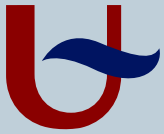
Copying Subgraphs within Model Repositories

Pieter van Gorp,
Hans Schippers,
Dirk Janssens

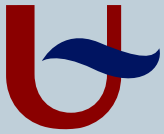
April 2, 2006

Universiteit Antwerpen

Contents



Outline



Context: Model Driven Software Engineering

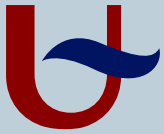
Model A simplified representation of a part of the world, named the system [Sei03]

Repositories Databases:

- serializing into standard formats (like XMI),
- exposing a query and transformation API (like OCL and JMI).

Challenge Manage consistency between and within models: check and transform

Platforms Matter Generative (more complex than interpretative, *cfr. ToolNet pres.*)

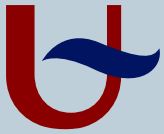


Context: Extending Graph Transformation

Why Graph Transformation? Ability to *model* a model transformation

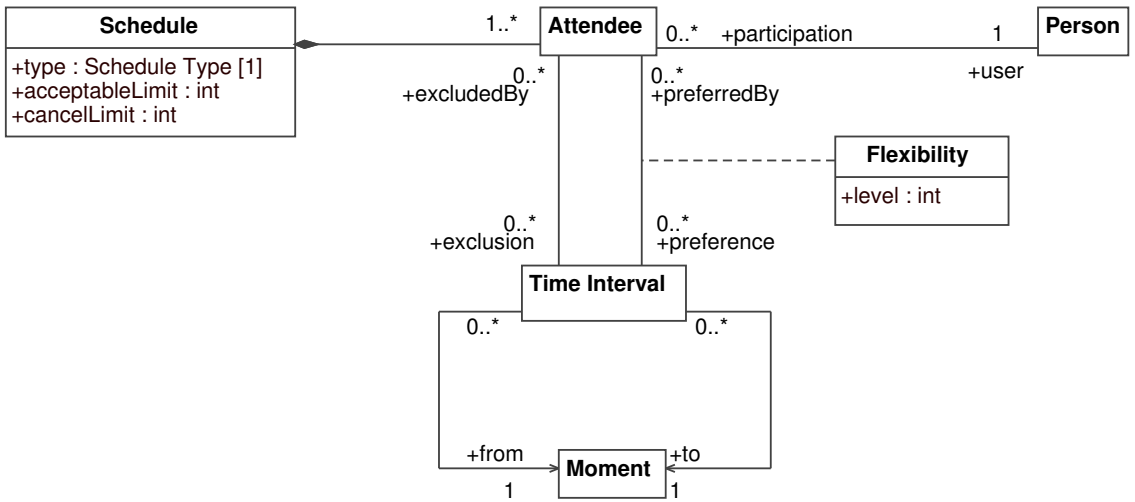
Research Goal Extend graph transformation:

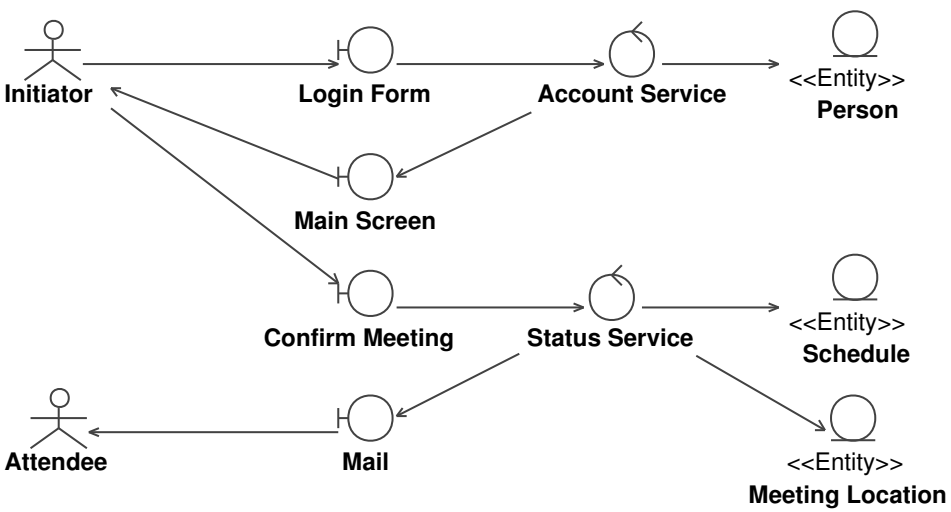
- model transformations can be programmed at a high level of abstraction, while
- the low-level APIs of mainstream model repositories are interfaced by means of compilers.
- ***not*** "just" interpreting the rules, execute them on a COTS modeling tool.



