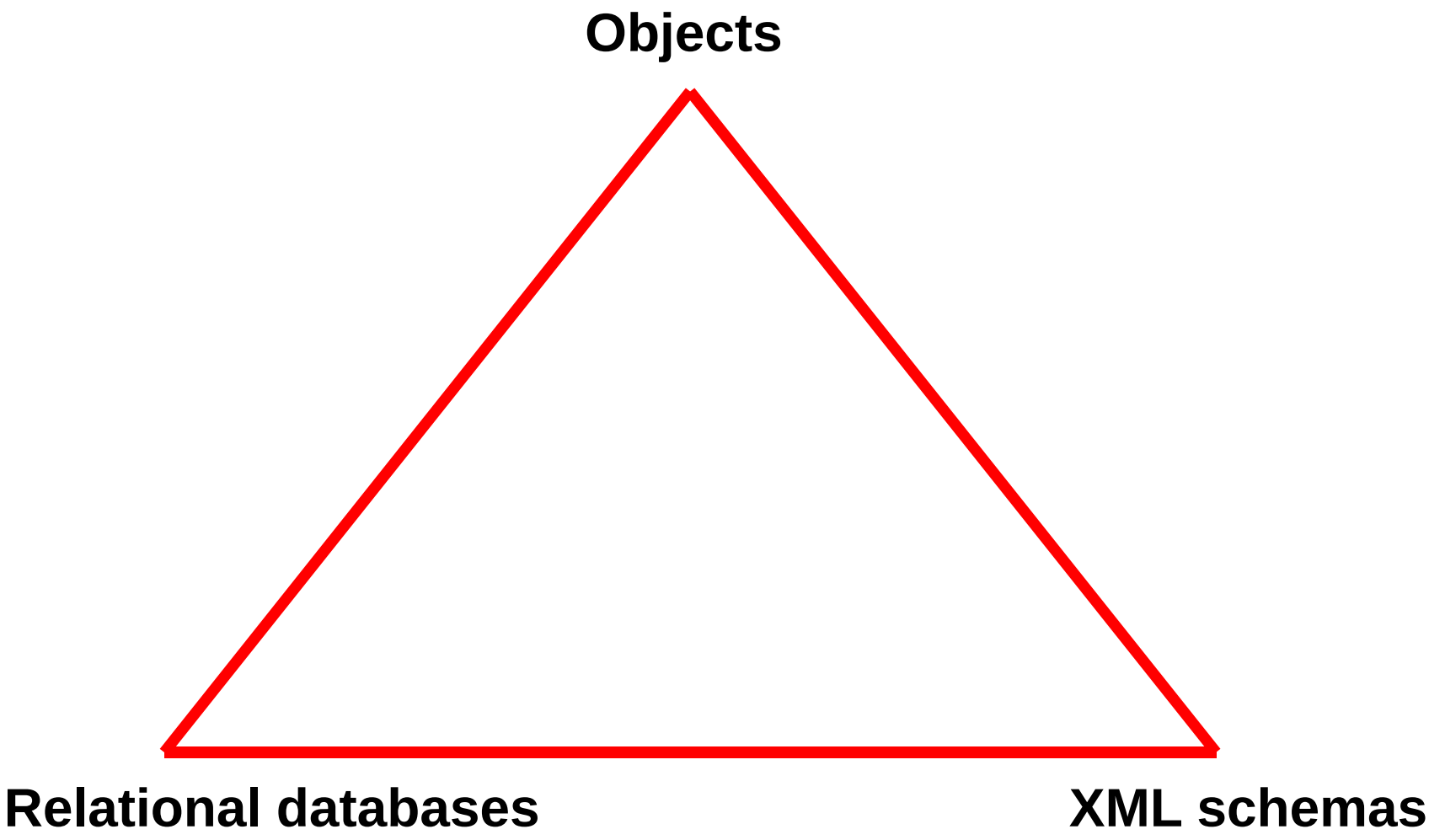
It provides a set of guidelines for the structuring of specifications, which are expressed as models.

Model-driven architecture is a kind of domain engineering, and supports model-driven engineering of software systems.

