DAIMLERCHRYSLER

Representation and Management of Reified Relationships in Protégé

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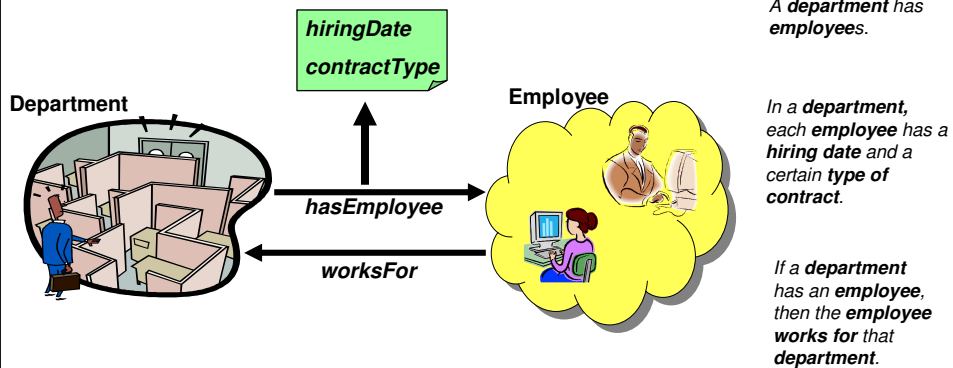Bethseda, 9. July 2004

Overview
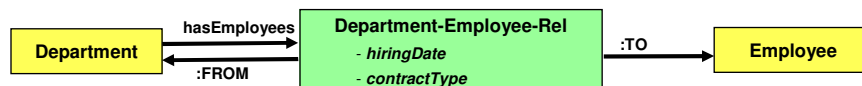
- From simple to reified relationships

- Problems

- Patterns for modeling reified relationships

- Tool Support

DAIMLERCHRYSLER

The context:
From Simple to Reified Relations

hiringDate
contractType

Department

Employee

hasEmployee

worksFor

A **department** has **employee**s.

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**.
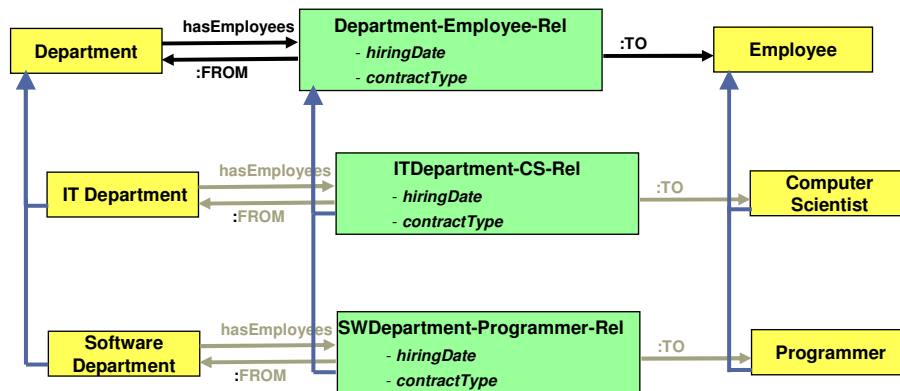