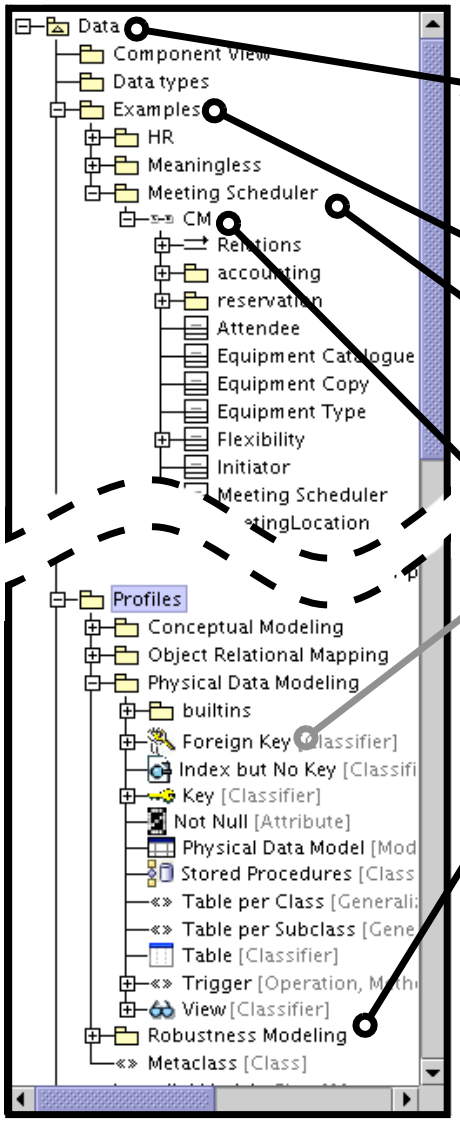
Outline

Conceptual and Robustness Model





Graphs? $\Longrightarrow$ Abstract Syntax



Node n1.
Type: Model
Attribute values:
name= "Data"
...
Links:
l1 (to n2, label "ownedElement")

Node n2.
Type: UmlPackage
Attribute values:
name= "Examples"
Links:
l2 (to n3, label "ownedElement")
l1 (to n1, label "namespace")

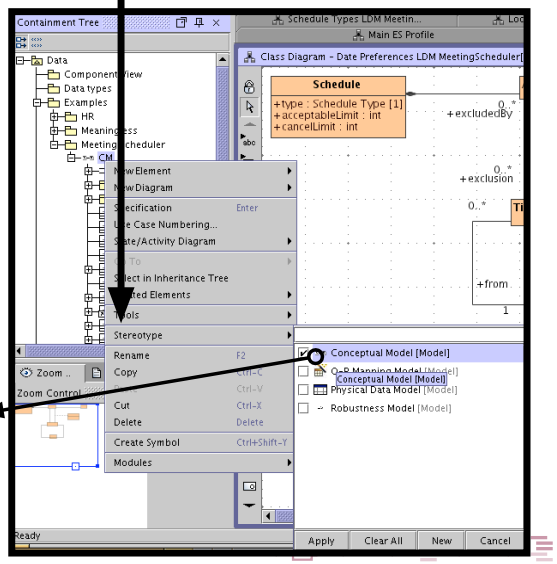
Node n3.
Type: UmlPackage
Attribute values:
name= "Meeting Scheduler"
Links:
l3 (to n4, label "ownedElement")
...

Node n4.
Type: Model
Attribute values:
name= "CM"
visibility= Public
Links:
l4 (to n7, label "stereotype")

Node n5.
Type: Stereotype
Attribute values:
name= "Foreign Key"
baseClass= "Classifier"
...

Node n6.
Type: UmlPackage
Attributes values:
name= "Robustness Modeling"
Links:
l5 (to n7, label
"ownedElement")

Node n7.
Type: Stereotype
Attributes values:
name= "Conceptual Model"
Links:
l4 (to n4, label "extendedElement")
l5 (to n6, label "namespace")



Case Study: Consistency Constraint

Each class in the conceptual model should correspond to an entity in the robustness model.

Constraint Violation Scenario only conceptual model has been developed, no robustness model yet

