"Model-Driven Development of Model Transformations"

Pieter Van Gorp

Universiteit Antwerpen

Overview

- Model Everything
  - Extend (Transformation) Modeling Language when desirable
- Modeling languages:
  - Do not re-invent the wheel
  - Applies especially to Transformation Modeling!
- **Copying** problem tackled
  - Transformation **Modeling** (not Programming)
  - *Challenge Me!*
- **Higher Order** Transformations
  - Extensible Transformation Languages
  - Tool-Independent Results
- **Success** Criteria
  - OUT: Evangelize Story Diagrams, Copy Operator
  - IN: Practical Experience with
    - » DEVS, Alloy
    - » MDE work for SoC

Model Everything

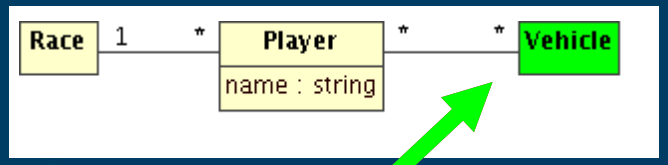
- Why?

I. Keep it Stable

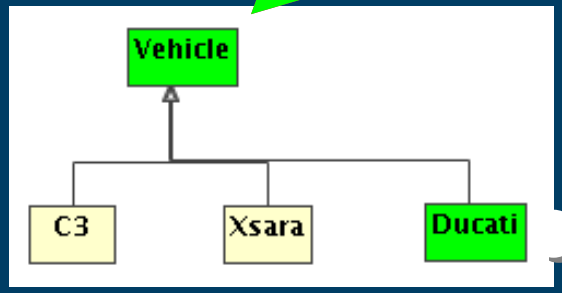
II. Make it Human-Friendly

Modeling Languages & Tools: Human Friendly

- Example: Decompose complex models into **Views**
  - **Association View**



  - **Inheritance View**



models

  - **Automatic Consistency Maintenance**
    - **Mainstream: several industrial tools**

Model-Driven Development of Model Transformations

- ***Model*** the **Model Transformations**
  - Evaluate the UML
  - State-of-the-art
    - Start
      - » Taxonomy: compare existing approaches, identify weaknesses
      - » Graph Transformation Languages: Story Diagrams
    - Standardize
    - Improve
      - » BUT: Maintain UML's interoperability
  - General Applicability

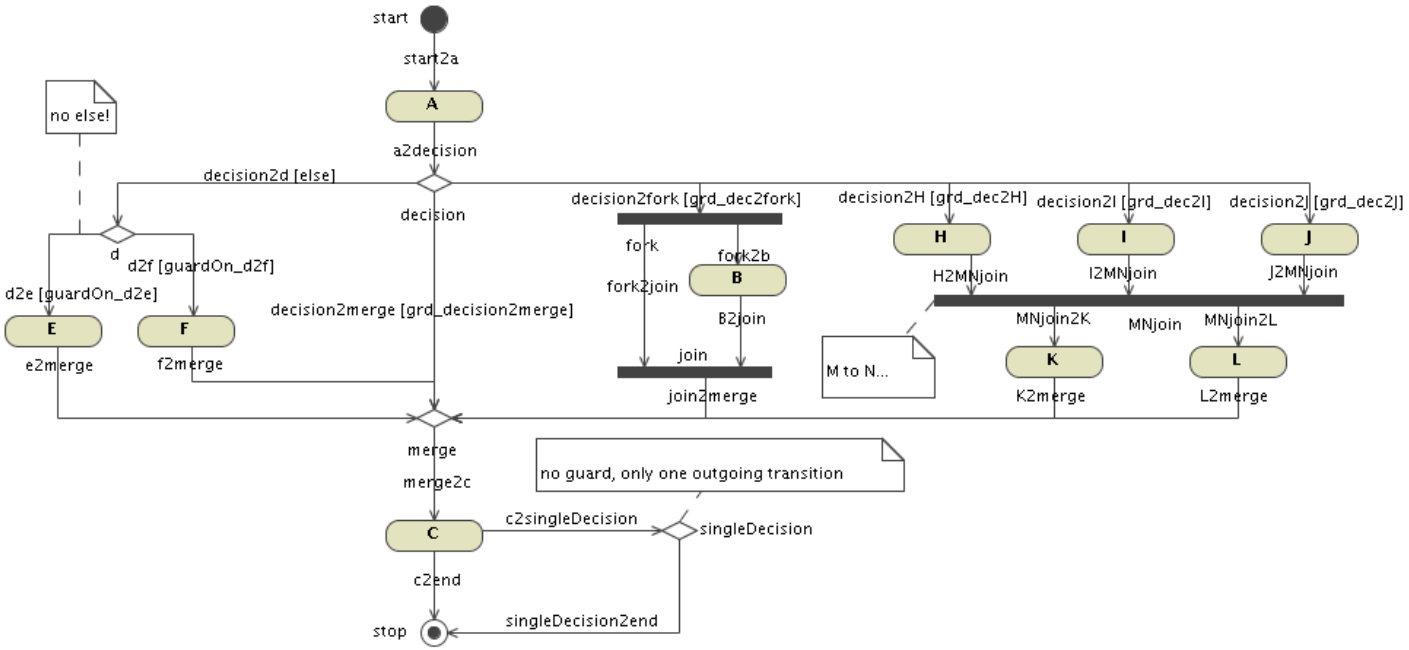
- **Transform** the **Transformation Models**