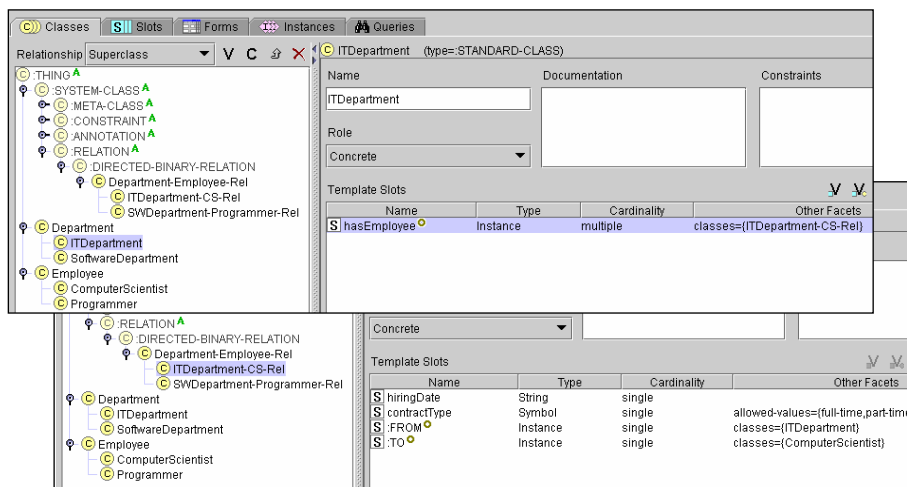**DAIMLERCHRYSLER**

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)**:
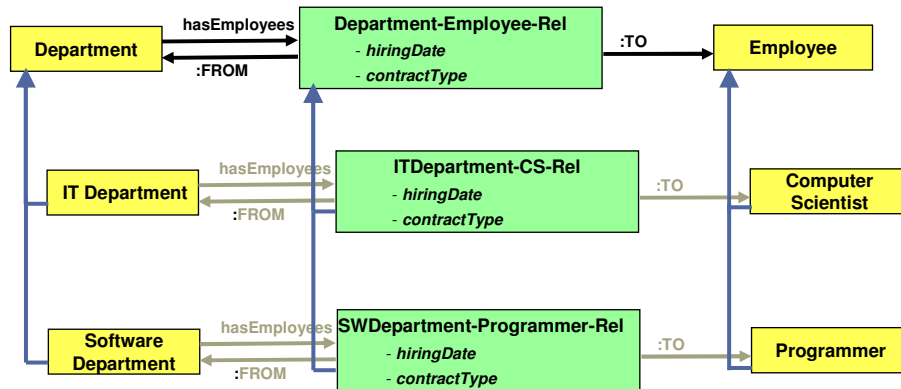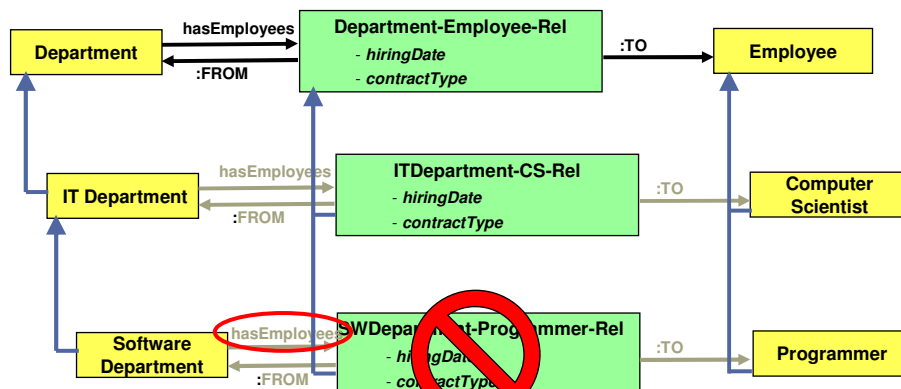   - a statement attached as a property to another statement

| Department | hasEmployees / :FROM | Department-Employee-Rel
- *hiringDate*
- *contractType* | :TO | Employee |

Source: http://www.wordiq.com/definition/Reification

A Simple Scenario:
Building departments

| Department | **hasEmployees** →
 ← **:FROM** | **Department-Employee-Rel**
 - *hiringDate*
 - *contractType* | **:TO** → | Employee |

| IT Department | **hasEmployees** →
 ← **:FROM** | **ITDepartment-CS-Rel**
 - *hiringDate*
 - *contractType* | **:TO** → | Computer Scientist |

| Software Department | **hasEmployees** →
 ← **:FROM** | **SWDepartment-Programmer-Rel**
 - *hiringDate*
 - *contractType* | **:TO** → | Programmer |

A Simple Scenario:
Screenshot of Protégé GUI

DAIMLERCHRYSLER

A Simple Scenario:
Building departments

| Department | —hasEmployees→ / ←:FROM— | Department-Employee-Rel
- *hiringDate*
- *contractType* | —:TO→ | Employee |

| IT Department | —hasEmployees→ / ←:FROM— | ITDepartment-CS-Rel
- *hiringDate*
- *contractType* | —:TO→ | Computer Scientist |

| Software Department | —hasEmployees→ / ←:FROM— | SWDepartment-Programmer-Rel
- *hiringDate*
- *contractType* | —:TO→ | Programmer |