Repairing Transformation generate robustness model from the conceptual model

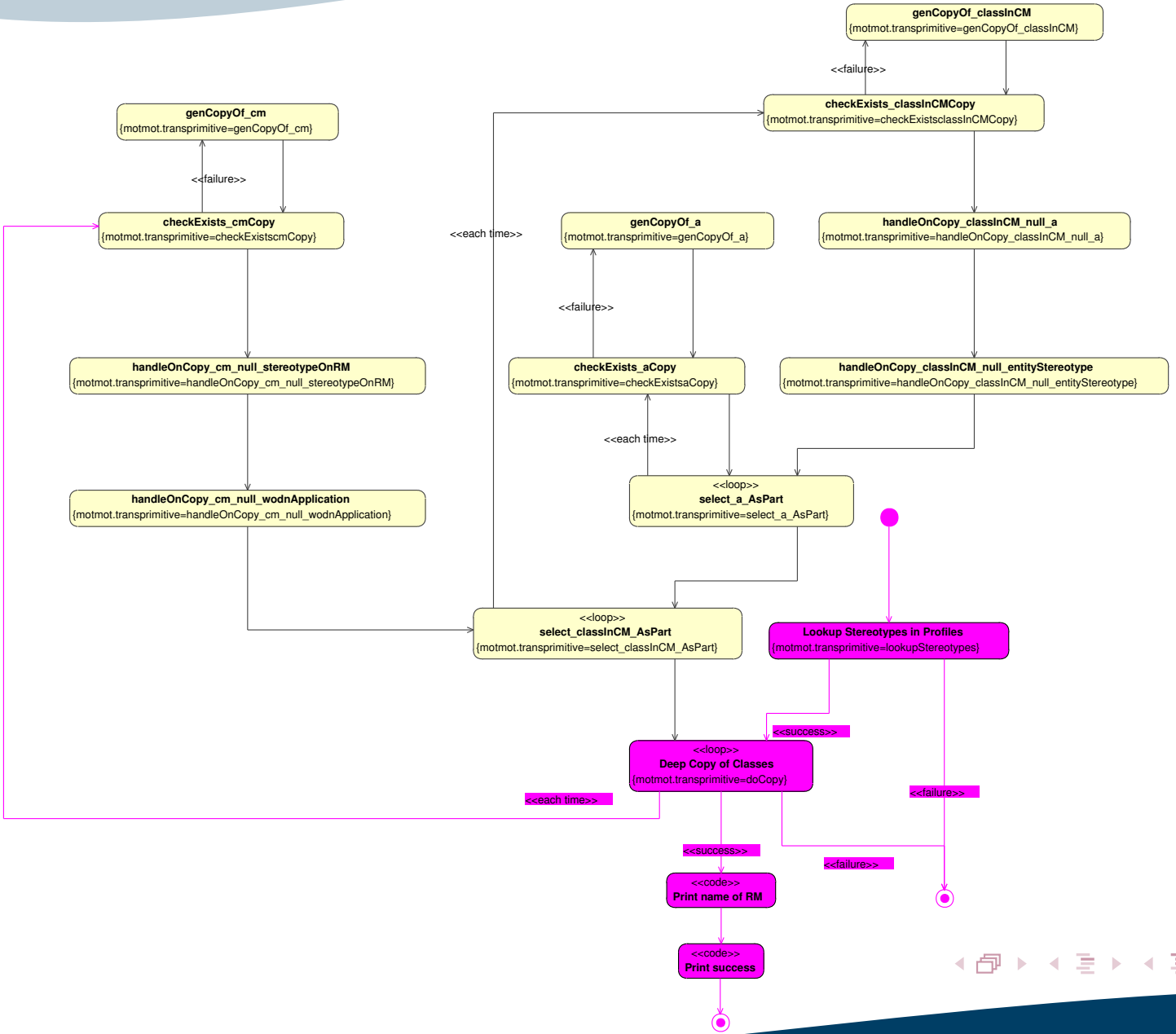
- duplicate classes with their attributes as *entities*
- no services, no screens