Part 1

Evaluating the UML for

Transformation Modeling

Universiteit Antwerpen




```
1  start2a = A → a2decision
2  a2decision = (
3   decision2J ⟨ grd_dec2J ⟩ (
4    decision2I ⟨ grd_dec2I ⟩ (
5     decision2H ⟨ grd_dec2H ⟩ (
6      decision2merge ⟨ grd_decision2merge ⟩ (
7        decision2fork ⟨ grd_dec2fork ⟩
8         decision2d
9      )
10    )
11   )
12  )
13 )
14 decision2fork = ( fork2join || fork2b )
15 c2end = SKIP
16 merge2c = C → c2end
17 fork2b = B → B2join
18 d2e = E → e2merge
19 d2f = F → f2merge
20 decision2d = (
21  d2f ⟨ guardOn_d2f ⟩ (
22   d2e ⟨ guardOn_d2e ⟩ SKIP
23  )
24 )
```

```
25 c2singleDecision = SKIP
26 decision2H = H → H2MNjoin
27 decision2I = I → I2MNjoin
28 decision2J = J → J2MNjoin
29 MNjoin2K = K → K2merge
30 MNjoin2L = L → L2merge
31 TrMNjoin_to_MNjoin = (
32   MNjoin2L || MNjoin2K
33  )
34 decision2merge = merge2c
35 join2merge = merge2c
36 f2merge = merge2c
37 e2merge = merge2c
38 K2merge = merge2c
39 L2merge = merge2c
40 fork2join = processJoin → join2merge
41 B2join = processJoin → SKIP
42 H2MNjoin = processJoin →
43              TrMNjoin_to_MNjoin
44 I2MNjoin = processJoin → SKIP
45 J2MNjoin = processJoin → SKIP
```

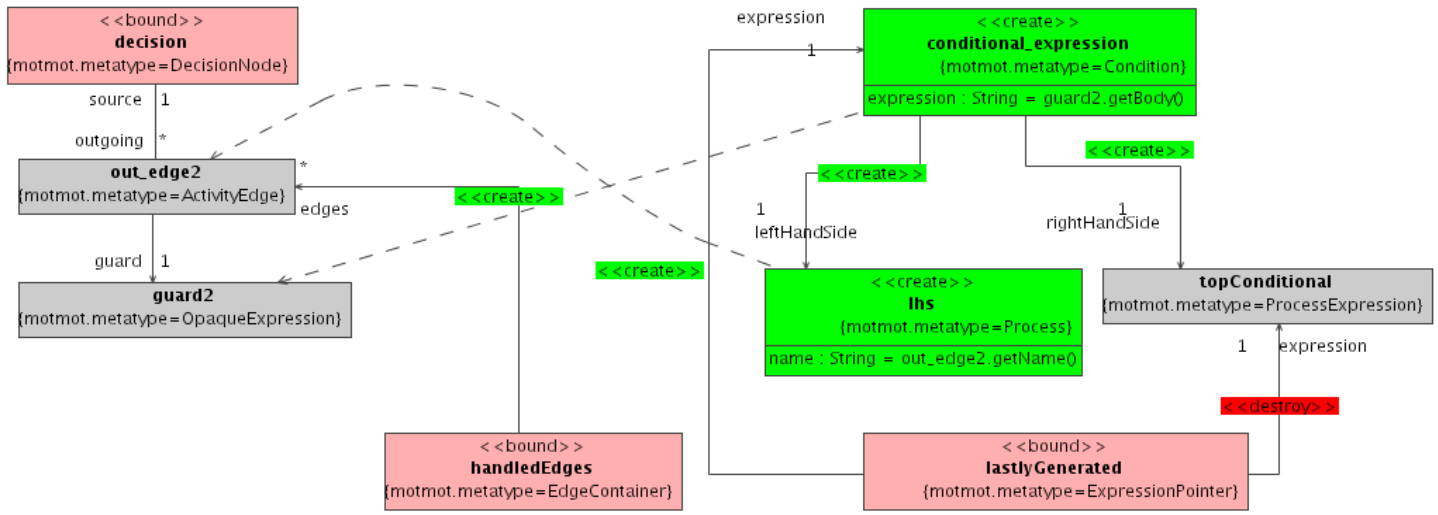
Input:
UML
Activity Diagram

Output:
CSP Program

Antwerpen

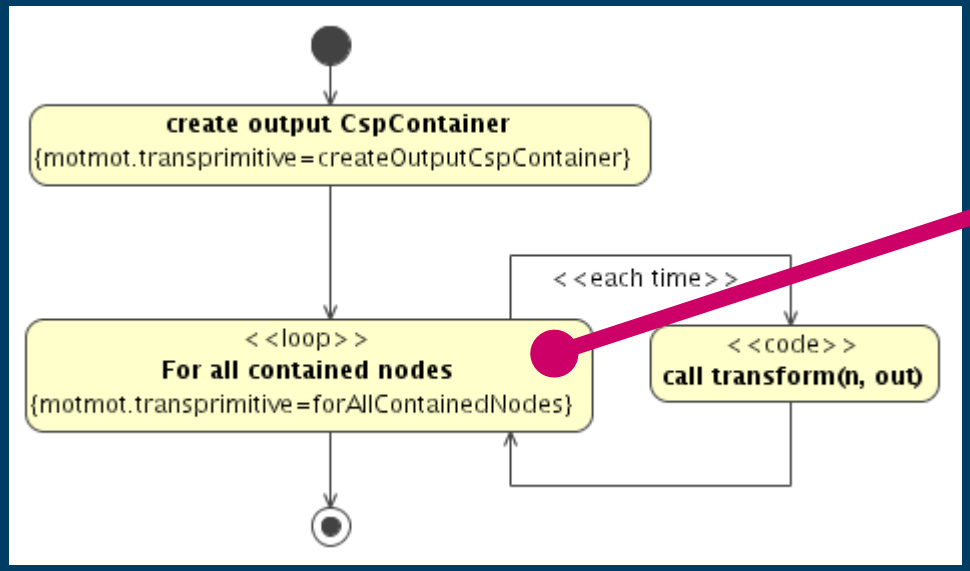
UML2CSP: *MoTMoT as state-of-the-art GT tool*