DAIMLERCHRYSLER

A Simple Scenario:
Changing the class taxonomy

| Department | —hasEmployees→ / ←:FROM— | Department-Employee-Rel
- *hiringDate*
- *contractType* | —:TO→ | Employee |

| IT Department | —hasEmployees→ / ←:FROM— | ITDepartment-CS-Rel
- *hiringDate*
- *contractType* | —:TO→ | Computer Scientist |

| Software Department | —hasEmployees→ / ←:FROM— | SWDepartment-Programmer-Rel
- *hiringDate*
- *contractType* | —:TO→ | Programmer |

DAIMLERCHRYSLER

A Simple Scenario
… becoming consistent

Department → hasEmployees → **Department-Employee-Rel**
- *hiringDate*
- *contractType*
:FROM
:TO → **Employee**

IT Department → hasEmployees → **ITDepartment-CS-Rel**
- *hiringDate*
- *contractType*
:FROM
:TO → **Computer Scientist**

Software Department → hasEmployees → **SWDepartment-Programmer-Rel**
- *hiringDate*
- *contractType*
:FROM
:TO → **Programmer**

DAIMLERCHRYSLER
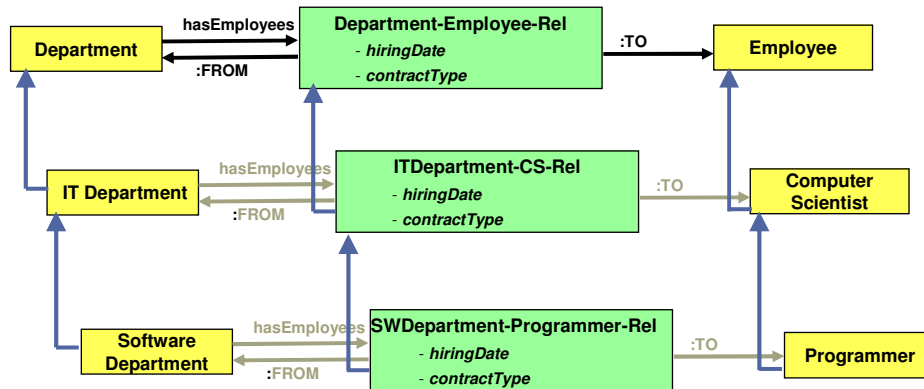
Another Simple Scenario:
Deleting a related instance

Software Department → hasEmployees → **Department-Employee-Rel**
- *hiringDate*
- *contractType*
:FROM
:TO → **Programmer**

instance of

instance of

instance of

Software Methods and Tools → hasEmployees → SM-John-Relationship
- *hiringDate: 01.03.2004*
- *contractType: permanent*
:FROM
:TO → John Foo Bar
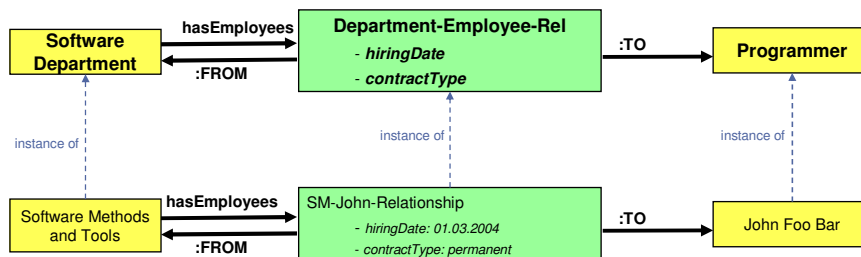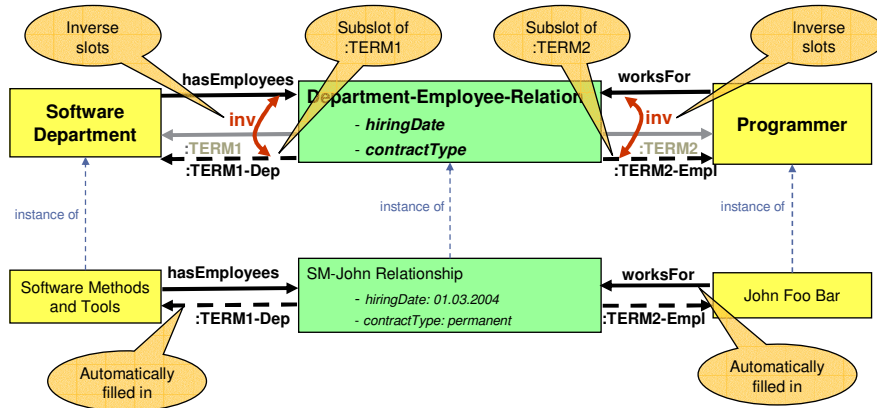
What should happen at deletion?
- Delete incomplete relation instances?
- Delete incomplete relation instances and related instances?
- Deletion should not be allowed?

