

Towards 2D Traceability

in a platform for
Contract Aware Visual Transformations
with *Tolerated Inconsistencies*

Pieter Van Gorp

pieter.vangorp@ua.ac.be

Frank Altheide

frank.altheide@gmail.com

Dirk Janssens

dirk.janssens@ua.ac.be

Universiteit Antwerpen



Antwerpen



Paderborn



Context

- *Heterogenous Models*
- *Levels of Traceability*
- *Levels of Consistency*

Background

- *CAViT*
- *ICONS*

Case Study

- *Requirements Specification*
- *Conceptual Model*
- *Robustness Model*
- *Models are Graphs*

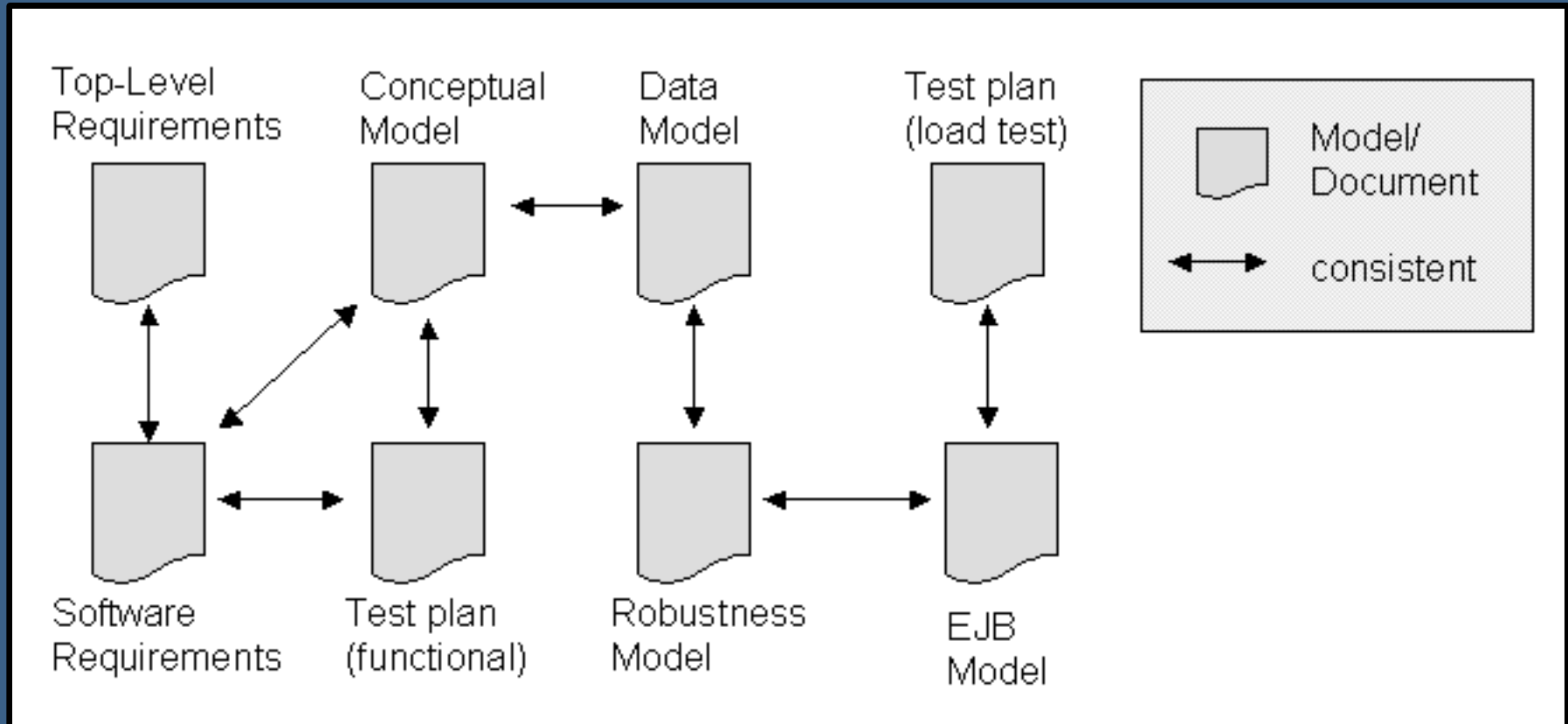
ICONS 1

- *OCL and Story Diagrams*
- *Recurring Patterns: too low-level*

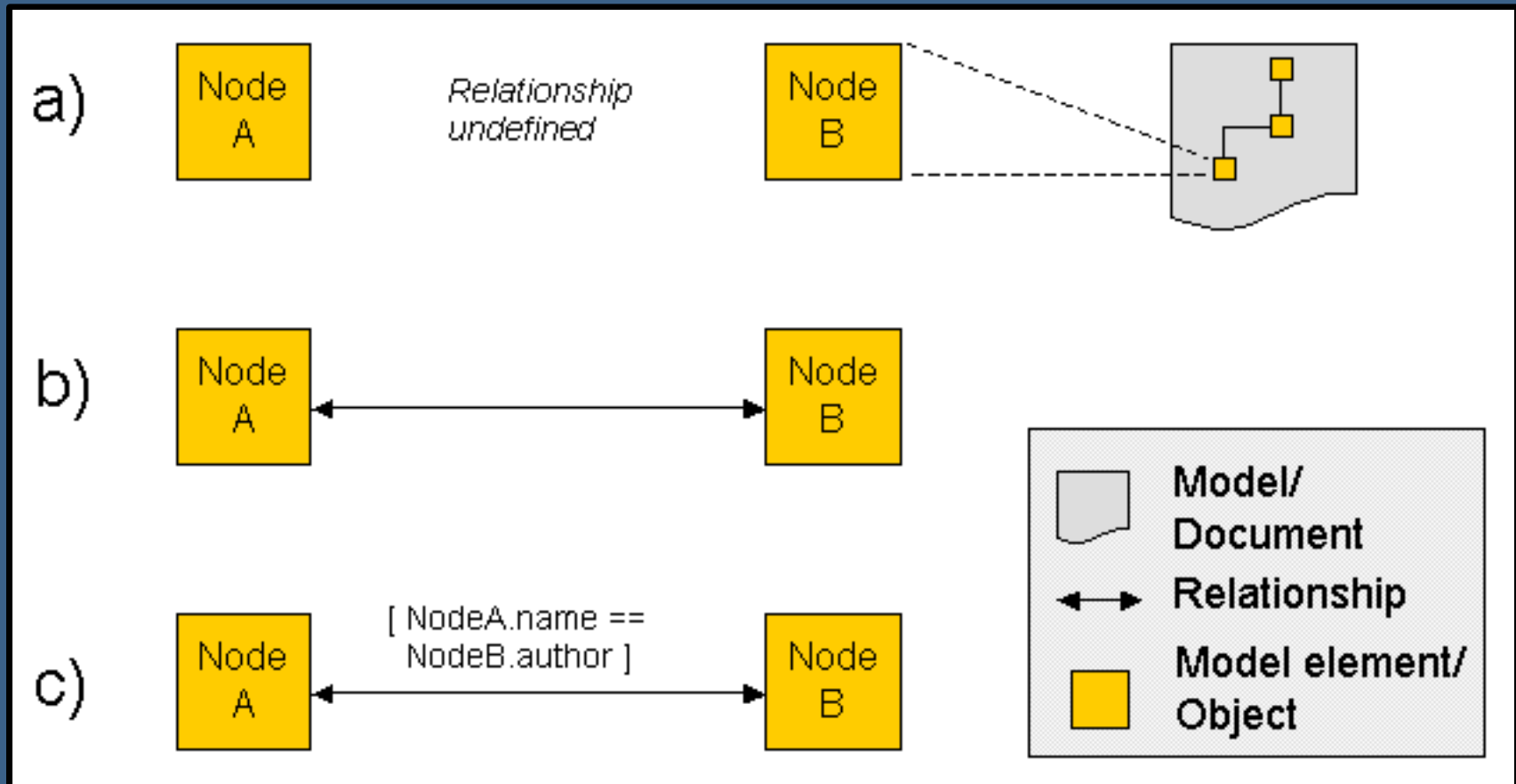
ICONS 2

- *Declarative TGG rules*
- *Refinement to Story Diagrams*
- *Tweaking Story Diagrams*
- ▶ **Traceability in 2nd Dimension!**

Heterogenous Models



Levels of Traceability



Not always feasible to reach level (c)

Consistency Maintenance

- Anthony Finkelstein. A foolish consistency: Technical challenges in consistency management. In Proceedings of the 11th International Workshop on Database and Expert Systems Applications, 2000.
- Bashar Nuseibeh, Steve Easterbrook, and Alessandra Russo. Leveraging inconsistency in software development. Computer, 33(4):pp. 24–29, 2000.

