

# Modeling the Content of Adaptive Web-Based System Using an Ontology

**Mária Bieliková and Michal Moravčík**

Institute of Informatics and Software Engineering  
Faculty of Informatics and Information Technologies  
Slovak University of Technology in Bratislava

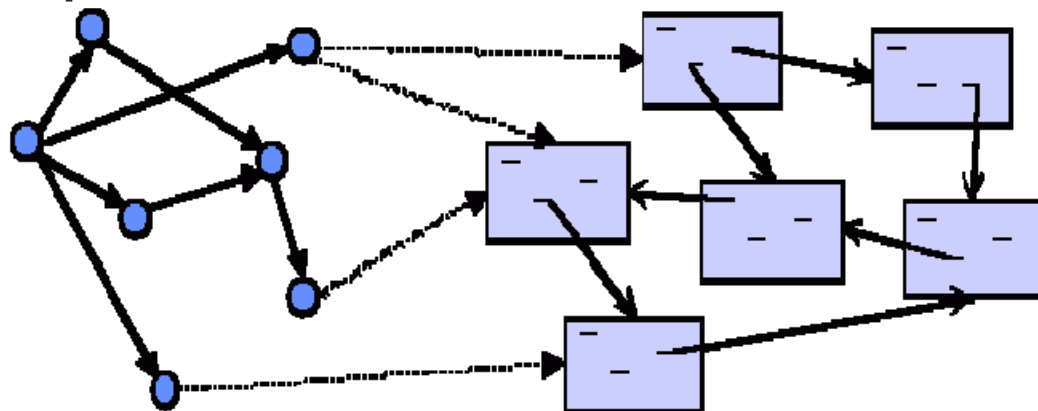
`bielik@fiit.stuba.sk`

# Motivation and Goals

- Effectiveness of creating tools for e-learning
  - software tools development together with corresponding infrastructure
  - content creation and maintenance
- **The goal is authoring once, delivering many**
- Developments of content models usable for many applications
  - application domain model (concepts)
  - information space (information fragments)
  - user model

# Process of Content Development

- Design and structuring of knowledge space
- Design of generic user model
- Design a set of goals
- Design and structuring of information space
- Interconnection of knowledge space with information space



# Content Representation

## ■ Relational database

- information content represented as a set of interconnected tables
- concept property = attribute in the relational data model
- good performance, low reusability

## ■ XML-based language

- content represented as values or attributes of specific tags
- better flexibility
- only general syntax without formally defined semantics  
→ difficulties with reasoning

# Content Representation (cont.)

## ■ **Ontology**

- an explicit shared specification of the conceptualization of a domain
- RDF/OWL takes XML representation (syntax) and eliminates its disadvantage by defining a vocabulary for describing properties and classes