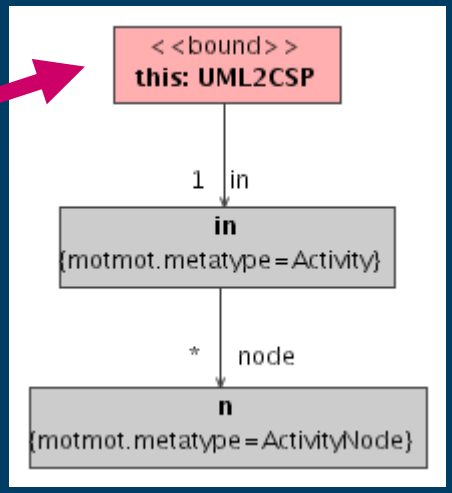
- *"industrial"* UML 2.0 input: MagicDraw 10 → standard with "variations" ☺
- Transformation defined **strictly** on UML 2.0 standard
- In-place normalization of input
  - Violating *Decision* node → Standard-Compliant *Merge* node
  - Violating *Fork* node → Standard-Compliant *Join* node
- Views: "for free" from application modeling tool (MagicDraw 9)



MoTMoT: *UML Profile for Story Diagrams*

– Explicit Control Flow modeling: branch, iterative loop (*forEach*), calls, ...





– Reasoning about termination (typical GG verification example)
= not an issue (yet?) in our case studies

Universiteit Antwerpen

Cognitive Evaluation

(of Transformation Modeling Language)

+	-
Closeness of Mapping (syntax ~ metamodeling)	Juxtaposition (embedding… native Fujaba)
Role-Expressiveness (edges: relation to whole)	Progressive Evaluation (tool issue?)
Secondary Notation (2D layout, color, …)	Diffuseness (however: issue of input/output metamodels!)
Consistency (able to infer <<onCopy>> <<destroy>>)	

T. R. G. Green. **Cognitive dimensions of notations**. In Alistair Sutcliffe and Linda Macaulay, editors, People and Computers V, pages 443–460, New York, NY, USA,1989. Cambridge University Press.

Universiteit Antwerpen

Generality

(of Transformation Modeling Language)

Synthesis

UML2CSP
CM2RM
Copy2GT

Refactoring

Pull Up Method
Extract Interface
Push Down Method

Normalization

UML2CSP

Chaining

UML2CSP

Model Synchronization

Operational
Declarative, Hybrid:
>> outline for future work

Part 2

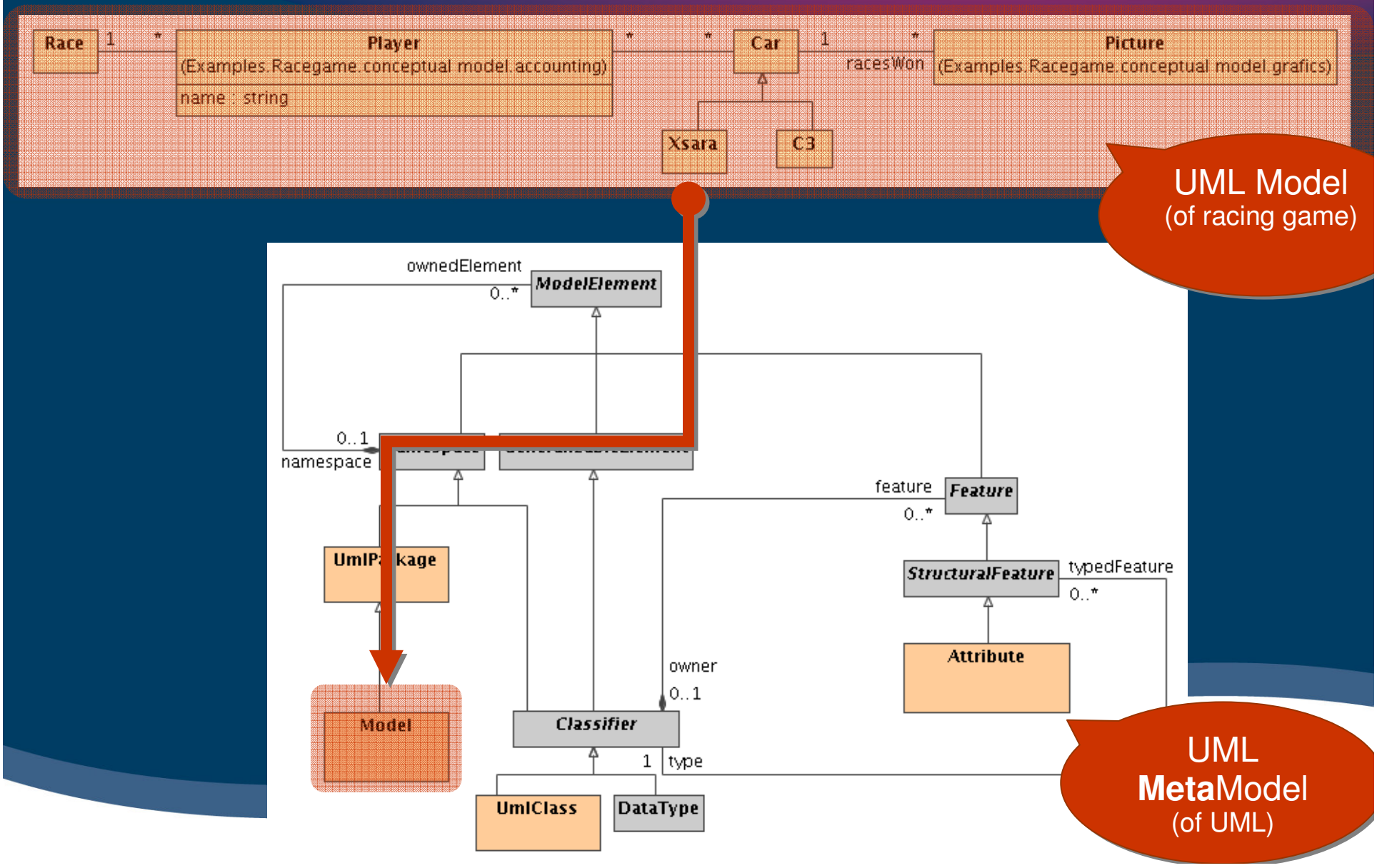
Improving the state-of-the-art in

Transformation Modeling

(Progres, Story Diagrams, QVT)