Tolerate Inconsistencies (Controlled)

MDA

- Anneke Kleppe, Jos Warmer, and Wim Bast. MDA explained: the model driven architecture: practice and promise. Object Technology Series. Addison – Wesley, 2003.

Enforce Consistency by Transformation

More recent: Model Weaving (etc.)



Context

- *Heterogenous Models*
- *Levels of Traceability*
- *Levels of Consistency*

Background

- *CAViT*
- *ICONS*

Case Study

- *Requirements Specification*
- *Conceptual Model*
- *Robustness Model*
- *Models are Graphs*

ICONS 1

- *OCL and Story Diagrams*
- *Recurring Patterns: too low-level*

ICONS 2

- *Declarative TGG rules*
- *Refinement to Story Diagrams*
- *Tweaking Story Diagrams*
- ▶ *Traceability in 2nd Dimension!*

Background (1): CAViT

Contract Aware Visual Transformations

- Why Contract Aware?
 - Constraints needed for *monitoring*
 - Use of OCL
with Inv, Pre, Post (<> ATL, YATL, ...)
 - Minimal extension of OCL's semantics:
 - » reactive behavior upon invariant violation
- Why Visual?
 - Evaluating UML as a visual QVT language
 - Graph Transformation: already quite mature in 2002

Background (2): ToolNet

The screenshot displays the ToolNet Desktop environment with several open windows:

- ToolNet Desktop:** Shows project settings for "Protector for ToolNet 010925". The "Object Text" field contains: "Das System Protector hat die Aufgabe, Schäden durch Auffahrunfälle zu verhindern oder zu verringern. Dies geschieht durch Beobachtung der Verkehrssituation, Bewertung der Verkehrssituation, Warnung des Fahrers und der Umwelt und Auslösung einer Notbremsung." The "Absolute Number" is TLR100.
- DOORS Database:** A table listing database entries:

Name	Type	Description
APG	Folder	
BaselineDiff Testum Orig	Project	Baseline Diff Testumgebung original
Beispiel für Inspection	Project	Protector-Demo
Beispiel Sitzmemory mb	Project	Beispiel-Projekt Sitzmemory (verwen
Beispielmodule	Project	Module für Screenshots in der Doku
Best Practices Telematik	Project	Artefacts, Activities and Roles
BP RefPrj Vorschlag	Project	
BP RefPrj Vorschlag 2	Project	
Copy 2 of Telematik-Beispiel für T...	Project	zur Vorbereitung XML-Datenaustaus
Copy 2 of Telematik-Beispiel für T...	Project	zur Vorbereitung XML-Datenaustaus
Copy of Beispiel Sitzmemory	Project	Beispiel-Projekt Sitzmemory
Copy of Referenzentwicklung Tele...	Project	
CteTest	Project	

- CIE CTE XL 1.4:** Shows a hierarchical diagram of a system. The root node is "NBS" (Inputs), which branches into "LKW" (Fahrerregelung), "Protector", and "Fahrzeugbedienug". "LKW" further branches into "modul" and "Fahrerregelung".
- Formal module:** A table listing requirements:

TLR	ID	Protector-Top-Level-Requirements
<input checked="" type="checkbox"/>	TLR100	Das System Protector hat die Aufgabe, Schäden verringern. Dies geschieht durch Beobachtung der Verke... Warnung des Fahrers und der Umwelt und A...
<input type="checkbox"/>	TLR101	Durch Protector entstehen keine Gefahren irr... notwendige Notbremsung, da durch diese un... Verkehrsteilnehmer ein unerwartetes Hindern... Wenn nachfolgend im Entwicklungsprozess E... Komponenten zu treffen sind, sind diese Ent... unberechtigte Bremsung nicht erfolgt; in Kon... ein konventionell ausgerüstetes Fahrzeug. Es wird also akzeptiert, dass es zu vermeidb... Notbremsungen auftreten. Unberechtigte Notbremsungen, dürfen auch t... eines Common Mode Fehlers) im System nic... Sicherheitskonzept erarbeitet.
<input type="checkbox"/>	TLR102	Das System ist so schnell wie möglich einzu... Projektmanagement werden in Kapitel 6 darg...
<input type="checkbox"/>	TLR114	Durch den Einsatz von Protector darf sich die... einem entsprechenden Fahrzeug mit ART ni...
<input type="checkbox"/>	TLR178	Dem Mißbrauch des Systems ducht Ausnutz... (Risikokompensation) ist vorzubeugen.
<input type="checkbox"/>	TLR179	Mißbrauch oder Erpressung durch angeblich...
<input checked="" type="checkbox"/>	TLR125	Für Protector sind folgende Verkehrssituat... a) die Verkehrssituation ist in Bezug auf eine... b) die Verkehrssituation ist in Bezug auf eine... durch eine Komfortbremsung beherrscht wer...

- Legend:** A list of operational states such as "Normalbetrieb", "Warnbetrieb", "Notbremsbetrieb", etc., with corresponding symbols.



Context

- *Heterogenous Models*
- *Levels of Traceability*
- *Levels of Consistency*

Background

- *CAViT*
- *ICONS*

Case Study

- *Requirements Specification*
- *Conceptual Model*
- *Robustness Model*
- *Models are Graphs*
- *Sample Constraint*

ICONS 1

- *OCL and Story Diagrams*
- *Recurring Patterns: too low-level*

ICONS 2

- *Declarative TGG rules*
- *Refinement to Story Diagrams*
- *Tweaking Story Diagrams*
- ▶ **Traceability in 2nd Dimension!**

Case Study: Meeting Scheduler

My Favorite Text Editor v12.2 MeetingScheduler.rtf - X

Meetings are typically arranged in the following way. A meeting initiator asks all potential meeting attendees for the following information based on their personal agenda:

- * a set of dates on which they cannot attend the meeting (hereafter referred as exclusion set);
- * a set of dates on which they would prefer the meeting to take place (hereafter referred as preference set).