## ■ **Advantages of using ontology**

- sharing (common understanding of domain)
- reusability of models
- reasoning

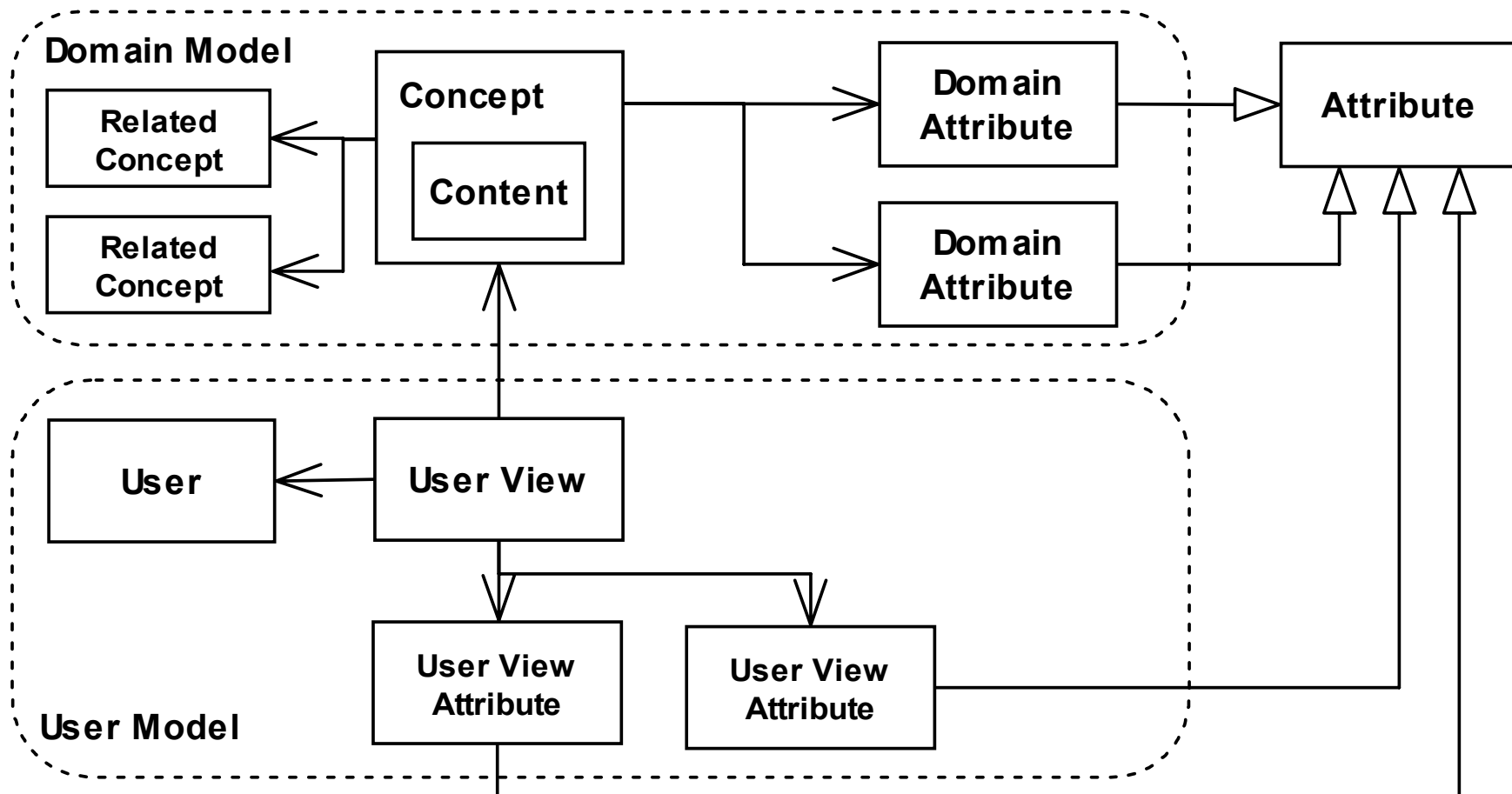
## ■ **Disadvantages of using ontology**

- non-uniform list of ontologies, various vocabularies
- difficult to model adaptation knowledge
- complexity of transformation ontology into the existing models
- immaturity of tools for creating and manipulation with ontologies

# Content Sharing

- Bottleneck of content-oriented web-based systems
  - creating a schema of the domain and instances of it
- How to share the content
  - commonly accepted model
  - designing converters between various representations
    - Interbook → AHA!
    - MOT (My Online Teacher) → AHA!, WHRULE
  - defining intermediate format