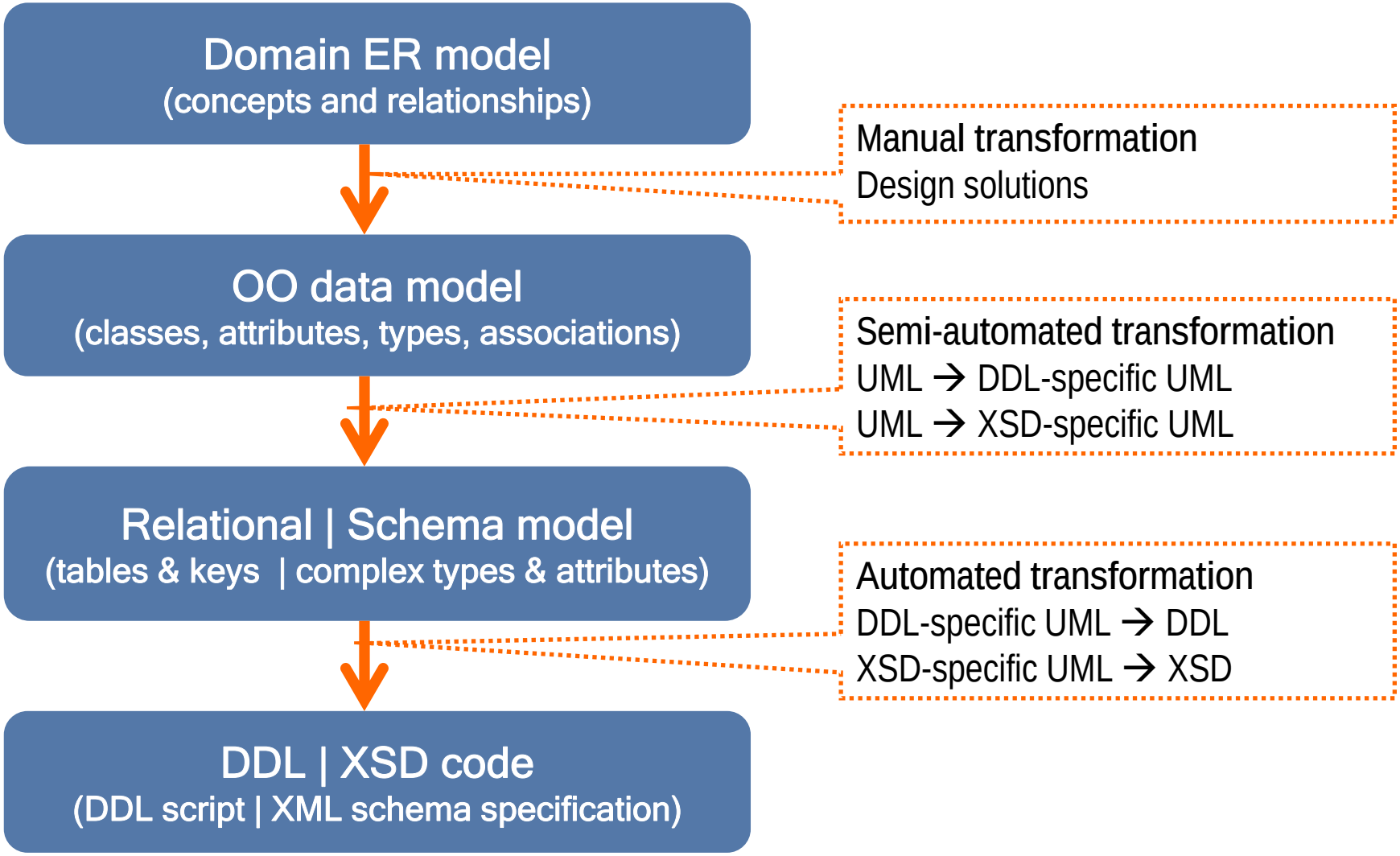
It was launched by the Object Management Group (OMG) in 2001.