The directed binary relationship –
Extending the Modeling Pattern
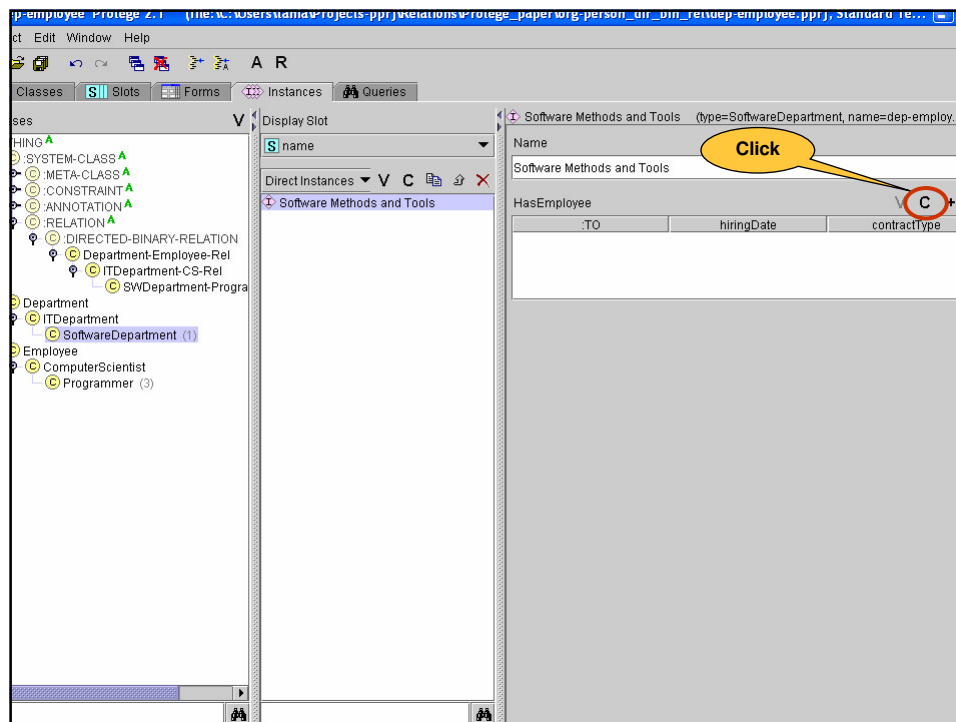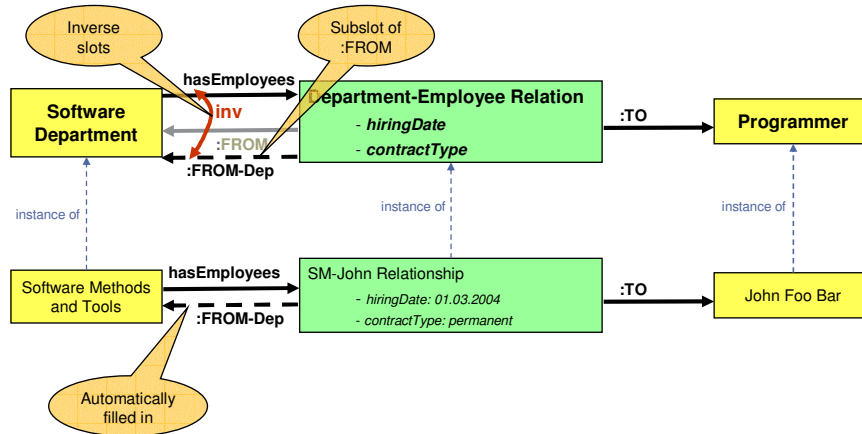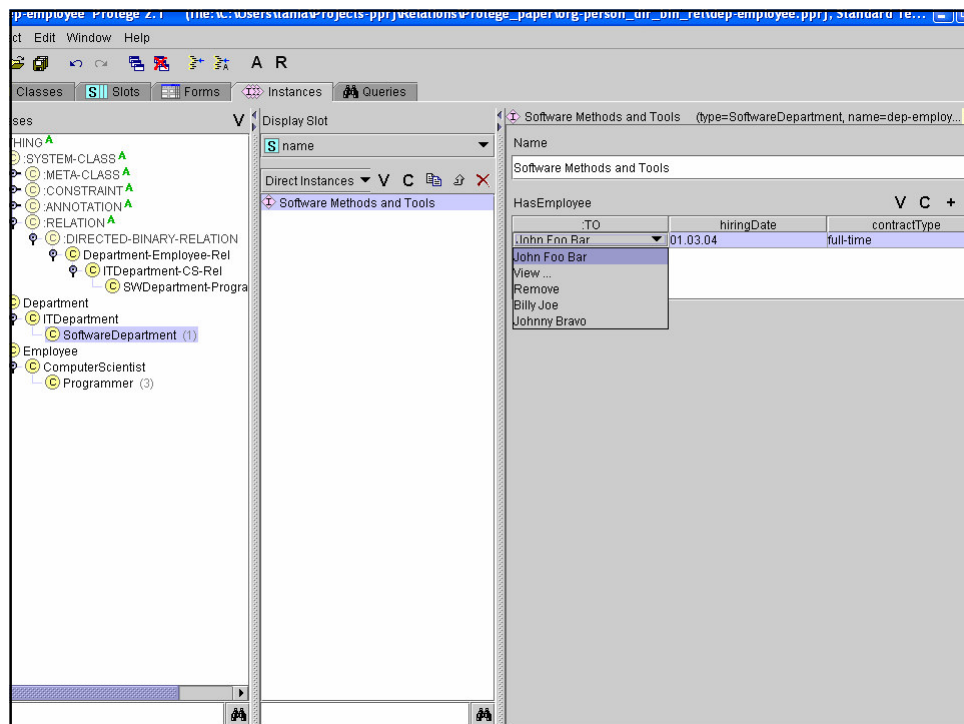