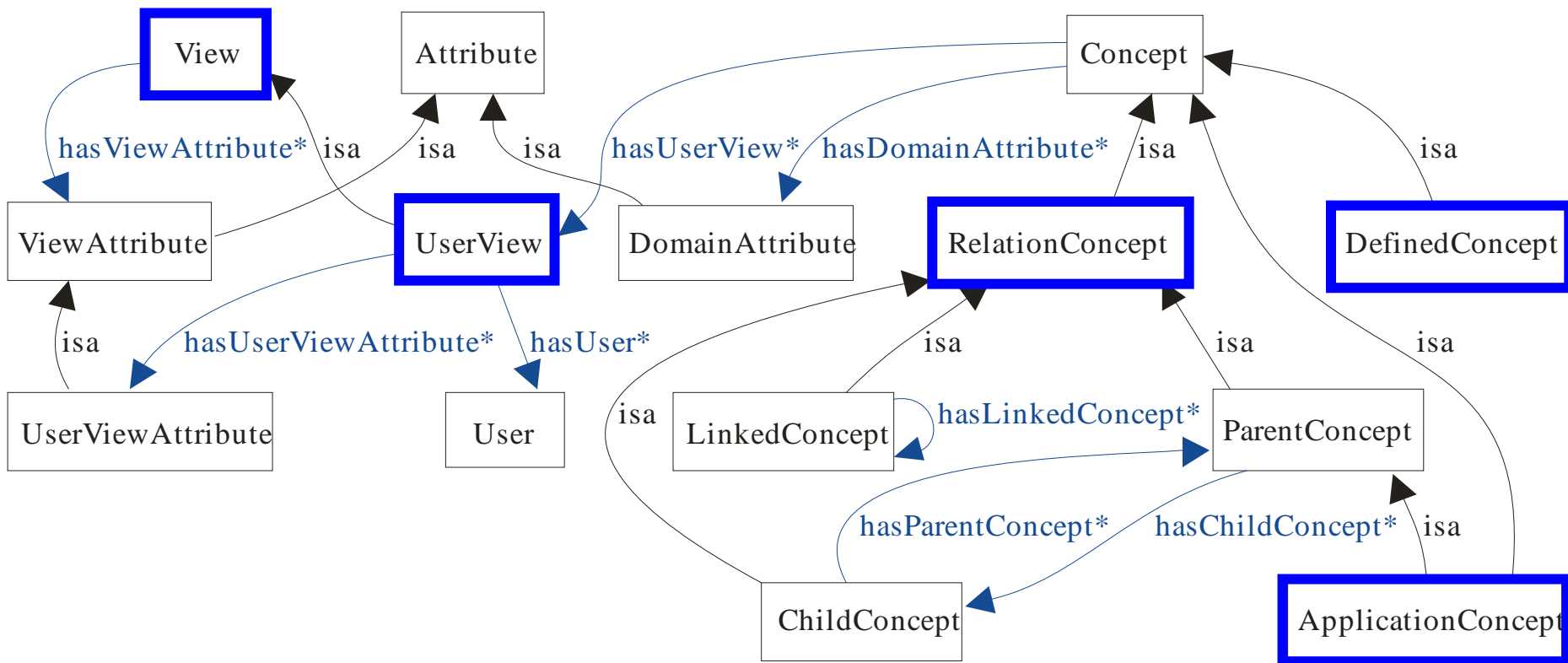
# Domain Model Schema



# Method for Modeling the Content

- 1. Specify the concept classes and the concept class hierarchy**
- 2. Define attributes for the concept classes**
  - data attributes
  - domain attributes
  - user attributes
- 3. Specify and type relations between concepts of specified classes**
- 4. Create concept instances as the class instances of the ontology**
- 5. Deliver the ontology content into existing adaptive system**