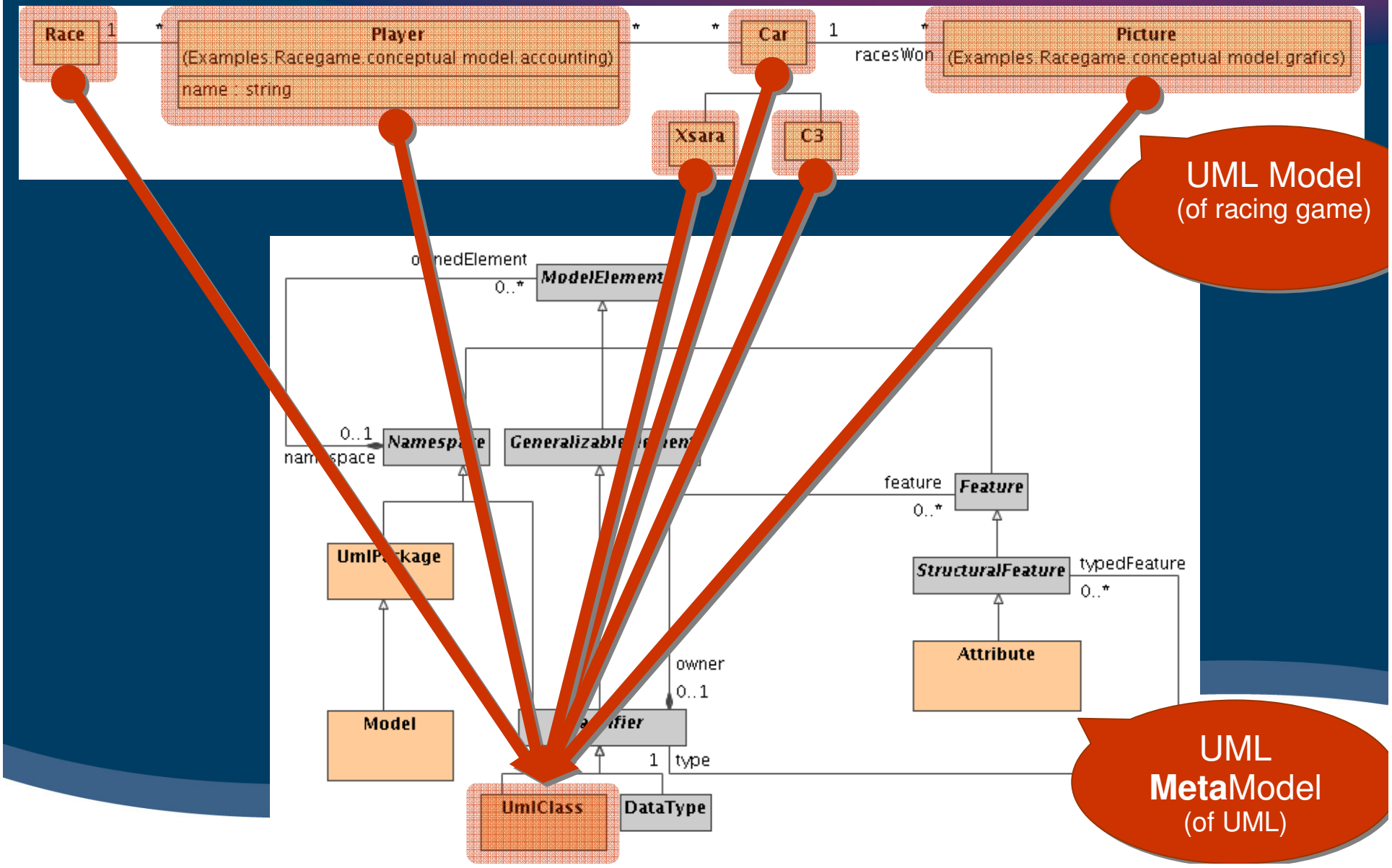
Universiteit Antwerpen

Back to Basics: the UML metamodel



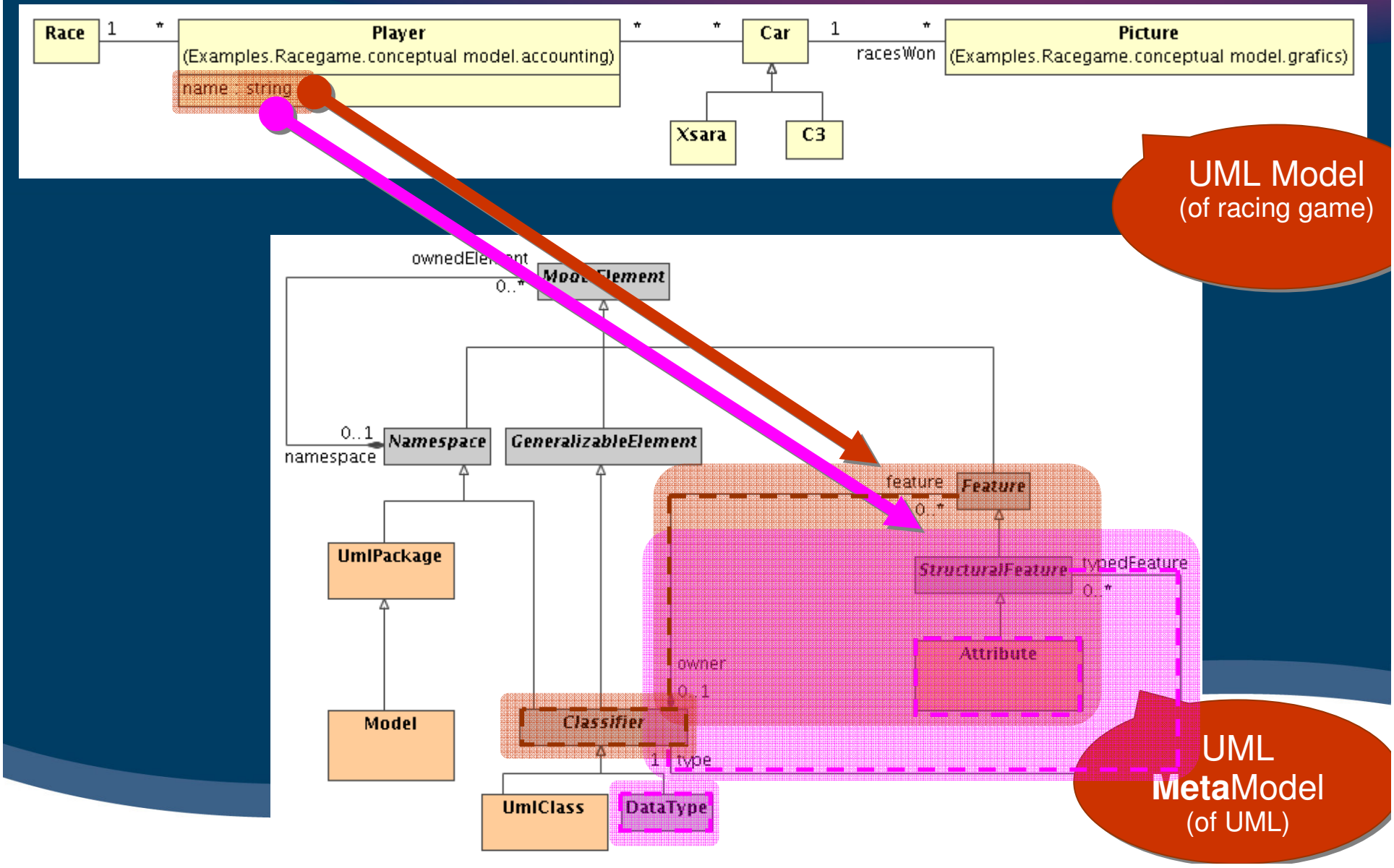
Standard Syntax for Metamodeling