Outline

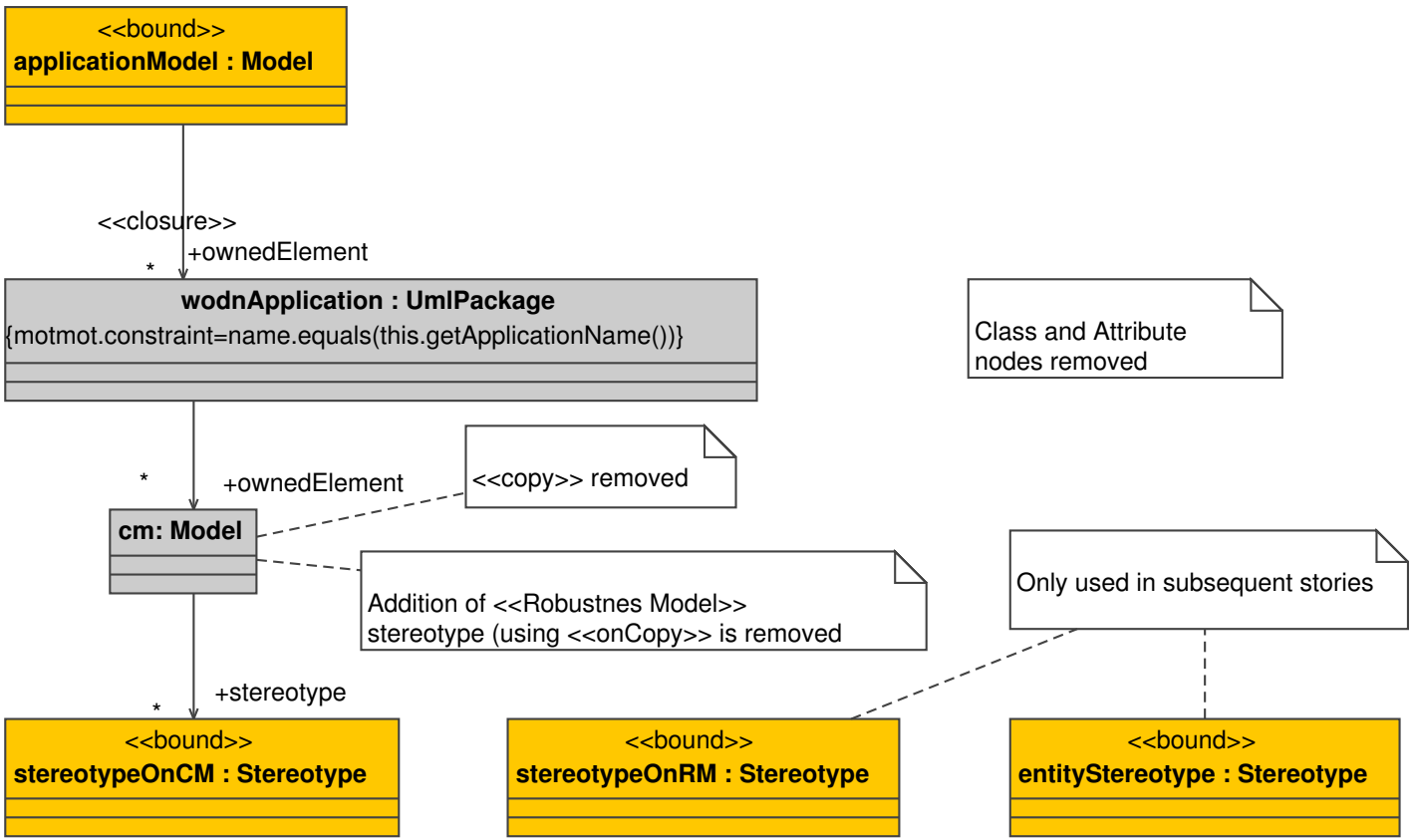
Trfo Language: Story Diagrams

- Node instances are typed by classes (attributes, associations, inheritance)
- Primitive Operations: Create/Delete a Node/Edge, Update attribute values
- Control Structure:
  - Sequence of rewrite rules: introduces $\ll$ *bound* $\gg$ nodes,
  - loop, branch, method call
- UML-alike syntax: Fujaba (large community)
- Standard UML syntax: UML Profile for SDM (MoTMoT)

