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
Overview

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

Top-Level Requirements

Conceptual Model

Data Model

Test plan (load test)

Software Requirements

Test plan (functional)

Robustness Model

EJB Model

Model/ Document consistent
Levels of Traceability

Not always feasible to reach level (c)
Levels of Consistency

Consistency Maintenance


Tolerate Inconsistencies (Controlled)

MDA


Enforce Consistency by Transformation

More recent: Model Weaving (etc.)
Overview

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
Overview

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

**Requirements Specification**

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 to as exclusion set).
- A set of dates on which they would prefer the meeting to take place (hereafter referred to as preference set).

A meeting date is defined by a pair (calendar date, time period). The preference sets are contained in some time interval described by a time period (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.

**Conceptual Model**

**Robustness Model**
Antwerpen

Paderborn

UML Profiled Models as Graphs

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

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

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

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

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

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

Node n7.
Type: Stereotype
Attribute values:
name= "Conceptual Model"
Links:
14 (to n4, label "extendedElement")
15 (to n6, label "namespace")
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.
Overview

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*

```plaintext
let eachClassTracesToAnEntity(): Boolean =
  conceptualModelTracesToRobustnessModel() and
  allClassesFromModel(cm) -> forall(cmc)
  allClassesFromModel(rm) -> exists(rmc)
  this.traceabilityLinks -> select(oclIsKindOf(Class2Entity)) -> exists(l)
  l.node -> contains(cmc) and
  l.node -> contains(rmc)

Declarative, Unidirectional
```

**Contract for Violation Scenario (PRE):**
- *Robustness Model exists*
- *There are classes without an entity*

```plaintext
context CMconsistentRM::fix_eachClassTracesToAnEntity_violated_rmExists(): Boolean
pre:
  conceptualModelTracesToRobustnessModel() and -- 'rm' not Undefined
  not eachClassTracesToAnEntity()
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

```cpp
setFocus("Associate with an entity", new Object[] {classInCM});
```
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..
Overview

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.**
Operational Rule Derivation

• 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 2\textsuperscript{nd} 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
Handling *Internal* Attribute Types
Handling *External* Attribute Types

**Diagram Description:**

- **Node** relationships denoted with `<<map>>`.
- **Attribute** relationships denoted with `<<create>>`.
- **UmlClass** type with ownedElement properties.
- **Model** nodes labeled with stereotypeOnCM and stereotypeOnRM.
Problem

• Overlapping Applicability

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ca=null &amp; ra=null</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>ca=null &amp; ra∈rm</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>ca=null &amp; ra∉rm</td>
<td>6</td>
</tr>
</tbody>
</table>

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
Tolerating Inconsistencies: Extra TGG Rule

Collecting traces of inconsistent attribute types

- Can be displayed to modeler and tolerated explicitly
- Overlapping TGG rule applicability again handled at operational level

TGG Rule causes similar ambiguity
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 2\(^{nd}\) 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/