Example: UML Metamodel



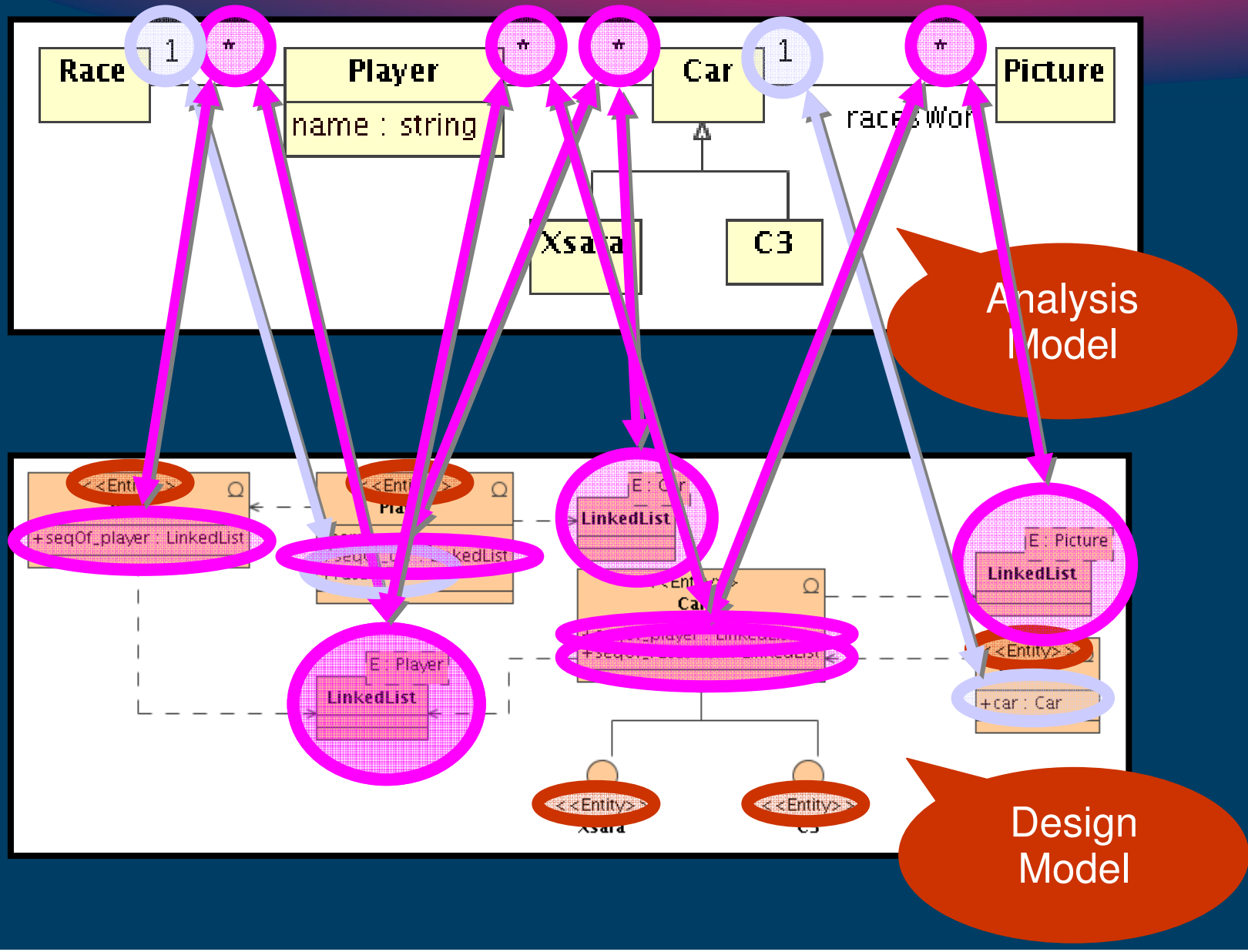
Standard Syntax for Metamodeling

Example: UML Metamodel



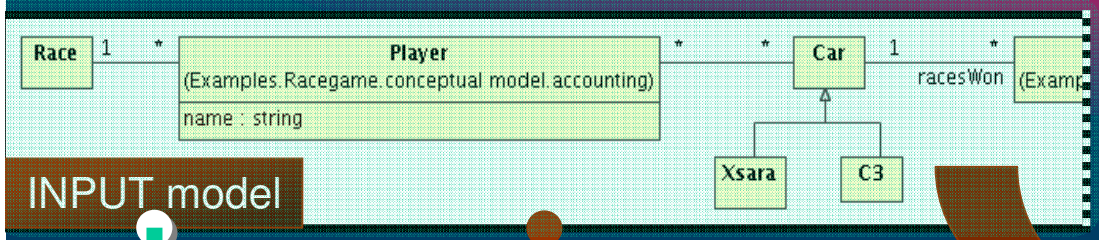
Related Software Models:

Correspondences in Detail

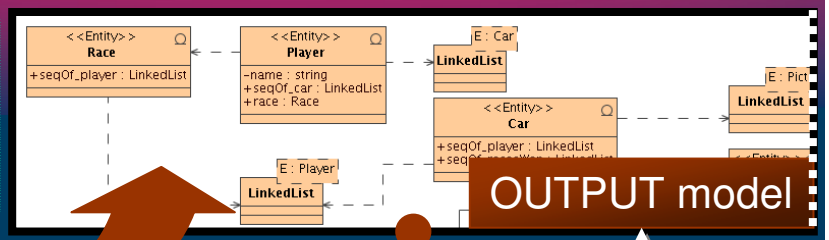


Modeling a **Copy** Operation

Derivation



INPUT model



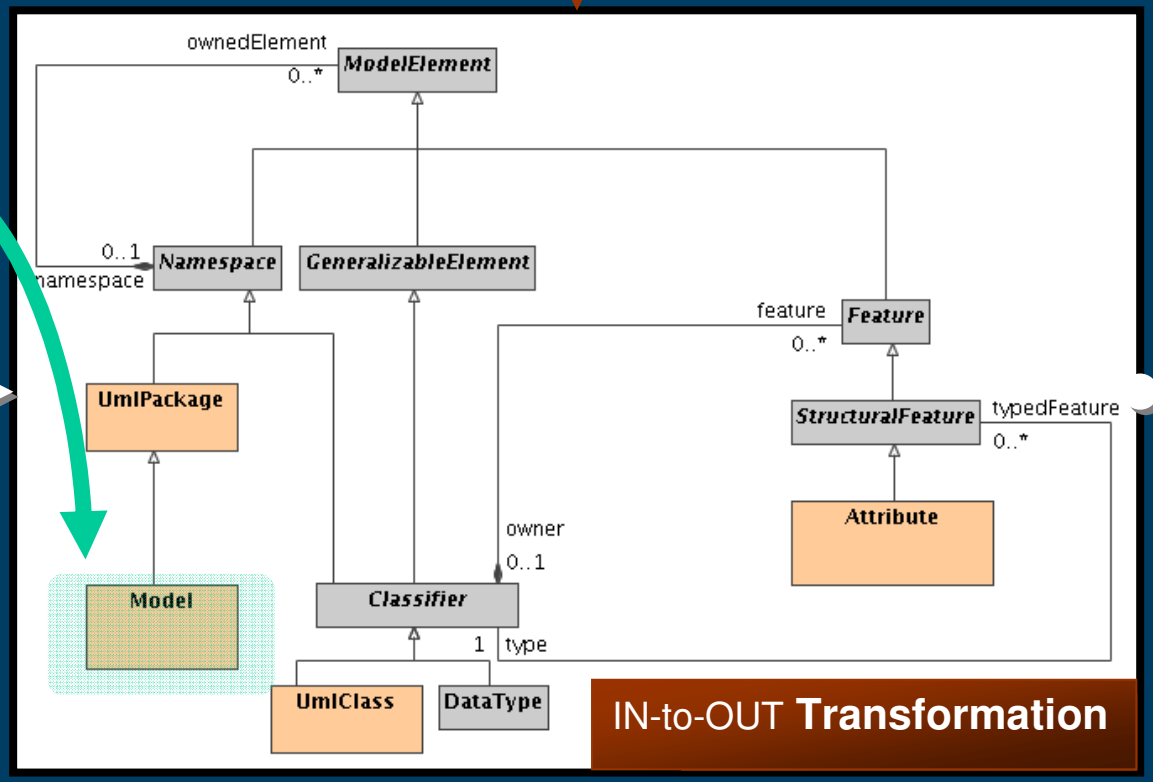
OUTPUT model

conforms to

processed by

conforms to

processed by



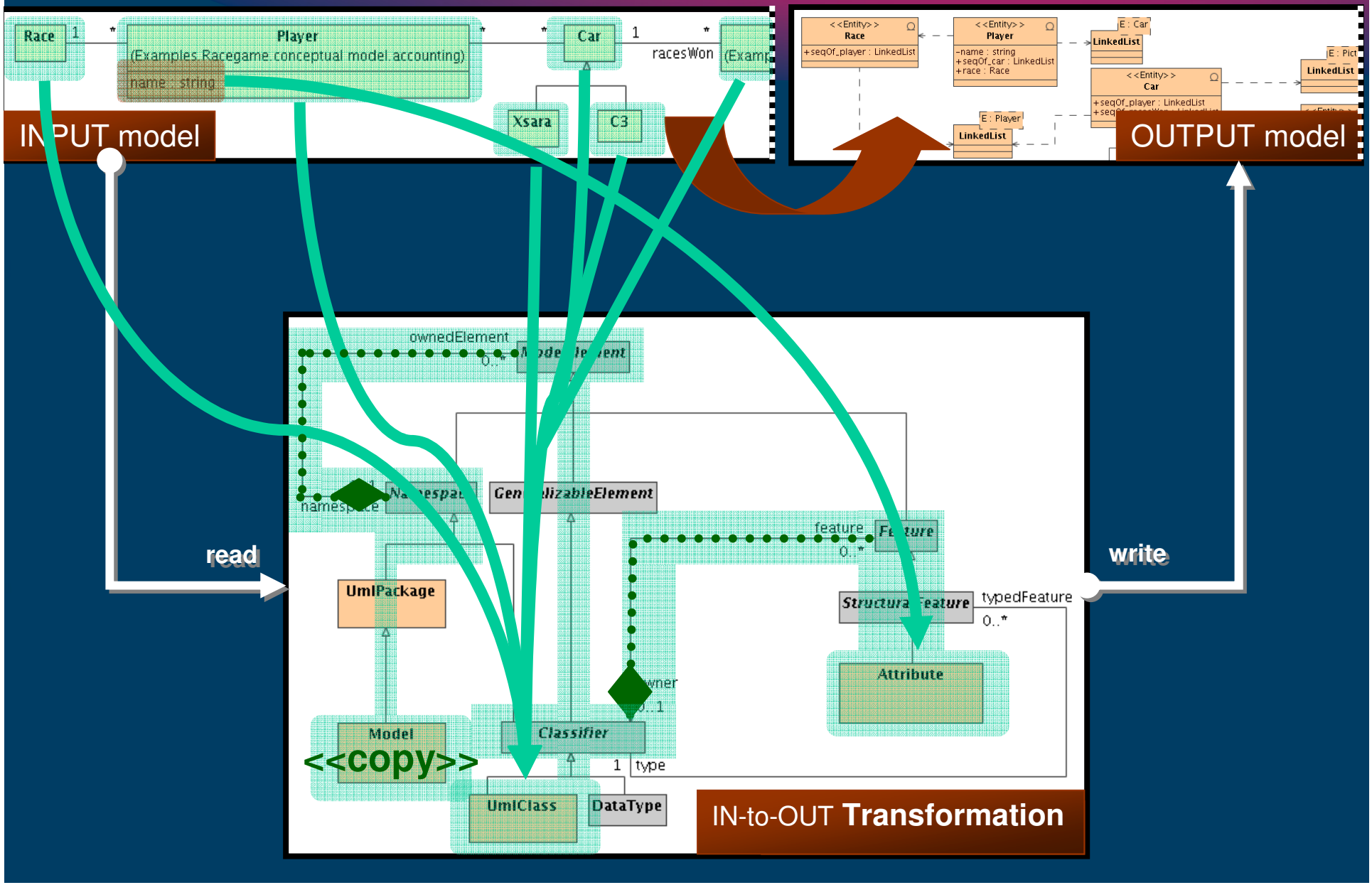
IN-to-OUT **Transformation**

read

write

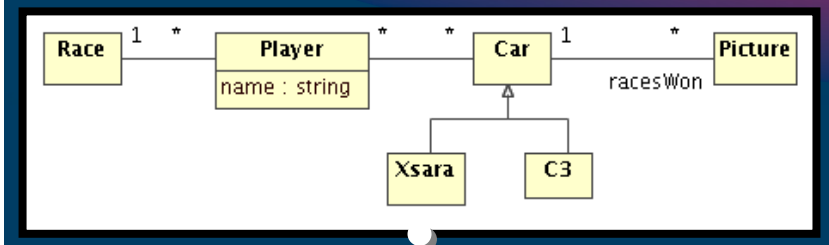
Modeling a **Copy** Operation