Show Links

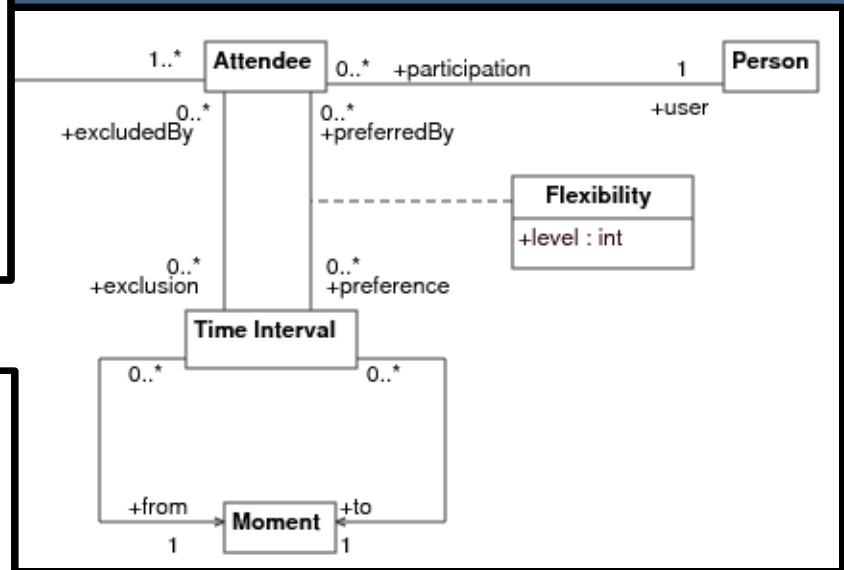
requirements>@0214("Meeting") applicationModel>@5222("Meeting") Show Delete

Add

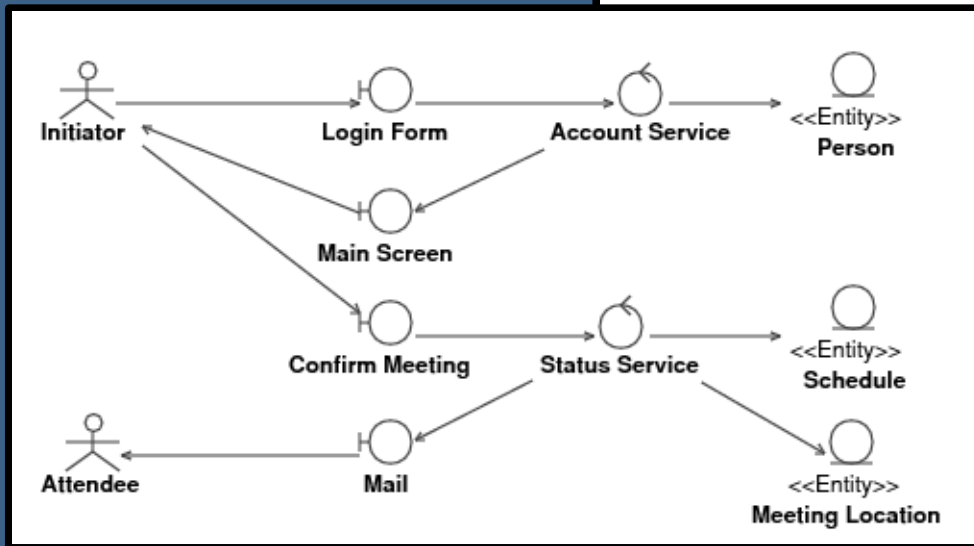
A meeting date is defined by a pair (calendar date, time period). The preference sets are contained in some time interval prescribed by (hereafter referred as date range).

The initiator also asks active participants to provide any special equipment requirements on the meeting location (e.g., overhead-projector, workstation, network connection, telephones, etc.). He/she may also ask important participants to state preferences about the meeting location.

Requirements Specification

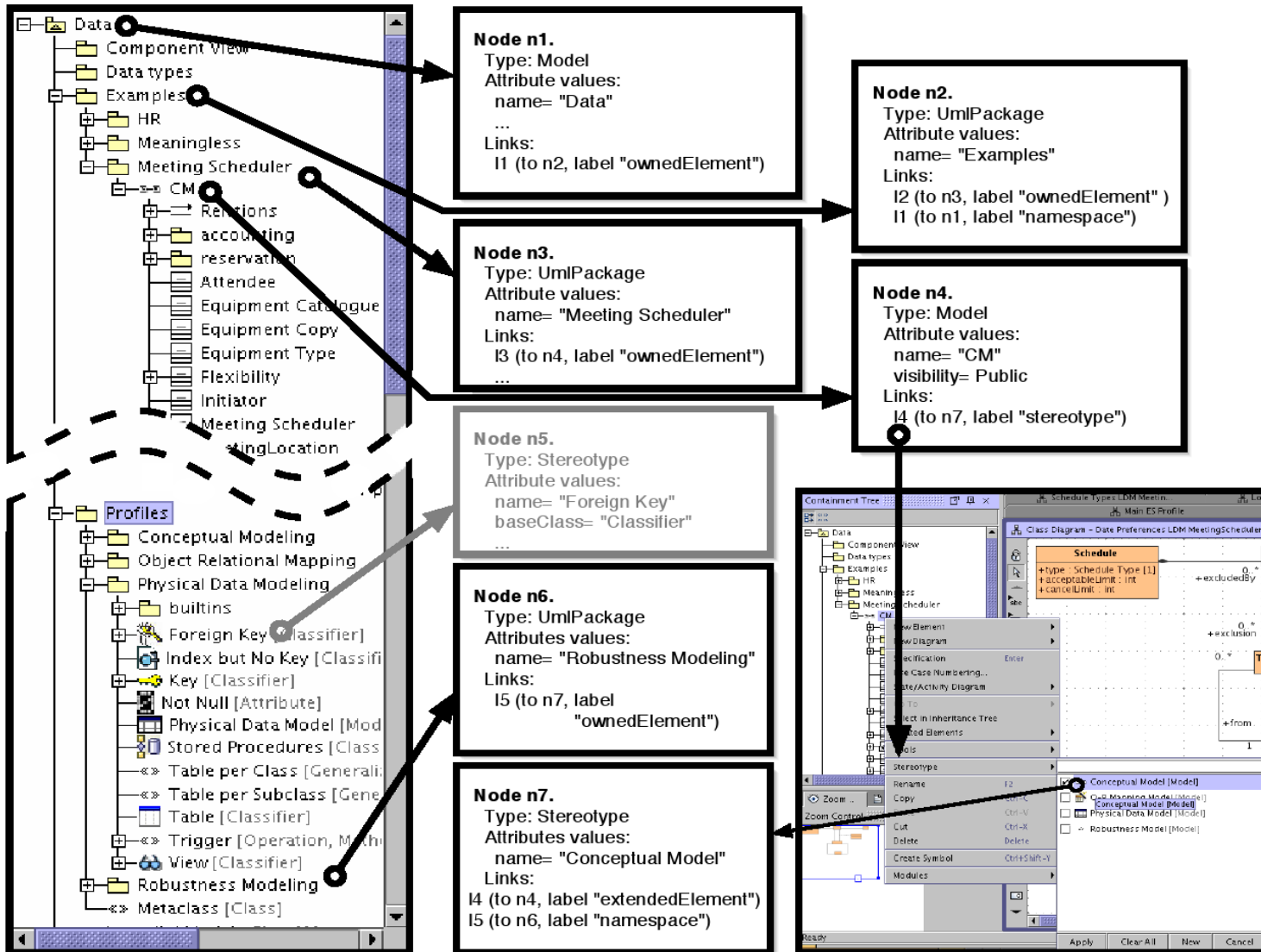


Conceptual Model



Robustness Model

UML Profiled Models as Graphs



Sample Constraint (Informal Version)

All classes from the conceptual model should correspond to entities in the robustness model. Their attributes and attribute types should correspond. Both internal types and library types should be supported.