# Core Ontology of the Content Model

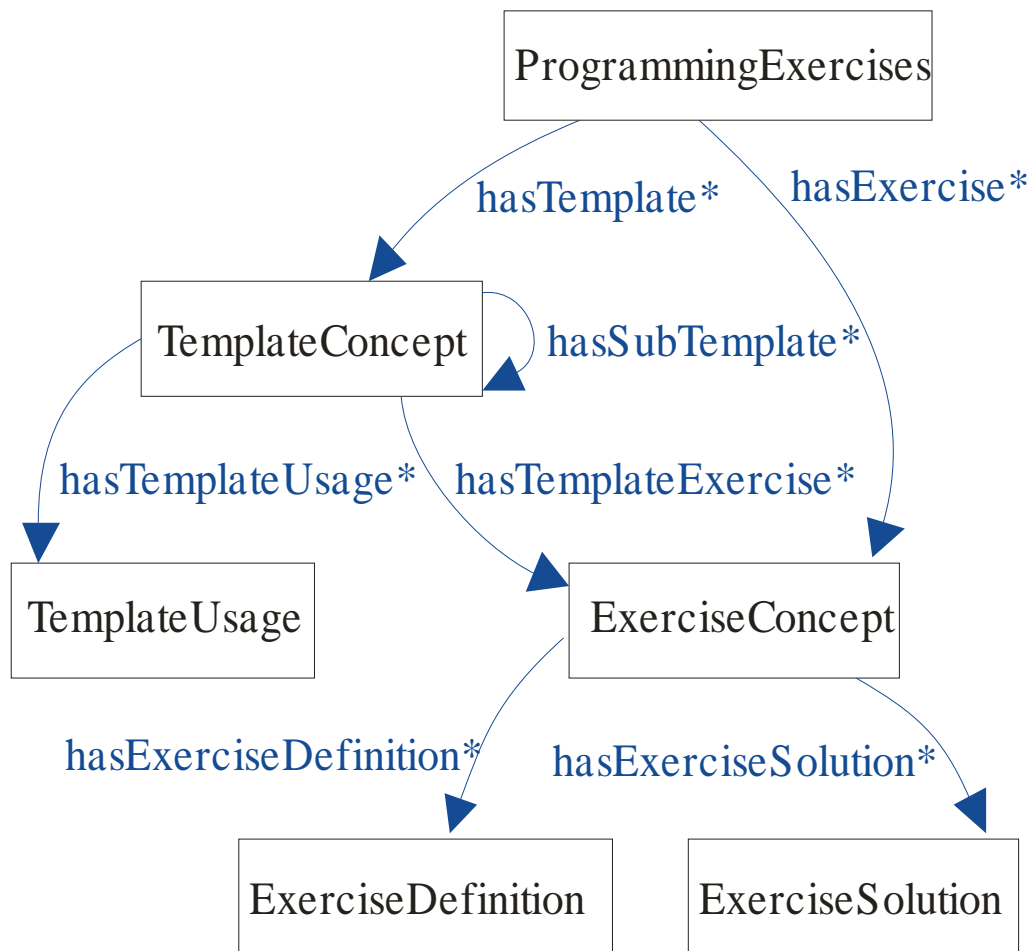


# Experiments with Content Transformation

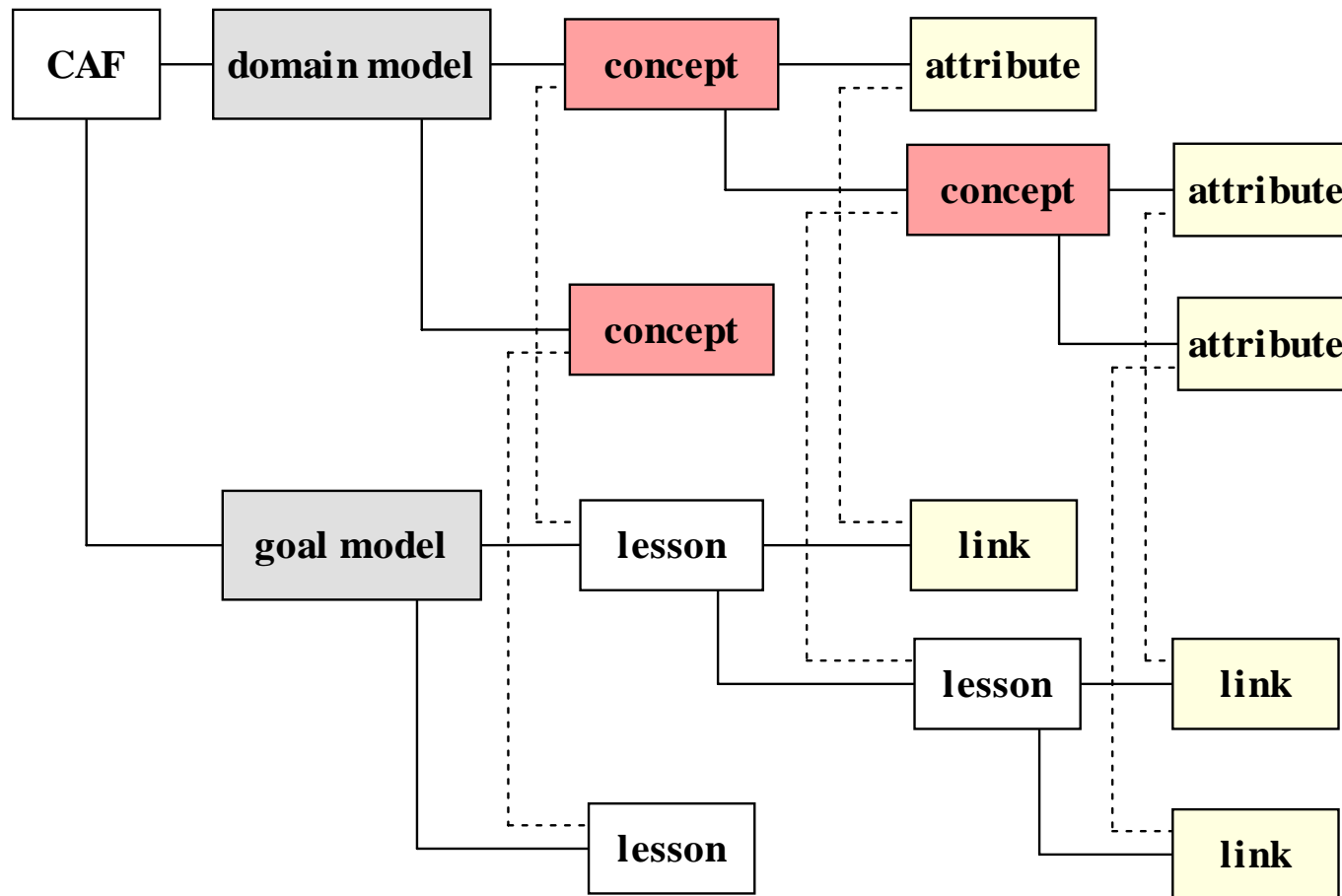
- Development of an ontology for the educational course of programming (Lisp and Prolog using program schemata and exercises)
- Filling ontology - instances
- Transformation of the content into existing adaptive hypermedia system (AHA!)
- Software support
  - ontology creation and editing
  - import of existing ontology
  - export of ontology into intermediate format (CAF, CAFE)



