Representation and Management of Reified Relationships in Protégé

Tania Tudorache
7th International Protégé Conference
Bethesda, 9. July 2004

Overview

- From simple to reified relationships
- Problems
- Patterns for modeling reified relationships
- Tool Support
The context:
From Simple to Reified Relations

A department has employees.

In a department, each employee has a hiring date and a certain type of contract.

If a department has an employee, then the employee works for that department.

What is a reified relation?

- **Reification:**
  - *Def.*: Treating an abstract concept as if it were a real, concrete thing
  - Knowledge representation:
    - used to represent facts that must then be manipulated in some way
  - RDF(S):
    - a statement attached as a property to another statement

Source: http://www.wordiq.com/definition/Reification
A Simple Scenario: Building departments

Department

Department-Employee-Rel
- hiringDate
- contractType

Employee

IT Department

ITDepartment-CS-Rel
- hiringDate
- contractType

Computer Scientist

Software Department

SWDepartment-Programmer-Rel
- hiringDate
- contractType

Programmer

Tania Tudorache
Representation and Management of Reified Relationships in Protégé

A Simple Scenario: Screenshot of Protégé GUI

Tania Tudorache
Representation and Management of Reified Relationships in Protégé
A Simple Scenario: Building departments

Tania Tudorache

A Simple Scenario: Changing the class taxonomy

Tania Tudorache
A Simple Scenario
... becoming consistent

Another Simple Scenario:
Deleting a related instance

What should happen at deletion?
- Delete incomplete relation instances?
- Delete incomplete relation instances and related instances?
- Deletion should not be allowed?
The directed binary relationship – Extending the Modeling Pattern

Software Department

hasEmployees

Department-Employee Relation

- hiringDate
- contractType

:TO

Programmer

Software Methods and Tools

hasEmployees

SM-John Relationship

- hiringDate: 01.03.2004
- contractType: permanent

:TO

John Foo Bar

Tania Tudorache

Representation and Management of Reflected Relationships in Protégé
The bidirectional inverse relationship - The modeling pattern

Software Department

Department-Employee-Relation

- hiringDate
- contractType

Sublot of :TERM1

Sublot of :TERM2

TERM1-Dep

Inverse slots

hasEmployees

TERM2-Dep

Inverses

SM-John Relationship

- hiringDate: 01.03.2004
- contractType: permanent

worksFor

John Foo Bar

TERM2

Programmer

hasEmployees

TERM1

Software Methods and Tools

Automatic filled in

Automatically filled in

User Filled-in

Relationship instance
What tool support is needed?

- **Modeling patterns for other types of relations**
  - Bidirectional inverse relation
  - N-ary relationship

- **Definition support at class level**
  - Automations that create all the necessary structures and fill them in correctly
  - Specialized widgets that support the definition of reified relations

- **Usage support at instance level**
  - Visualizations and editing plug-ins

- **Consistency check**

- **API Support for reified relationships**

Conclusions

- Reified relationships add more expressiveness to your ontology

  - ... but:
    - They are hard to manage
    - They involve extra effort in keeping consistent

  - That’s why we need:
    - Automation support for repetitive operations (e.g. creation, deletion, etc.)
    - Specialized GUI-s