Context

- *Heterogenous Models*
- *Levels of Traceability*
- *Levels of Consistency*

Background

- *CAViT*
- *ICONS*



Case Study

- *Requirements Specification*
- *Conceptual Model*
- *Robustness Model*
- *Models are Graphs*
- *Sample Constraint*

ICONS 1

- *OCL and Story Diagrams*
- *Recurring Patterns: too low-level*

ICONS 2

- *Declarative TGG rules*
- *Refinement to Story Diagrams*
- *Tweaking Story Diagrams*
- ▶ **Traceability in 2nd Dimension!**

CAViT: Imperative Approach OCL Transformation Contracts

Contract (INV, POST)

- *Each class traces to an entity*

```

89 -- Evaluate whether each class in the conceptual model traces to
90 -- an entity in the robustness model
91 let eachClassTracesToAnEntity(): Boolean=
92   conceptualmodelTracesToRobustnessmodel() and
93   allClassesFromModel(cm)->forAll(cmcl
94     allClassesFromModel(rm)->exists(rmcl
95       this.traceabilityLinks->select(ocllsKindOf(Class2Entity))->exists(l
96         l.node->contains(cmc) and
97         l.node->contains(rmc)
98       )
99     )
100 )

```

Declarative,
Unidirectional

Contract for Violation Scenario (PRE):

- *Robustness Model exists*
- *There are classes without an entity*

```

114 -- Transformation launched when there are classes unrelated to an entity while
115 -- RM does exist already...
116 context CMconsistentRM::fix_eachClassTracesToAnEntity_violated_rmExists(): Boolean
117 pre:
118   conceptualmodelTracesToRobustnessmodel() and -- 'rm' not Undefined
119   not eachClassTracesToAnEntity()
120
121 post: eachClassTracesToAnEntity()

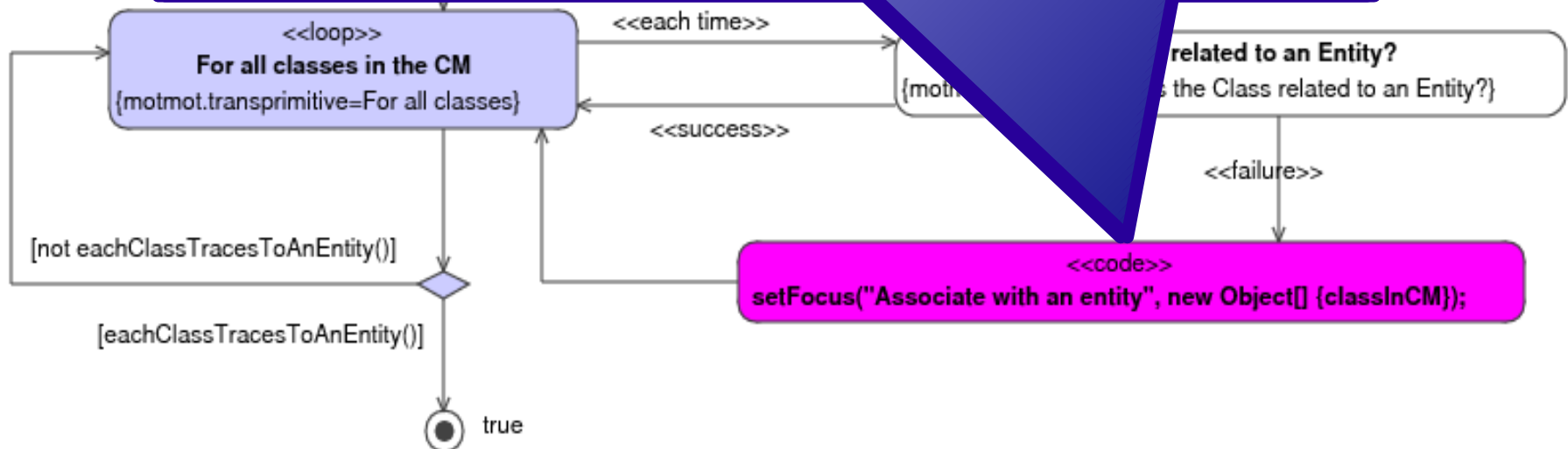
```

Engine monitors for violated
invariants, triggers proper
transformation

ICONS: Story Diagrams triggering ToolNet-GUI

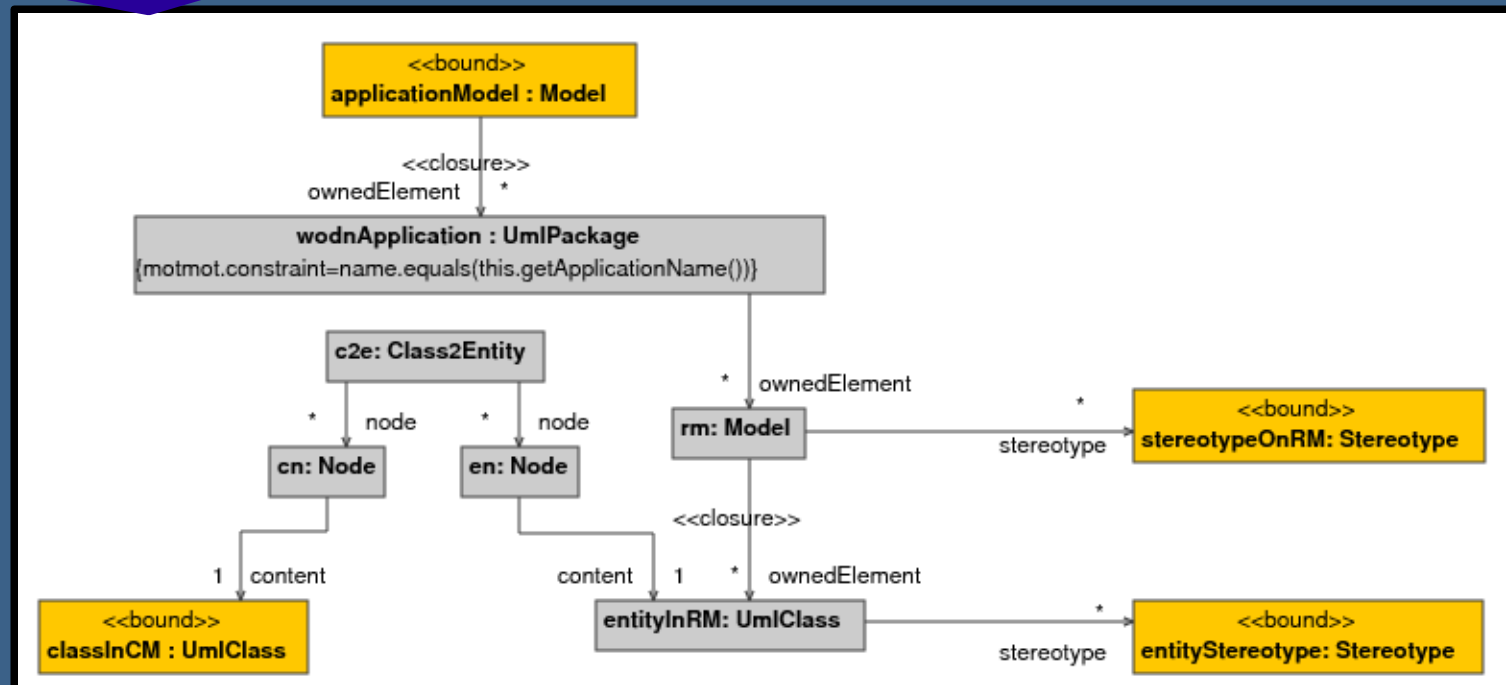
Usability:

- *Story Diagrams is formalism of transformation writer*
- *End-user (= modeler)*
 - ✓ *interacts with dialogs only.*
 - ✓ *is guided through elements in his models*



Story Pattern: “Is the Class related to an Entity?”

Model Query as a primitive graph transformation rule



Problem

- Too low-level
 - » additional example in paper
- Recurring Patterns
 - *Story diagram for creating elements*
 - *Story diagram for incremental update*
 - *Story diagram for manual resolution*
 - ...