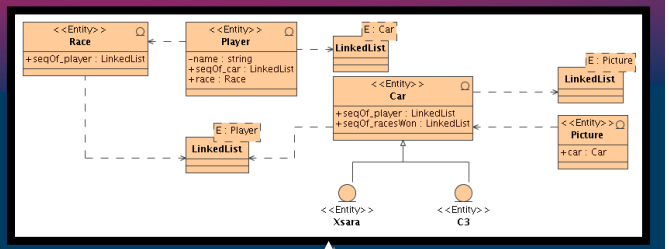
Derivation



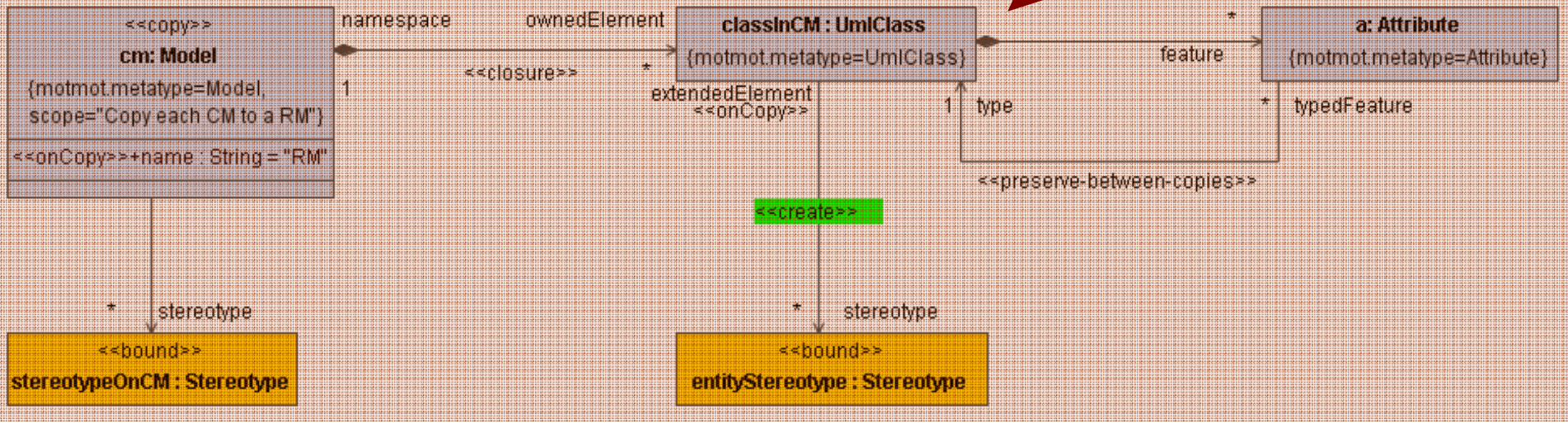
Modeling a **Copy** Operation

Summary



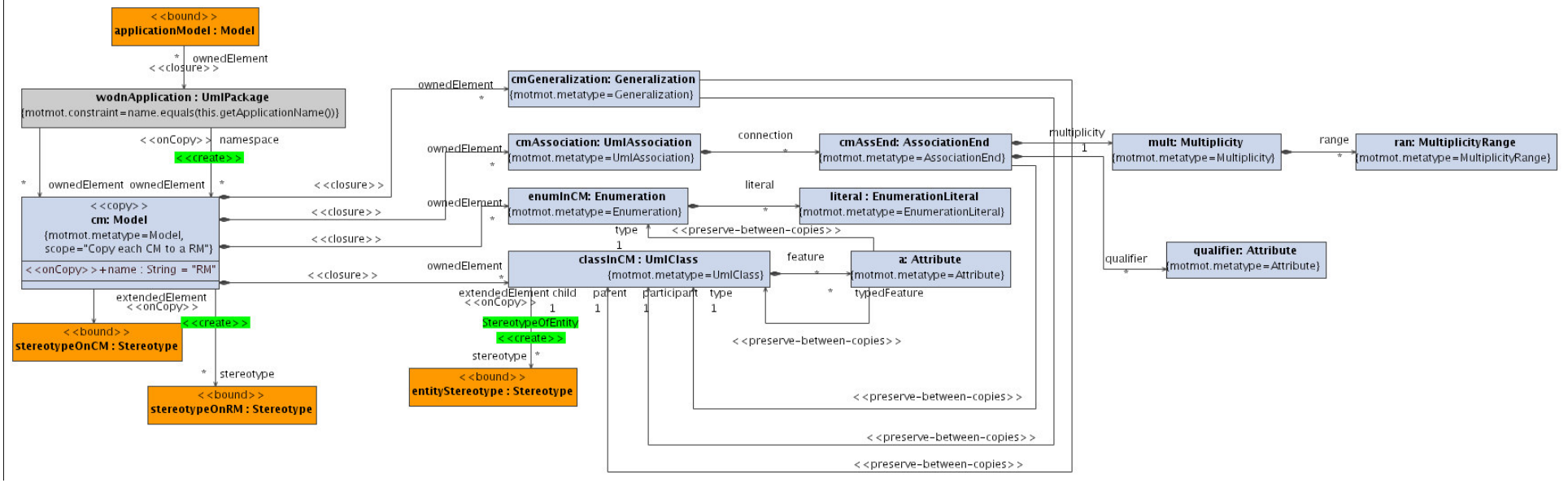


Transformation Modeling Language:
(extension of) so-called **"Story Diagrams"**, in **Standard UML** syntax



Copying: Example IN CONTEXT

- *Complete copy operation in **ONE rewrite rule***