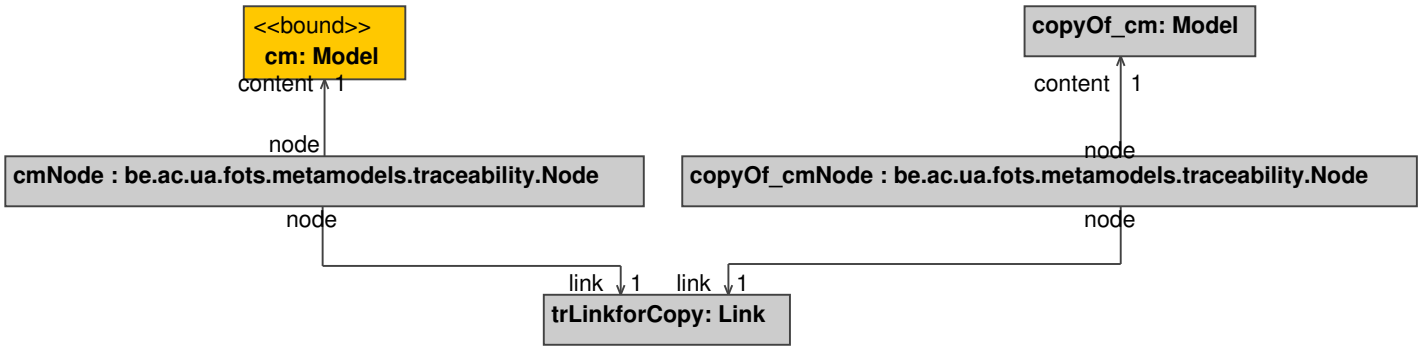
Control Flow



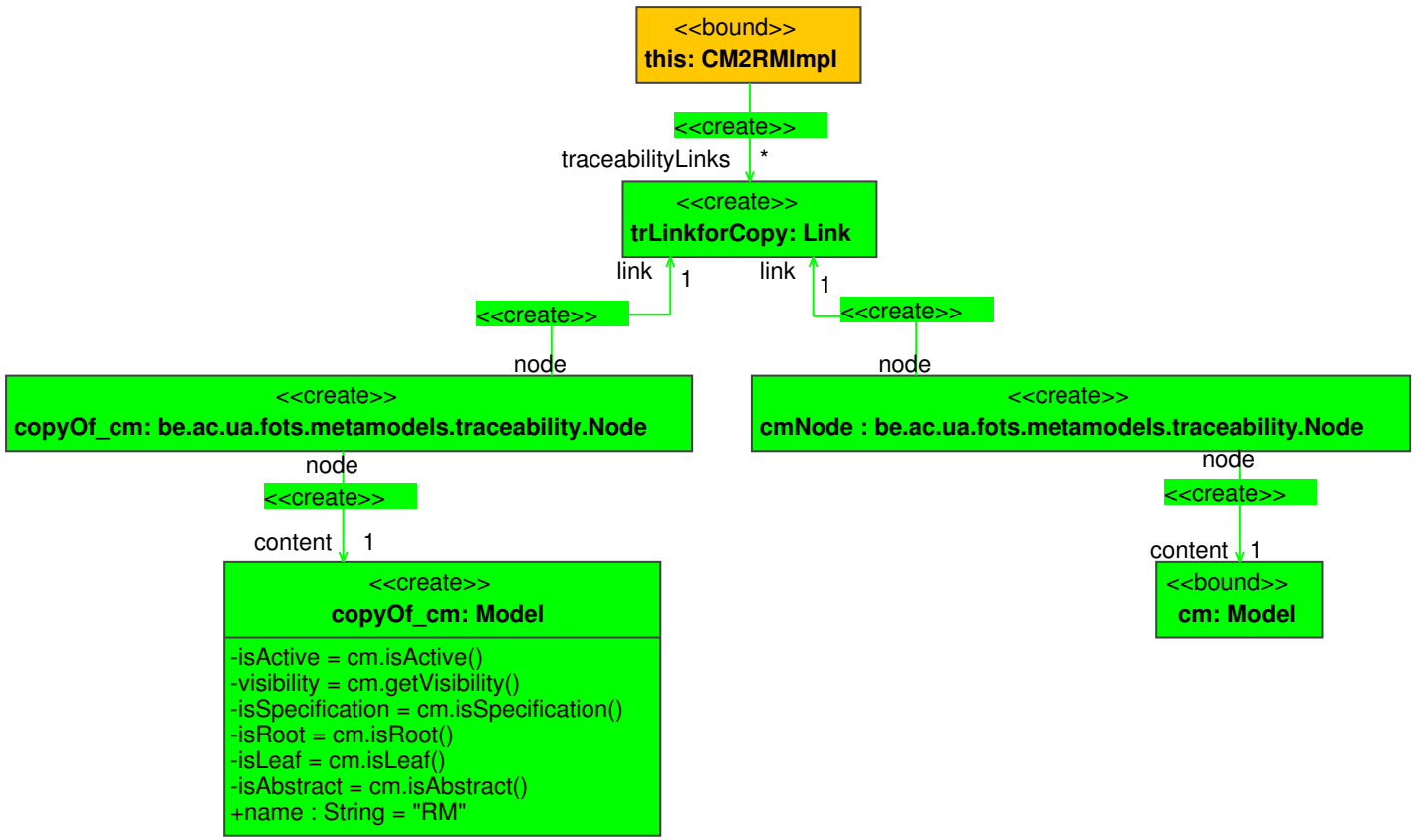
Selecting the conceptual model (CM)



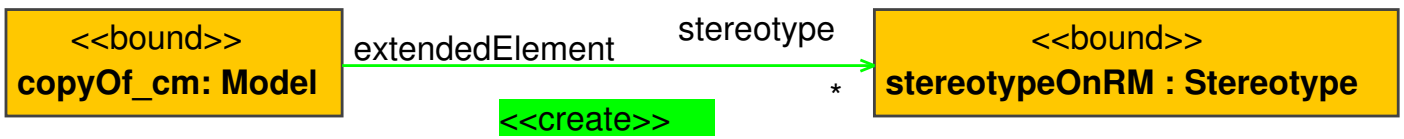
Is CM already copied?



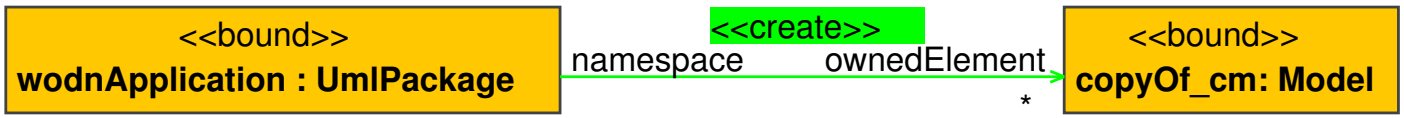
Generate Copy of CM



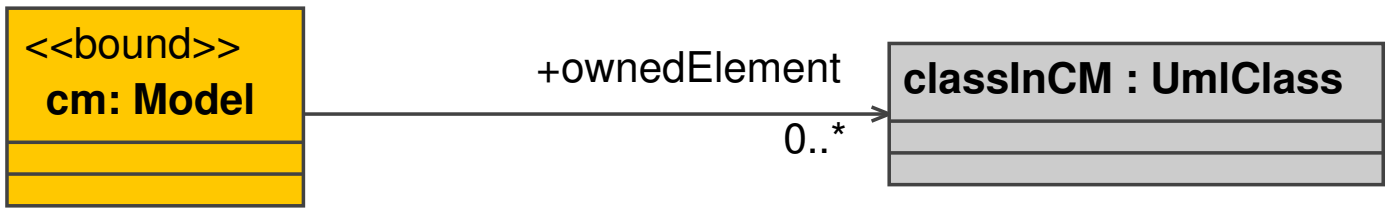
Add ≪ *Conceptual Model* ≫ Stereotype to Copy