*Can be abstracted by
TGG rules..*



Context

- *Heterogenous Models*
- *Levels of Traceability*
- *Levels of Consistency*

Background

- *CAViT*
- *ICONS*

Case Study

- *Requirements Specification*
- *Conceptual Model*
- *Robustness Model*
- *Models are Graphs*
- *Sample Constraint*

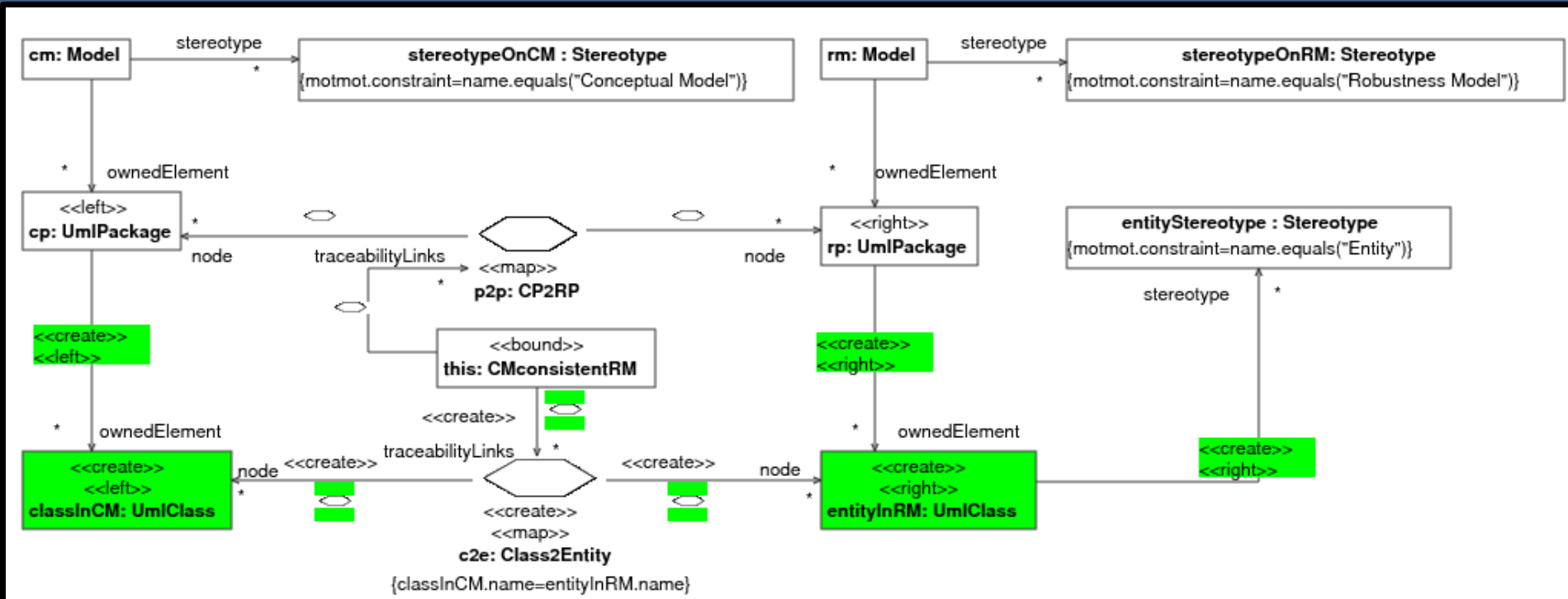


ICONS 1

- *OCL and Story Diagrams*
- *Recurring Patterns: too low-level*

ICONS 2

- *Declarative TGG rules*
- *Refinement to Story Diagrams*
- *Tweaking Story Diagrams*
- ▶ **Traceability in 2nd Dimension!**

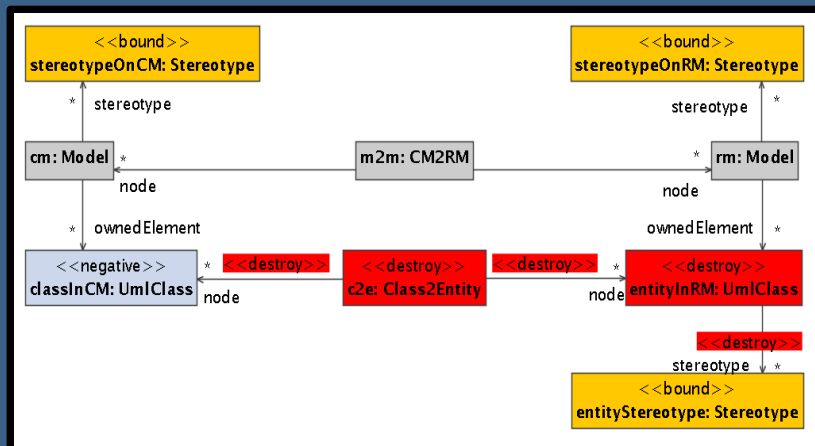
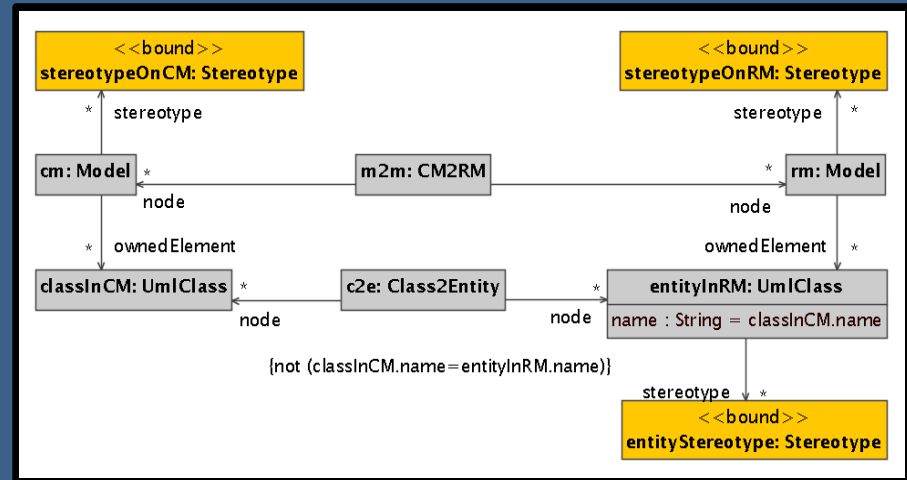
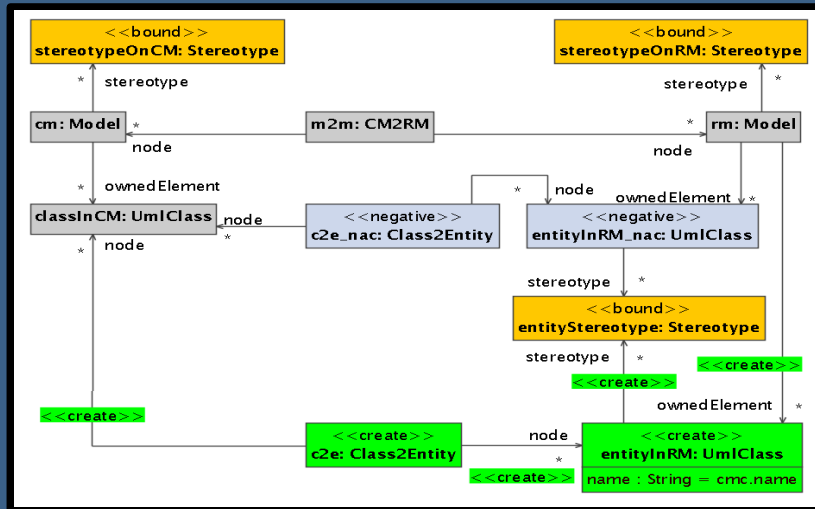


Added Value of Triple Graph Grammar rules:



- *One declarative rule covers 6 (to 9) operational rules!*
- *Reduced duplication, better focus.*

