MESAM : A Protégé Plug-in for the Specialization of Models

Nadjet Zemirline, Yolaine Bourda, Chantal Reynaud and Fabrice Popineau

Supelec Gif-sur-Yvette, LRI Paris-Sud university and Inria Saclay - France

June 26th, 2008
Reuse generic platforms to create new systems

Usually, the creator

1. comes with his own models and resources
2. would like to reuse the generic system over his resources

→ integrate his models and resources in the system

There are some problems, the creator

- needs to translate his models into the specific format understood by the system
- needs to translate all the instantiations of his models
- ...

→ tedious and time-consuming

The goal is to allow the creator to reuse his models and his models instantiations without any change of format or vocabulary
Authoring Adaptive Hypermedia Systems using generic platforms (1)

An adaptive Hypermedia System (AHS) can adapt its behavior to the needs of individual users

Architecture of an AHS ...

How to author AHS??
Usually, the creator comes with his user and domain models. He can reuse an adaptation engine, if he

- translates his resources in a format understood by the adaptation engine
- integrates (without translation) his resources in the platform

**What do we mean by integration??**
Specialization of generic models

Two kinds of approaches
1. bottom-up approaches (based on instances): FCA-MERGE
2. top-down approaches (based on concepts): ODE-MERGE, PROMPT

Specificities of the specialization process in the context of authoring systems
1. we have..
   - Small models
   - The creator has a very good understanding of his models
2. we need..
   - Mappings defined between generic and specific elements with a very high precision (100 % if possible)
   - ...

Traditional approaches can not be applied → Need new solutions !!
Agenda

MESAM Plug-in
  Approach, new specialization process
  Functionality

MESAM Plug-in Implementation
  MESAM plug-in Architecture
  Interaction with MESAM Plug-in

Conclusion
..
New specialization process

Our proposal

*a semi-automatic approach helping a creator in reusing his models*

Main aspects of this approach ...

- A semi-automatic process
- An help to support the definition of mappings between generic and specific models
- An help to create consistent and relevant models integrating generic and specific models
Step 1: Specification of mappings between classes

**Specification of mappings between classes**

Two kinds of mappings can be specified:

1. **Generic model**
   - Class A
   - Class B

2. **Specific model**
   - Class C
   - Class D

**Specialization mapping**

**Equivalence mapping**

**Creator**

- Computation of additional mappings between classes
- Structural verification
- Validation of suggestions

**Automatic Deductions**
Step 2: computational of additional mappings between classes

Using patterns

Deduction of additional mappings based on the composition properties of $R_{equivalentClass}$ and $R_{subClass}$

1. $R_{equivalentClass} \circ R_{equivalentClass} = R_{equivalentClass}$
2. $R_{equivalentClass} \circ R_{subClass} = R_{subClass}$
3. $R_{subClass} \circ R_{equivalentClass} = R_{subClass}$
4. $R_{subClass} \circ R_{subClass} = R_{subClass}$

Creator

Structural verification

Validation of suggestions
Step 3: Structural verification (1)

Two goals:

1. automatically deduce mappings between properties
2. check the consistency of the new model created by the merging process

→ exploitation of structural knowledge, as: domain, range, cardinalities ...
Exploitation of structural knowledge is represented in a OWL meta-model:

- some parts of the OWL meta-model were reused
- new primitives were added (to represent the needed structural knowledge)

→ a generic solution (usable for any model expressed in OWL)
Step 3: Structural verification (3)

Deduction made using SWRL rules based on the meta-model
Step 4: Validation of suggestions

1. A creator specifies that ..
   - C is sub-class of A
   - D is sub-class of B

2. The system deduces that ..
   - P2 is probably a sub-property of P1
   - P3 is probably an equivalent property of P1
MESAM plug-in architecture

Architecture includes two parts : A knowledge part, a processing part ..

Some characteristics ..

- implemented as a Protege plug-in
- first tests were done in e-learning domain
MESAM workspace

1 – specification of the generic and specific models

2 – ask to load models
MESAM workspace
MESAM workspace

4 - specification of mappings between classes

5 - run structural deduction
MESAM workspace

6 – mappings: deductions between properties
7 – inconsistency deductions
8 – no deductions for properties of the generic model
9 – save inconsistency problems and missing mappings
10 – save the merging results in a new model
Conclusion and Future work

Results

New approach for merging models, characterized by the
- **8 patterns**
- expression of constraints by **25 rules**
- definition of a generic solution, based on a *modified version of the OWL meta-model*
- precision of results

Future works

Thinking about several extensions for the plug-in, as
- consideration of inconsistency problems and helping the creator in resolving them
- consideration of other kinds of mappings than equivalence and specialization, like disjunctions
- *for AHS*, help to personalize adaptation strategies supported by specialized domain and user models
Thank you! Questions?