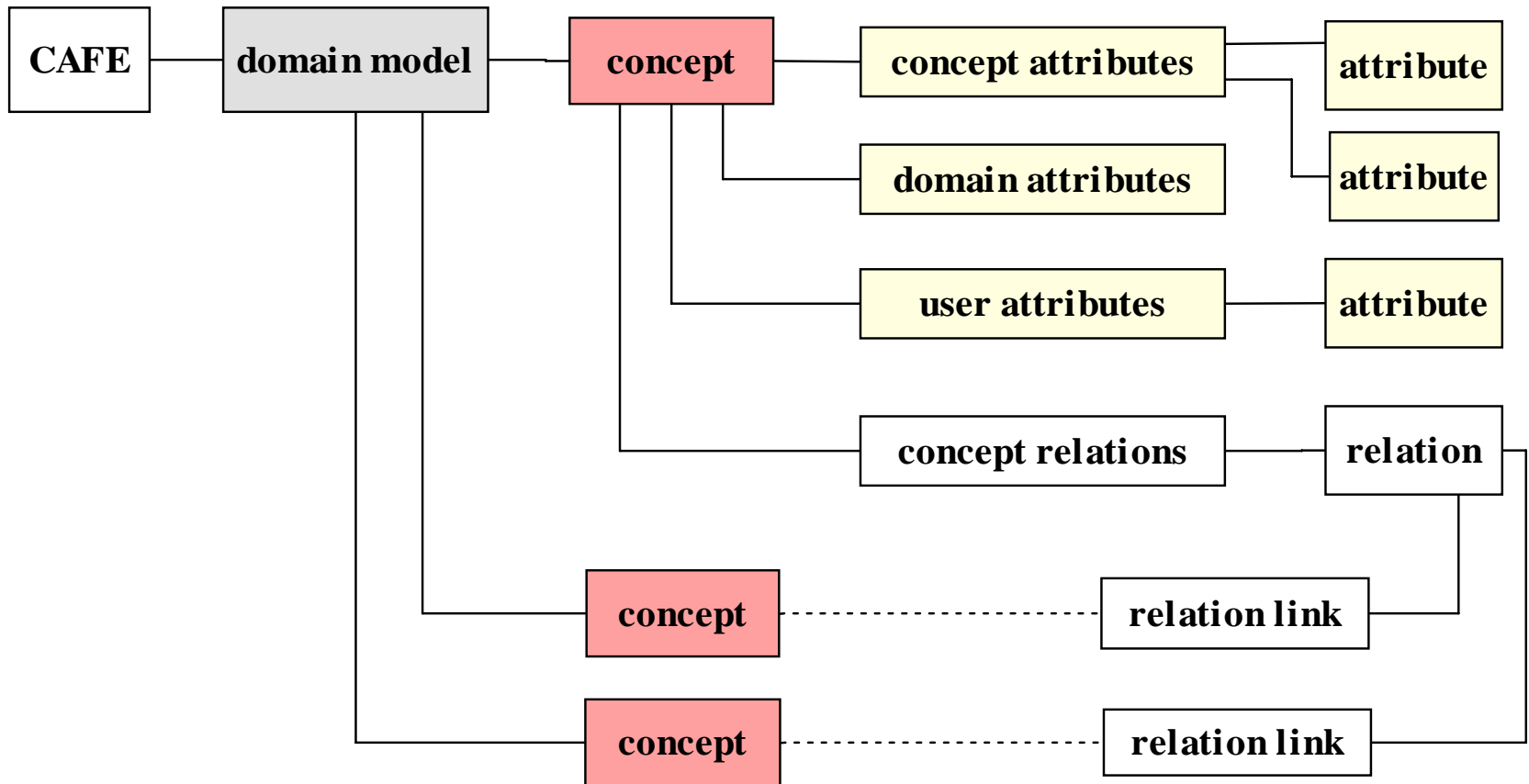
# Programming Course Domain Model



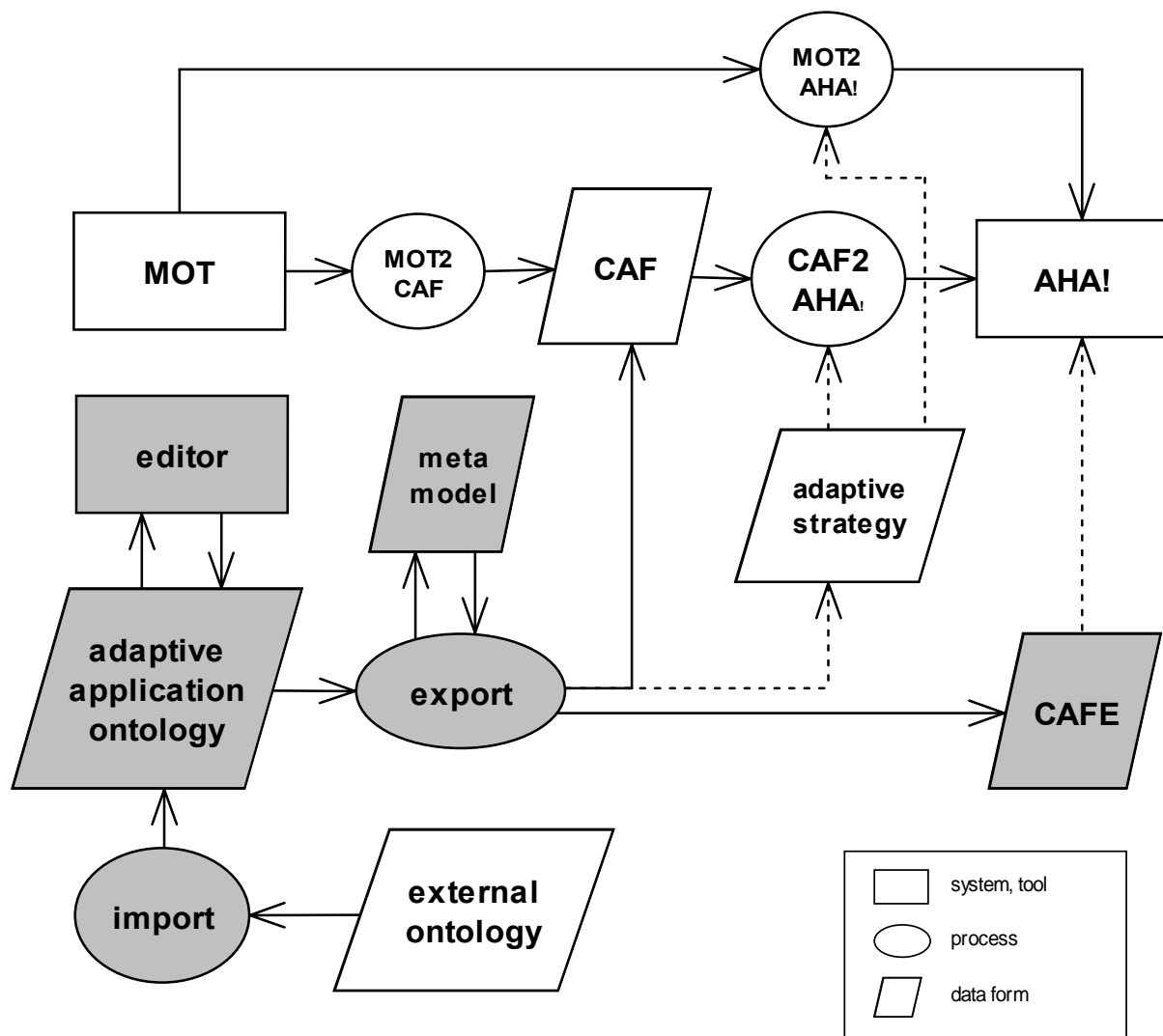
# Common Adaptation Format (CAF)



# CAFE (CAF Extended)



# Experiments with Content Transformation



# Ontology Editor

The screenshot displays the 'Adaptive Application Ontology Editor (exercises.owl)' interface. It features a menu bar (File, Import, Export, Show, About) and a toolbar with icons for adding, deleting, and searching. The main workspace is divided into several panels:

- Concept classes:** A tree view showing a hierarchy starting with 'DefinedConcept', followed by 'EducationalConcept', 'ExerciseConcept', 'ProgrammingExercises', and 'TemplateConcept'. Below this, 'ExerciseDefinition', 'ExerciseSolution', and 'TemplateUsage' are listed. A blue label 'hierarchy of concept classes' is overlaid on this panel.
- Concept instances:** A list of instances including 'member', 'LispA1207', 'LispA1418', 'PrikladDelete', 'PrikladDel\_Nuly', 'PrikladDruha\_Mocnina', 'PrikladSucin', 'PrikladPorovnaj\_Vektory', and 'PrikladPreklad'. A blue label 'instances' is overlaid on this panel.
- Concept attributes:** A table listing attributes for 'ExerciseConcept':

conceptText	[multiple string]
conceptDescription	[single string]
exerciseHint	[multiple string]
conceptName	[single string]

A blue label 'concept attributes' is overlaid on this panel.
- Domain attributes:** A table listing domain attributes:

conceptWeight	[single int]
exerciseDifficulty	=0 [single int]
exerciseType	=default [multiple string]

A blue label 'domain attributes' is overlaid on this panel.
- User attributes:** A table listing user attributes:

educationalInterest	=50 [single int]
conceptVisited	=false [single boolean]
educationalKnowledge	=0 [single int]
exerciseSolved	=false [single boolean]

A blue label 'user attributes' is overlaid on this panel.
- Related concepts:** A table showing relationships:

ExerciseSolution	hasExerciseSolution	isSolutionOfExercise	[multiple fragment]
TemplateConcept	hasExerciseTemplate	hasTemplateExercise	[multiple parent]
ExerciseDefinition	hasExerciseDefinition	isDefinitionOfExercise	[multiple fragment]

A blue label 'relations to other concept classes' is overlaid on this panel.

# Transformed Content in AHA!

http://localhost:8080/aha/Get?concept=exercises.189

- Príklady programovania v jazyku Lisp
  - Prolog
    - Na najvyššej úrovni
      - Schema prechádzanie prvkov
        - Schema Filter
          - Príklad delete
            - Riešenie
            - Definicia
            - Príklad del\_nuly
            - Použitie schemy

Prepíšme si najprv obe vetvy bez definovania samotných testov.

*Predikat je podrobne vysvetlený v učebnici [[Bielikova Maria, Navrat Pavol: Funkcionálne a logické programovanie](#)].*

```
delete(P, [H | T1], [H | T2]) :-  
<element_test>(H, P),  
delete(P, T1, T2).  
delete(P, [H | T1], T2) :-  
<not_element_test>(H, P),  
delete(P, T1, T2).
```

*Riešenie je podobné ako predikat z príkladu [ODKAZ:PríkladDel\_Nuly%del\_nuly].*

Predikat *delete(+Prvok, +Zoznam, ?Vysledok)* sa splní, ak zoznam *Vysledok* obsahuje všetky prvky zoznamu *Zoznam* bez výskytov prvého argumentu.

Glossary  
Content

- Riešenie
- Príklad LispA1207
- Riešenie
- Definicia
- Použitie schemy
- Príklad LispA1418
- Riešenie
- Definicia
- Použitie schemy
- Schema Hľadaj
- Príklad member
- Riešenie
- Definicia
- Použitie schemy
- Riešenie
- Definicia
- Príklad del\_nuly
- Riešenie
- Definicia
- Použitie schemy
- Schemy
- Príklad preklad
- Riešenie
- Definicia
- Príklad

# Conclusions

- Modeling using a metaphor “authoring once, delivering many”
- Use of ontology as knowledge representation in adaptive web-based applications
  - it allows building both closed and open corpus systems using the same processes
  - development of adaptive application domain model – explicit definition of semantics
  - sharing and reuse of the content and structure of adaptive applications
  - utilization of existing ontology as a domain model
- Variability of ontology structure without the loss of ontology data

# Future Work

- Integration of our software solution into “standard” solutions (e.g., Protégé)
- Transformation of domain model application into other adaptive systems
- Standardization of the CAFE format for transformation of adaptive applications between adaptive web-based systems
- Creation of web services for sharing the ontology
- Application of new possibilities of adaptation based on ontology reasoning