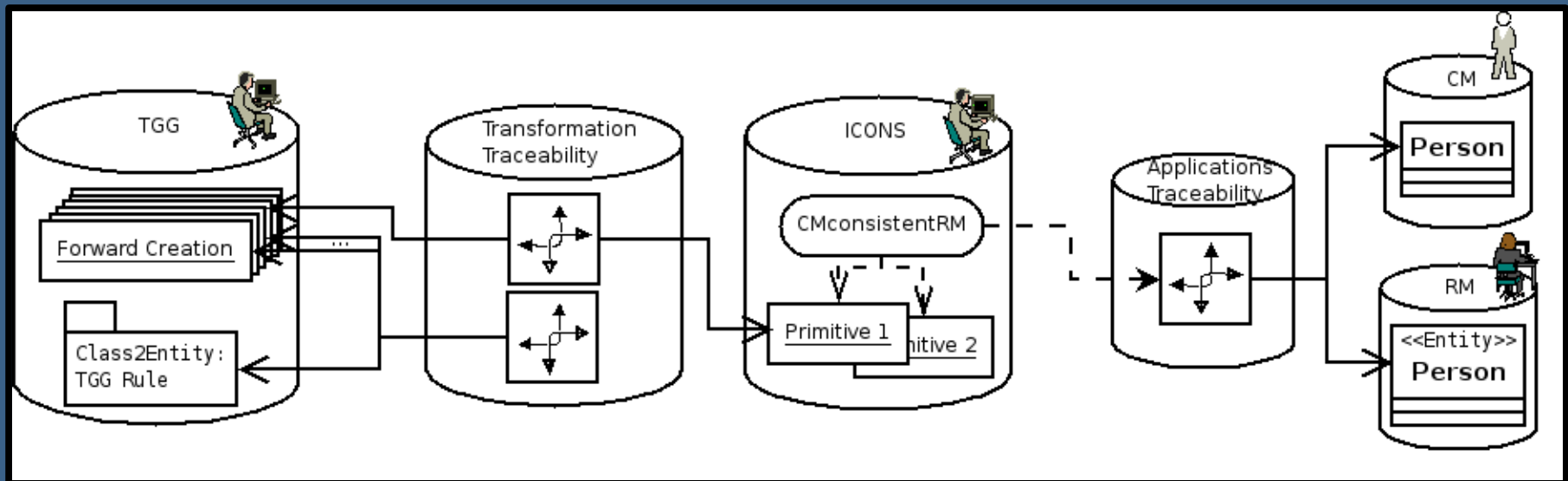
- A Higher Order Transformation produces:



- *Forward-Create*
- *Forward-Delete*
- *Forward-Consistency*
- *Backward-Create*
- *Backward-Delete*
- *Backward-Consistency*

Towards 2D Traceability

- Traceability links across application models, *created by first order transformations.*
- Traceability links between transformation models, *created by second order transformations.*



►► *Why*

Traceability in 2nd Dimension?

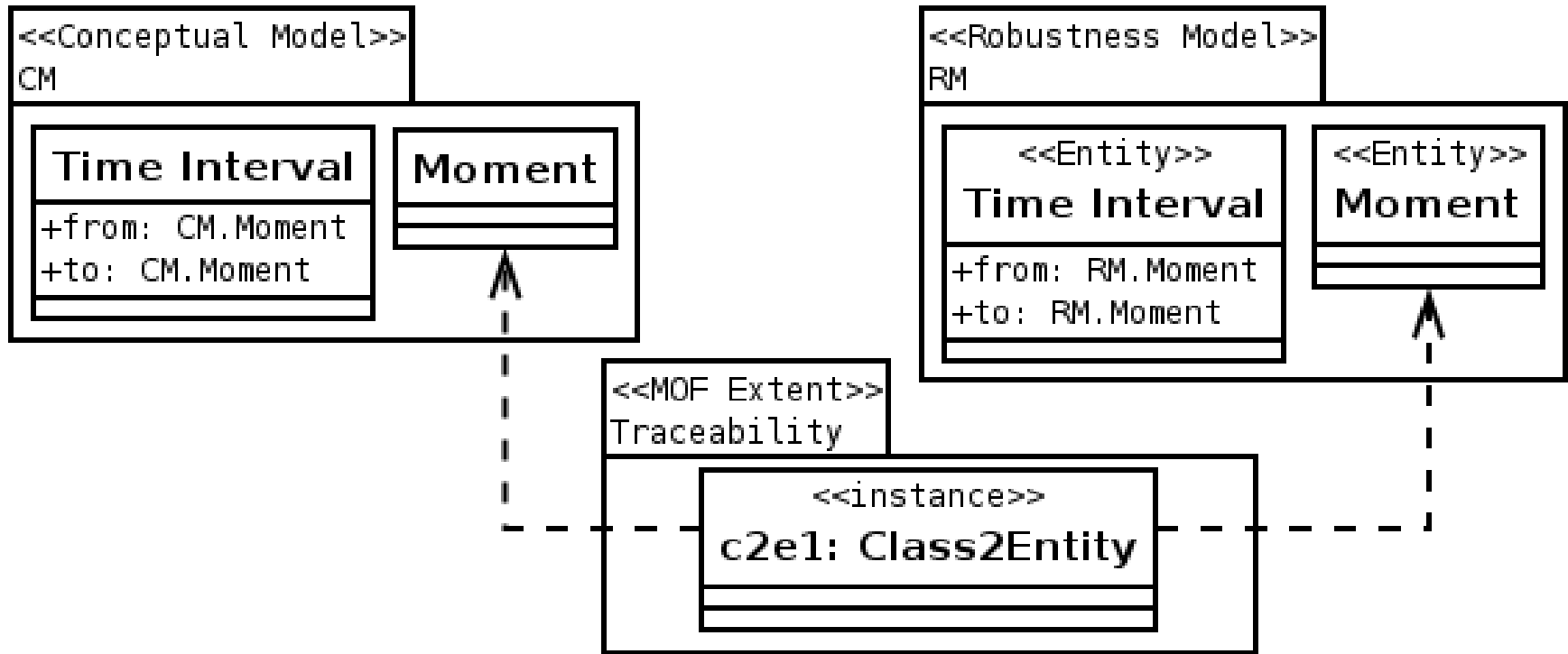
- *Not only: understandability & debugging, but also*
- *expressiveness*

Examples from case study...

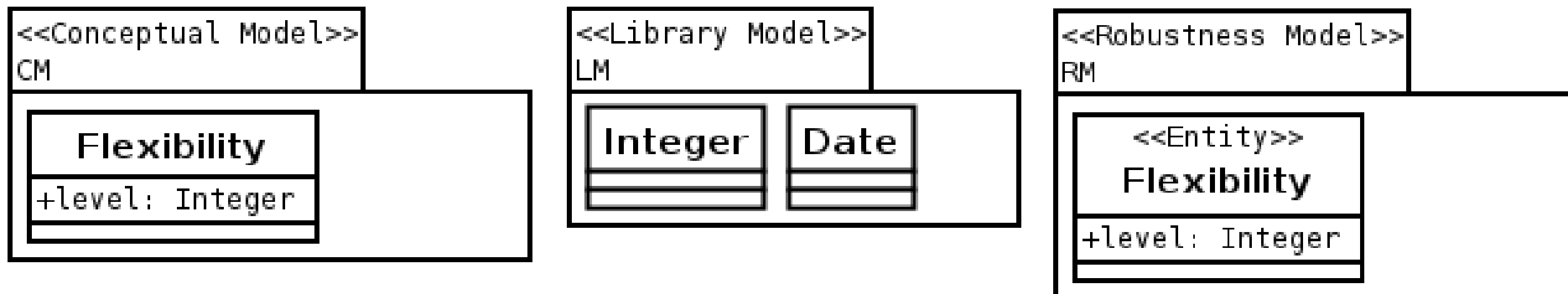
Manual Completion of derived operational rules

- Example: consistency of attribute types
 - Internal Attribute Types
 - » Type of attribute in conceptual model resides within conceptual model
 - » Type of corresponding attribute in robustness model resides in robustness model
 - External Attribute Types
 - » Type of attribute in conceptual model resides in *library* model
 - » Type of corresponding attribute in robustness model should be that library type too
 - Declarative TGG rules lead to non-determinism