→ http://en.wikipedia.org/wiki/Model-driven_architecture

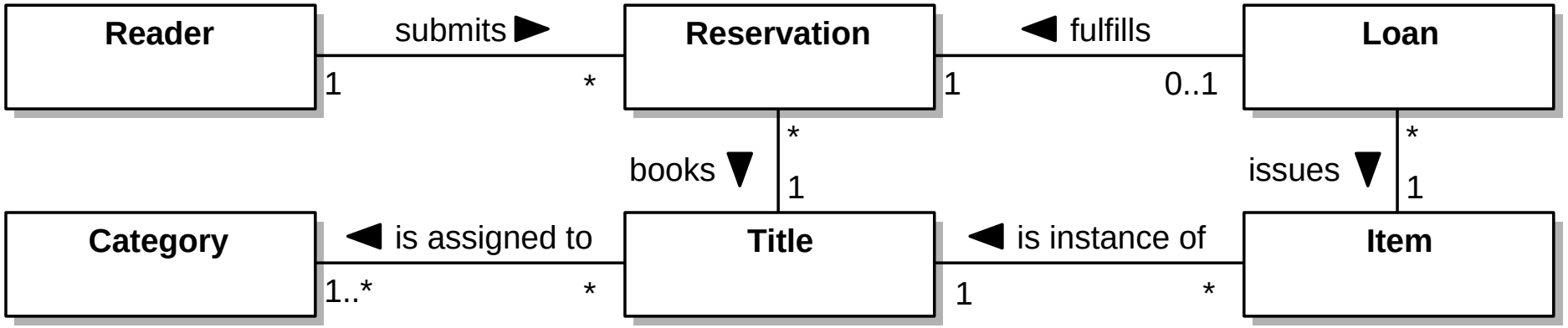
# Data Modeler’s Bermuda Triangle



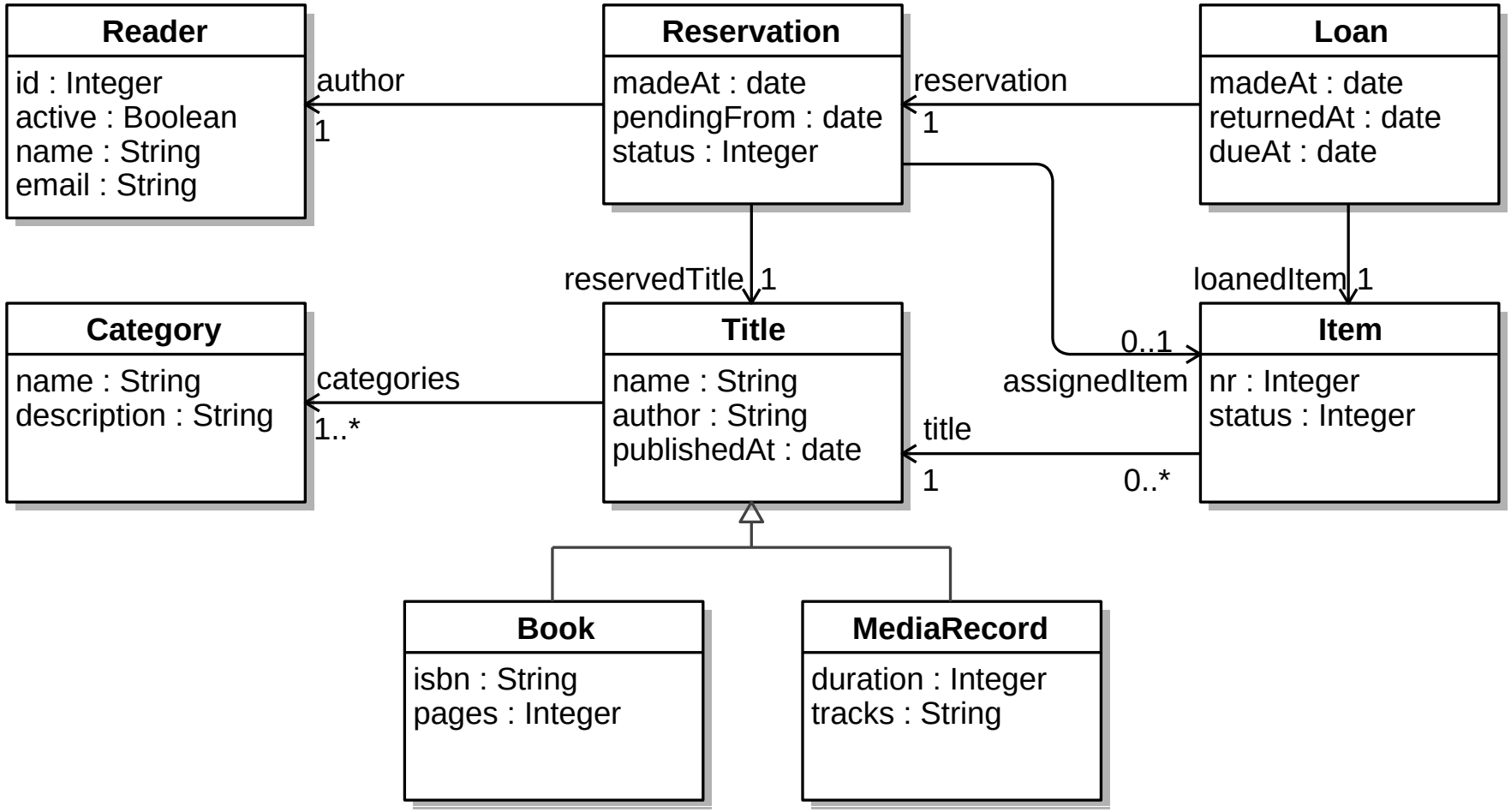
# Applying MDA to Data Design



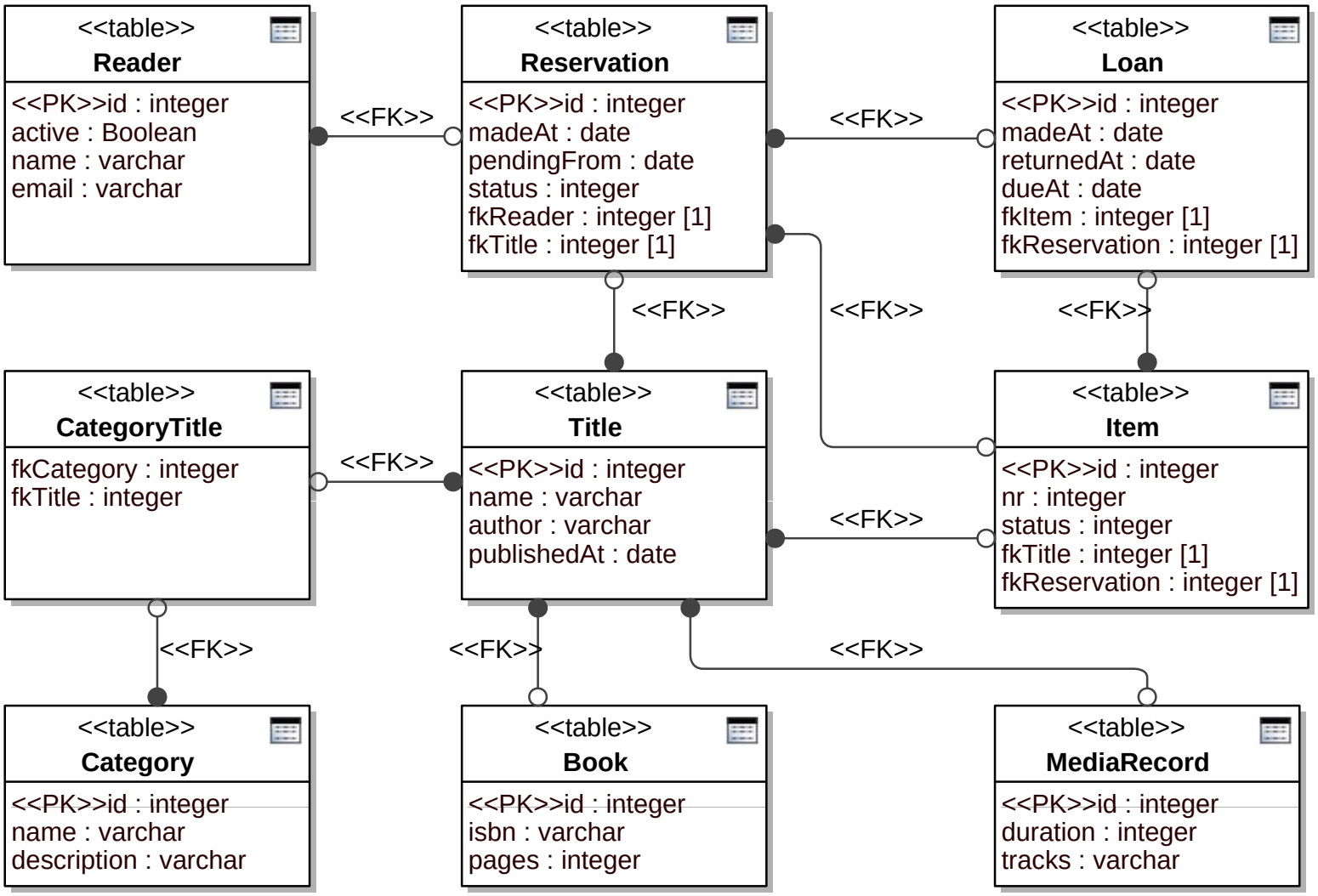
# Case Study: Domain ER Model



# Case Study: Object-Oriented Data Model



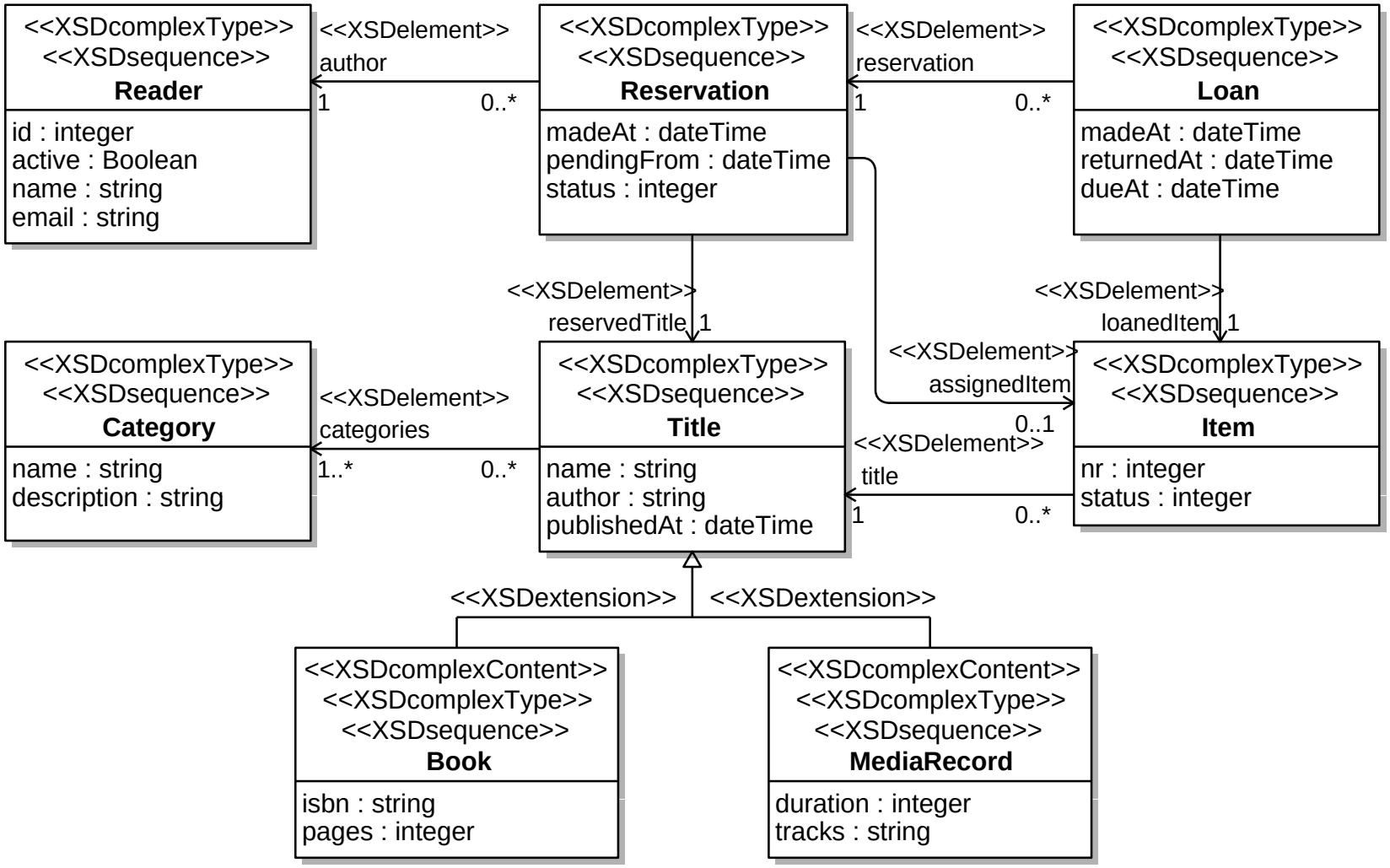
# Case Study: DDL-Specific Model



# Case Study: DDL Script (Fragment)

```
...
CREATE TABLE Reservation (
  id integer PRIMARY KEY,
  madeAt date,
  pendingFrom date,
  status integer,
  fkReader integer NOT NULL,
  fkTitle integer NOT NULL,
  FOREIGN KEY(fkTitle) REFERENCES Title(id),
  FOREIGN KEY(fkReader) REFERENCES Reader(id));
...
```

# Case Study: XSD-Specific Model



# Case Study: XML Schema (Fragment)

```
...
<xs:complexType name="Reservation">
  <xs:sequence>
  <xs:element maxOccurs="1" name="author" type="Reader"/>
  <xs:element maxOccurs="1" name="reservedTitle" type="Title"/>
  <xs:element maxOccurs="1" minOccurs="0" name="assignedItem" type="Item"/>
  </xs:sequence>
  <xs:attribute name="madeAt" type="xs:dateTime"/>
  <xs:attribute name="pendingFrom" type="xs:dateTime"/>
  <xs:attribute name="status" type="xs:integer"/>
</xs:complexType>
...
```

# 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**