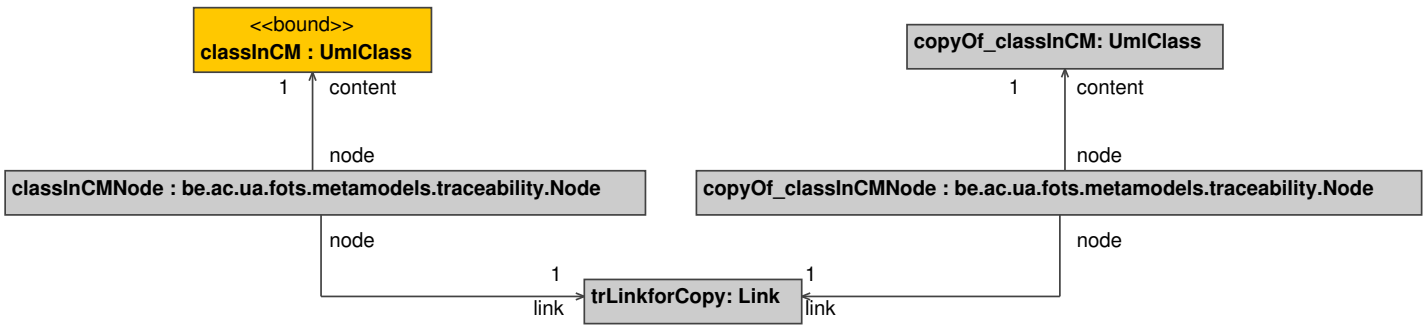
Add Copy to containing package



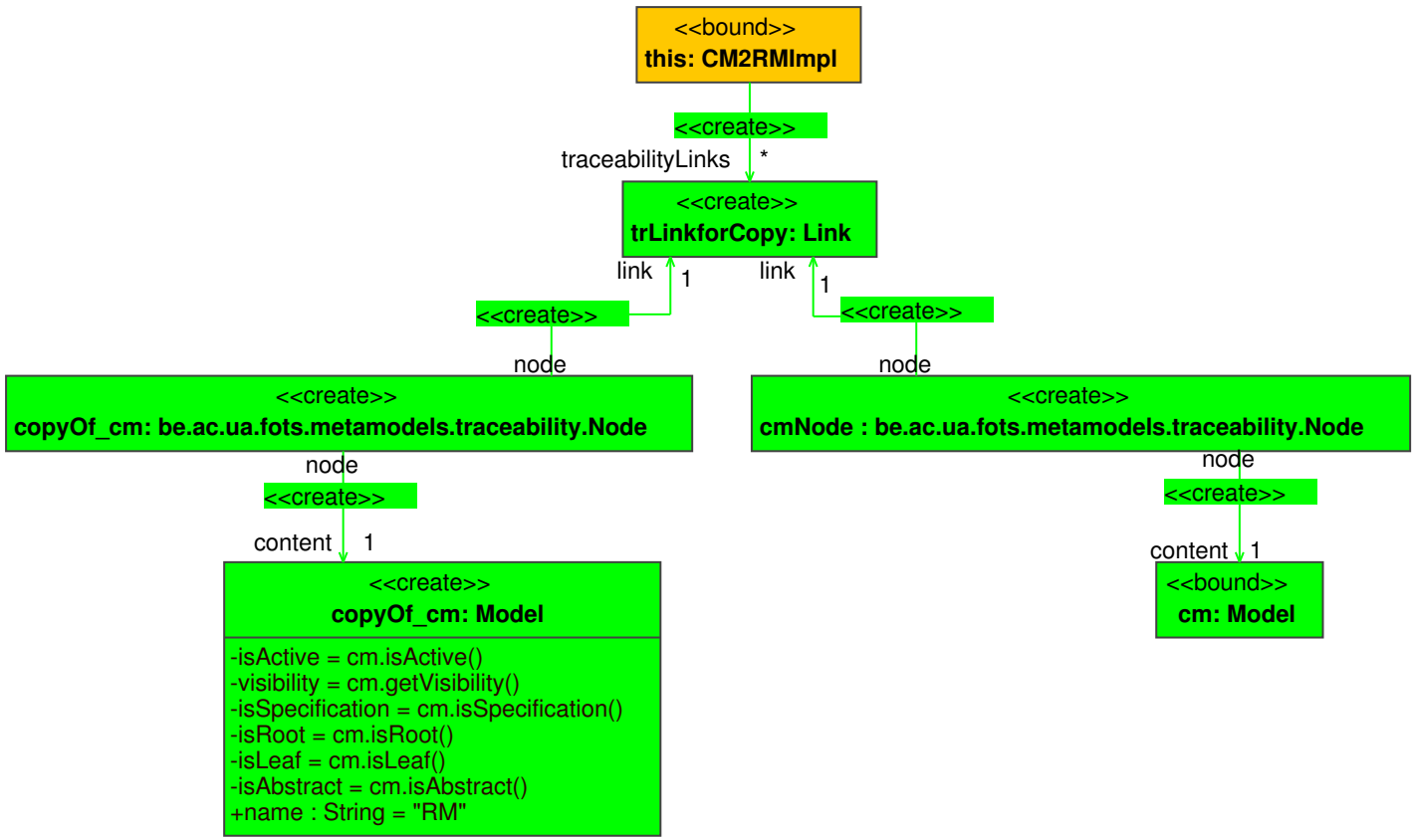
Selecting class in CM



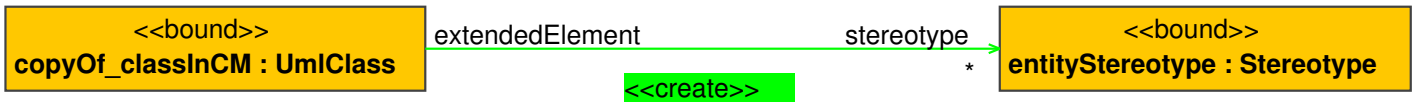
Is class already copied?