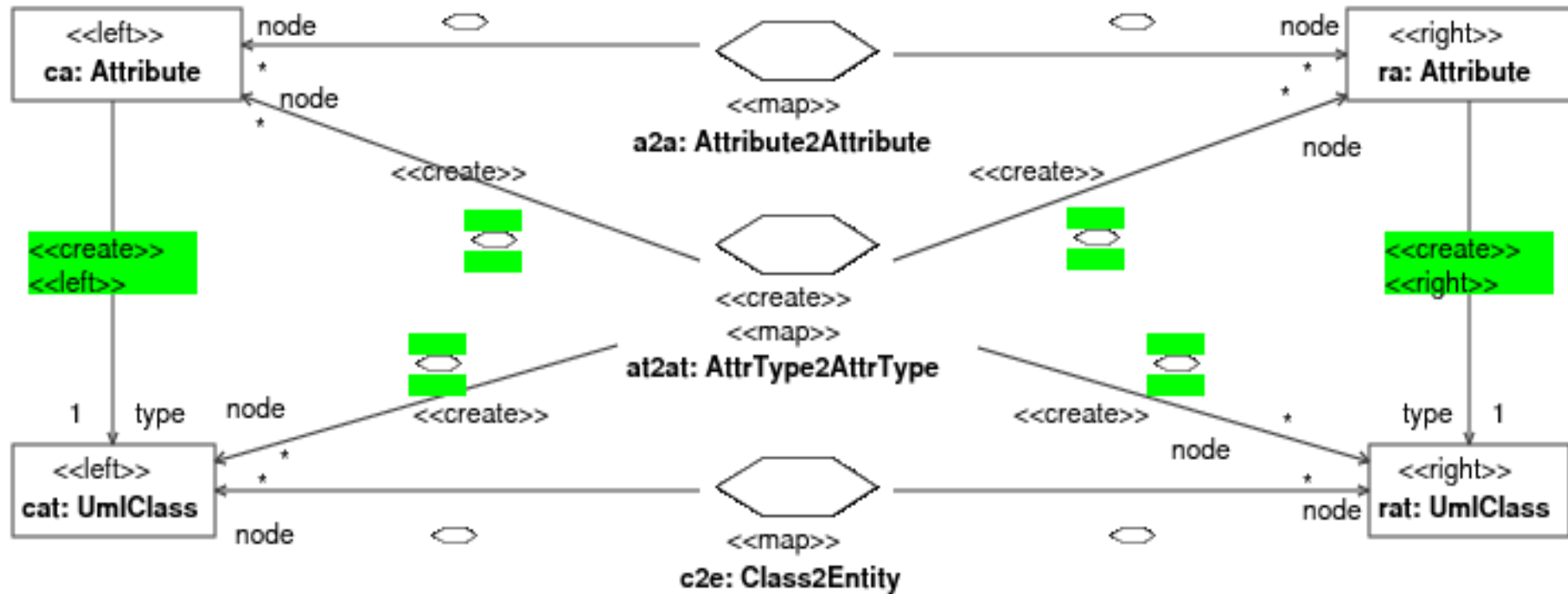
Internal Attribute Types



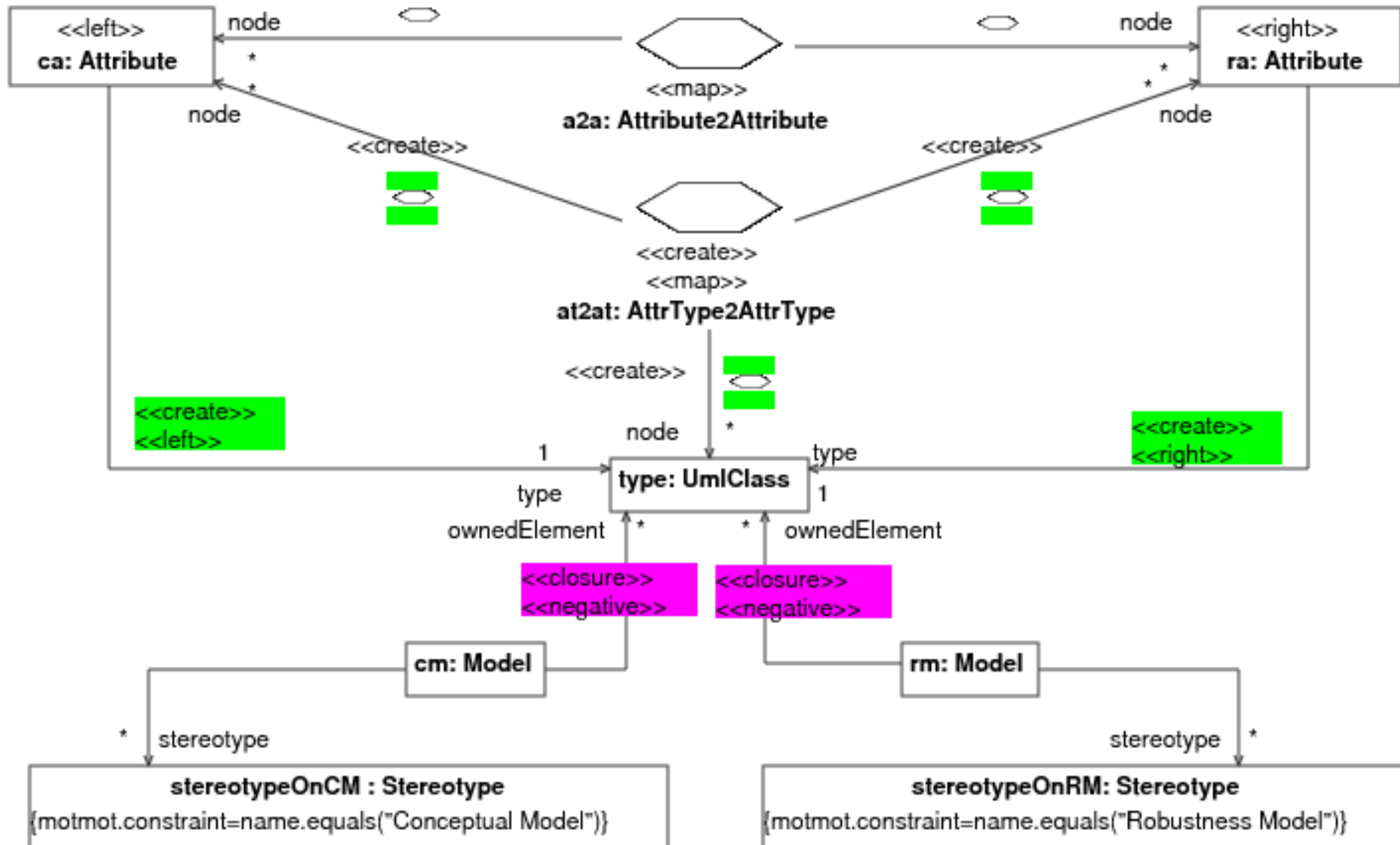
External Attribute Types



Handling *Internal* Attribute Types



Handling *External* Attribute Types



Problem

- Overlapping Applicability

1	$ca=null \ \& \ ra=null$	4	$ca \in cm \ \& \ ra=null$	7	$ca \notin cm \ \& \ ra=null$
2	$ca=null \ \& \ ra \in rm$	5	$ca \in cm \ \& \ ra \in rm$	8	$ca \notin cm \ \& \ ra \in rm$
3	$ca=null \ \& \ ra \notin rm$	6	$ca \in cm \ \& \ ra \notin rm$	9	$ca \notin cm \ \& \ ra \notin rm$