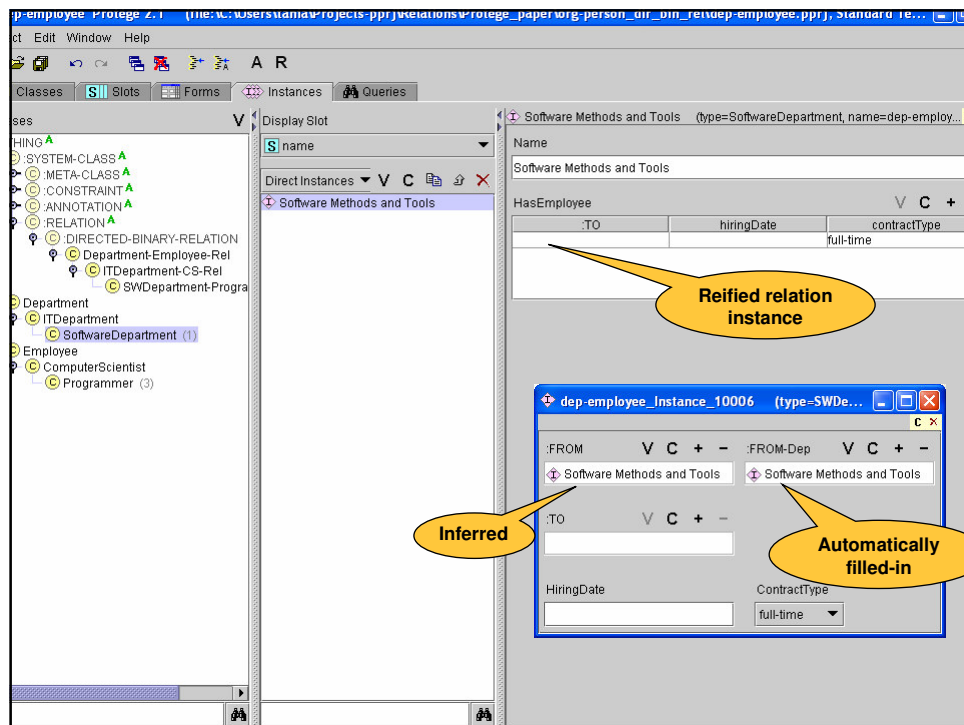
The bidirectional inverse relationship -
The modeling pattern

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