Generate Copy of class



Add ≪ *Entity* ≫ Stereotype to Copy



Story Patterns for *Attribute*

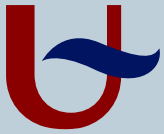
Once more:

- Check if already copied,
- Generate Copy
- Create links to copy

Conclusion from plain SDM

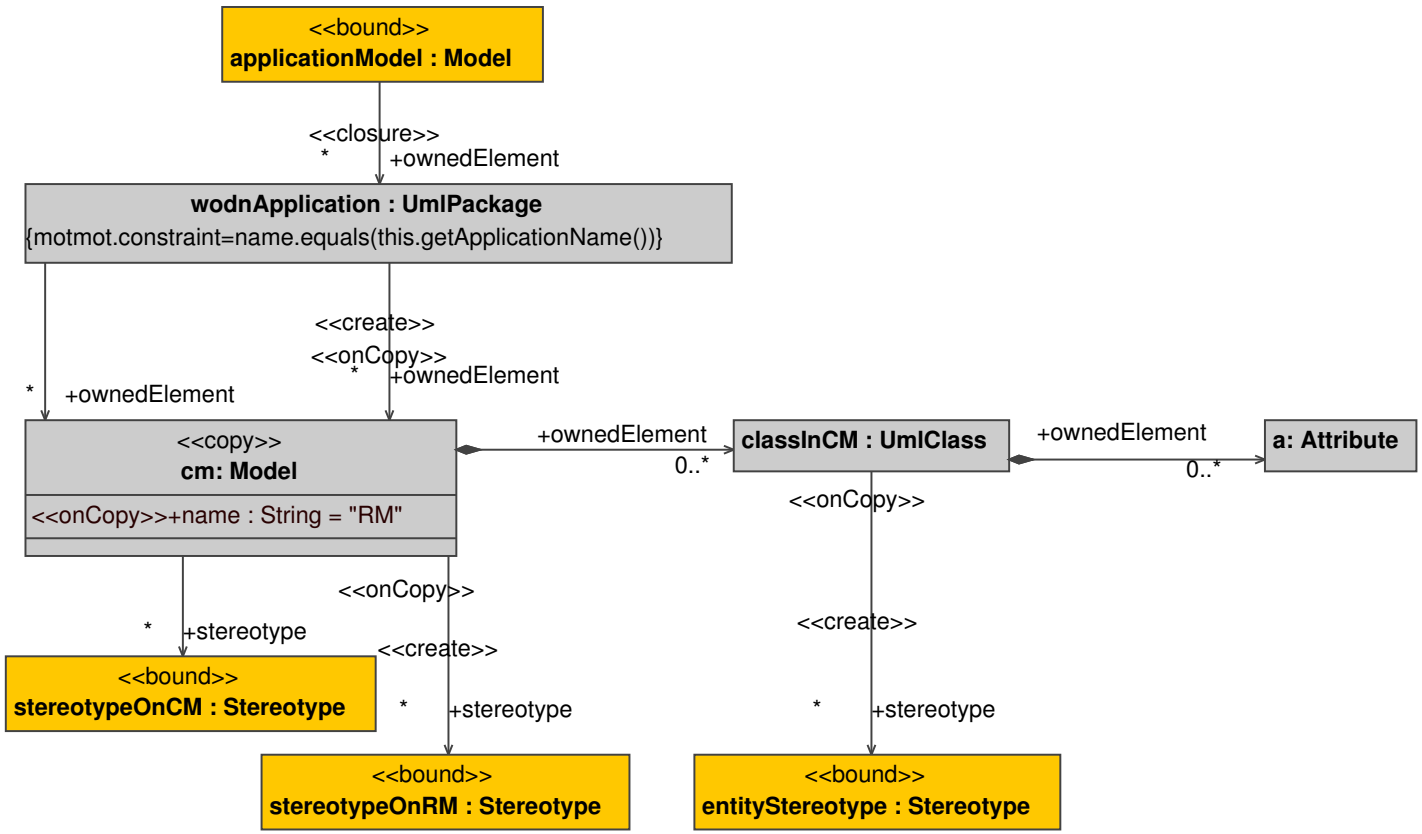
This is not a *model* of the transformation!

$\Longrightarrow$ Extend the UML Profile for SDM



Outline

CM to RM using Copy Operator



Extension to UML Profile

≪ copy ≫ entry point to the subgraph that needs to be copied

- root of `decorated tree`

composition Each node and link on this path will be copied.

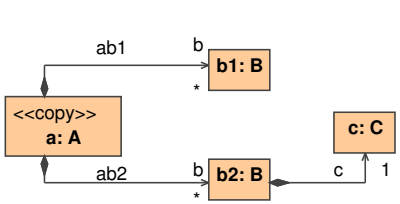
≪ onCopy ≫ instruction executed on the copy of an element.