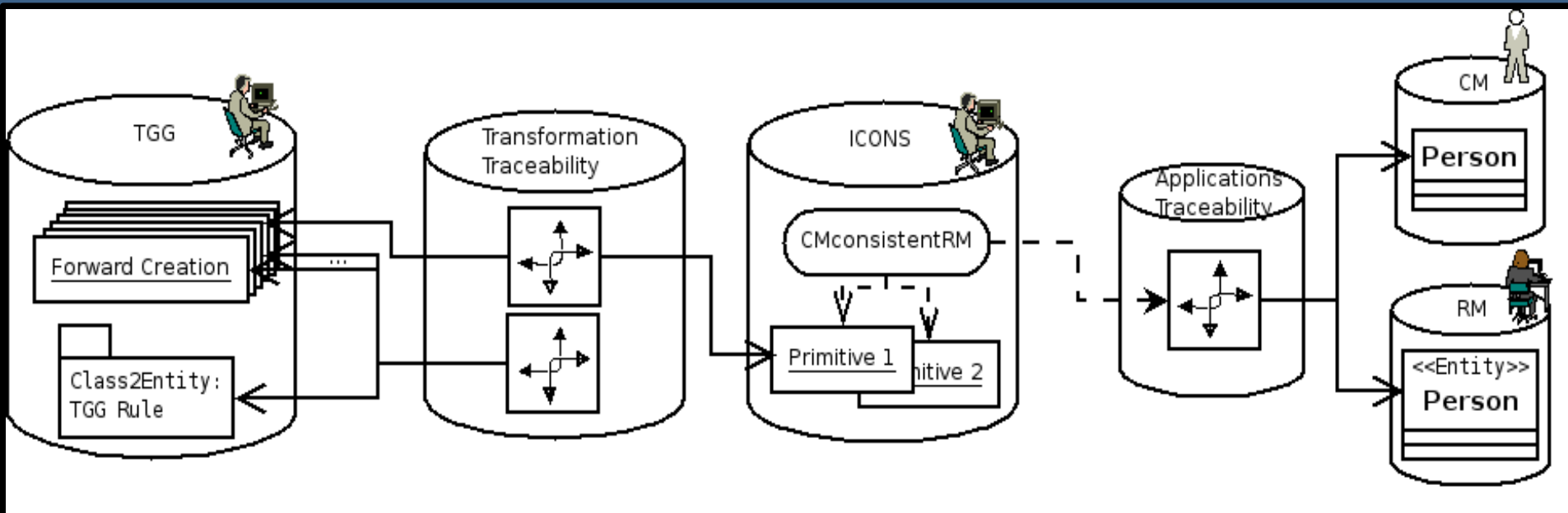
Table 1. Possible inconsistencies for attribute types.

- Need user decision to resolve



Solution

- Embed some operational rules in a control flow
 - » forward-consistency and backward-consistency in this case...
- Add/remove some operational rules manually
- Calls to ToolNet GUI takes care of interaction
 - » *setFocus, chooseAlternative*

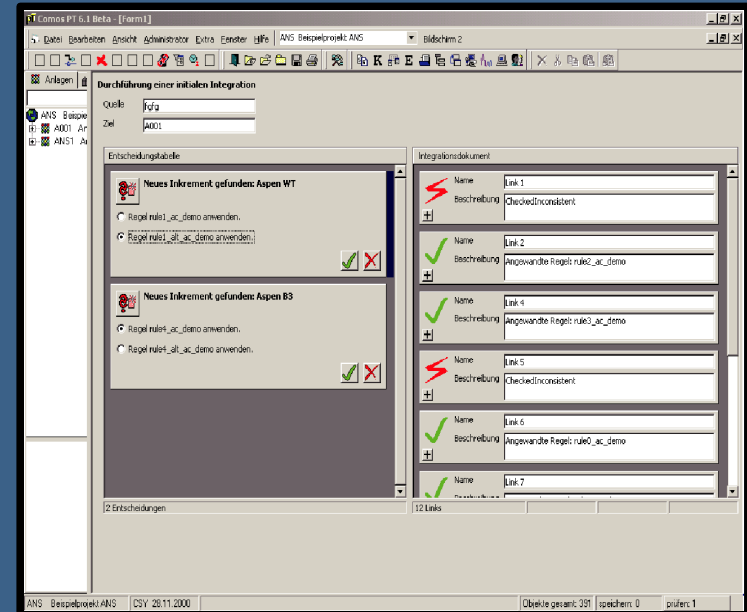
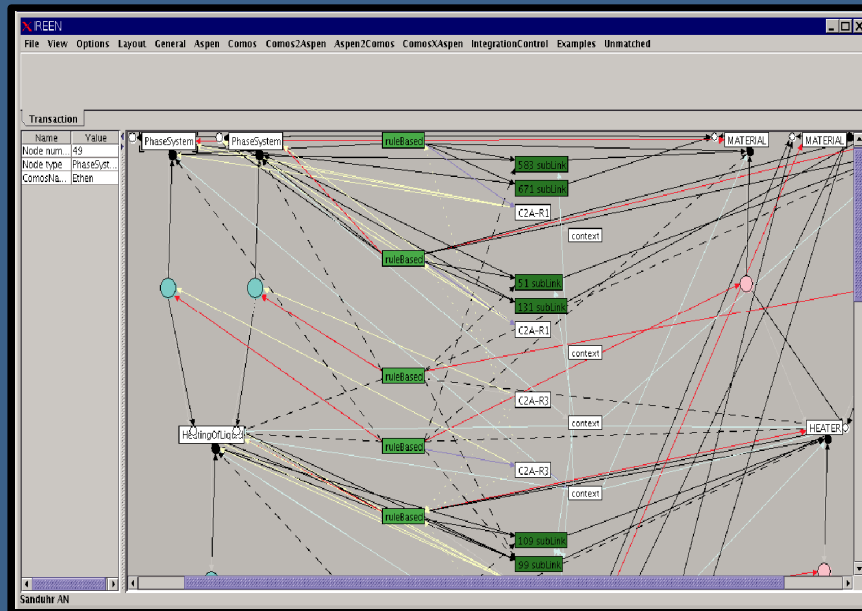


Conclusions

- Model Transformations need to
 - **Interact** with Modelers
 - Tolerate **Inconsistencies** (controlled)
- **Model Transformation Languages**
 - Need combination of declarative and imperative features
 - Manage **Complexity**
 - *Divide and Conqueror*
 - *Clean separation between declarative and imperative language*
- **2D Traceability?**
 - Transformation models become first class
 - Traceability between high-level and low-level transformation models (**PIM to PSM ~ PIT to PST**)
 - Hide 2nd Dimension from Application Modelers
 - However: essential for managing complete set of models
 - *Static Comprehension: eases transition to declarative languages*
 - *Runtime Debugging*
 - *Manual Completion*

Related Work

- Simon Becker, RWTH Aachen



- QVT Relational

- » Graphical QVT maps very closely to Triple Graph Grammars
- » Also 2 layer language: QVT relational mapped to QVT core
- » QVT could benefit from 2D traceability as well

Current & Future Work

- **Implement on top of Fujaba Plugins**
 - » MOFLON, or
 - » MOTE/MORTE
- **Extend H.O.T deriving Operational Rules from TGGs**
 - » *ignoreConstraints* property can be set on traceability link (correspondence node)
 - » no problem in ICONS 1
 - » impact on rule derivation strategy needs to be investigated further:
e.g. “*aClass.name==anEntity.name || trace.ignoreConstraints*”
- **Compare with Alternative**

Hybrid Transformation Language:

 - » *Story Diagrams, integrated with*
 - » *TGG Rules*
 - ▶▶ *Eliminates need for Transformation Traceability*
 - ▶▶ *Several Pro's/Con's*

Thanks for your Attention

Questions?

pieter.vangorp@ua.ac.be

<http://www.fots.ua.ac.be/~pvgorp/research/>