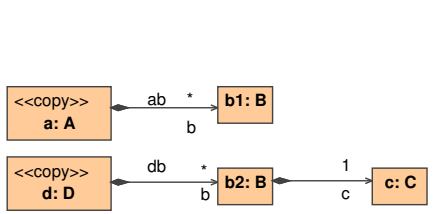
- association ends of ≪ create ≫
- attribute assignments

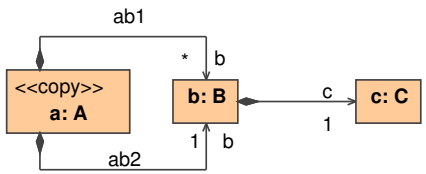
Note: **composition** also specifies selection of source elements (↔ association with two ≪ onCopy ≫ ends)

Well-formedness Rules

- At least one link should be created from the host graph to a node from the copied subgraph.
- Apply ≪ onCopy ≫ instruction only on:
  - attributes inside a copied node, or
  - association ends connected to a copied node.
- A node should be part of at most one composition.







Remark: OCL Implementation

Part 2 of WFR 2:

```
context AssociationEnd
inv:

hasStereotype(self, "onCopy") implies (
 -- connected class contains <<copy>> stereotype
 hasStereotype(self.participant, "copy") or (
  -- or end of class at other side is composite
  self.participant.oclIsKindOf(Classifier) and
  self.participant.oclAsType(Classifier).
   association.association.connection->exists(end2 |
   self<>end2 and
   end2.aggregation=AggregationKind::composite
  )
 )
)
```

Outline

Introduction

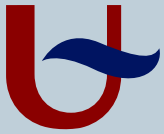
Context
- MDE
- Extending GT

Case Study
- Example Models
- Consistency Constraint

Plain SDM Trfo
- Story Diagrams
- Control Flow
- Primitives for Model
- Primitives for Class
- Primitives for Attribute
- Conclusion

Proposed Solution
- CM2RM Copy Operator
- Ext. to UML Profile
- WFRs

▷ ***Wrapping Up***
- Related Work
- Conclusion



Related Work

Hierarchical GT Assumption: transformed graphs can be decomposed into “frames” where edges are not allowed to cross frame boundaries.

- required for proving that rewrite rules do not violate grammatical constraints, but
- nested visual languages like the UML require a more flexible decomposition mechanism [DHP02],
- feedback for copy operator: automatically copy *all* edges between the nodes in a *frame*

Related Work

Shape Grammars define frame variables in the scope of a rewrite rule instead of in the scope of the complete rewriting system [Ber03].

Refactoring

- Another model transformation case study $\Rightarrow$ can apply transformation model compiler!
- Extend case study from [HJE05]: consider more grammatical constructs.
- Strict grammar based approach more attractive than controlled GT?

Conclusion

Graph Transformation Theory

- New operation: *copying subgraphs*
- More compact specification in context of model refinement, refactoring
- Only implementation is specific to model repositories, concept is applicable to any graph with OO types

In Practice

- Integrated into UML Profile for Story Driven Modeling
- M2M transformation most promising for making operator executable

Ongoing and Future Work

Making *Copy Operator* Executable

- Extend M2C Templates for SDM
- M2M refinement into plain SDM: MoTMoT independent!
  - Using conditionals presented here
  - Using Reflective helper method

Complete MDE Case Studies Learn from Integrating:

- UML Profiles (CM, RM),
- MOF (Traceability),
- OCL (Check Consistency),
- Graph Transfo (Realize Consistency),
- JMI, J2EE, ... (Platforms)

$\Longrightarrow$ Maximal Reuse, Minimal Lockin

The end

Thank you!

Questions?

References

Berthold Hoffmann.

Abstraction and Control for Shapely Nested Graph Transformation.
Fundamenta Informaticae, 58(1):39–65, 2003.

Frank Drewes, Berthold Hoffmann, and Detlef Plump.

Hierarchical graph transformation.
Journal of Computer and System Sciences, 64:249–283, 2002.

Berthold Hoffmann, Dirk Janssens, and Niels Van Eetvelde.

Cloning and expanding graph transformation rules for refactoring.
In *International Workshop on Graph and Model Transformation*, Tallinn, Estonia, 2005.
A satellite event of GPCE'05.

E Seidewitz.

What models mean.
IEEE Software, 20, Sept.-Oct. 2003.

Use of UML Profiles

- No restiction:
  - CM and RM could have different MOF metamodels, could even reside in different repositories!
  - Simple correspondence:
    - baseclass property of stereotype indicates the superclass of a new metaclass with the name of the stereotype
    - tags defined on stereotype become attributes of the new metaclass
- UML Tool Implementation:
  - The new metaclasses (M2) could be referenced from UML XMI files (M1).
  - Only accessible through MOF reflection, not through static (e.g. generated JMI) interfaces of repository.

Use of UML Profiles (cont)

- Unconventional: meta-object protocol using a *conforms to* relation different from the *instance of* relation
- Not dramatic:
  - Concrete syntax of CM and RM is OK!
  - Model transformation doesn't become more complex.
- **Most Importantly:** Copying subgraphs would still be required!