- *Enables one to assess **overall effect***
- *Complexity can be managed using **views** (!)*
- *Only specify WHAT is copied, **don't bother about***
  - » ***Traceability***
  - » ***Properties of metaclasses***
  - » ***…***

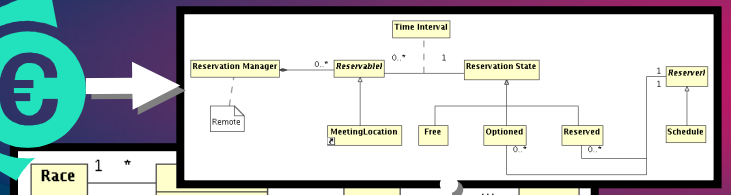


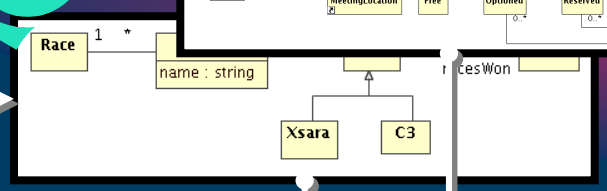
Part 3

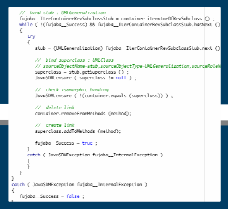
Higher Order Transformations

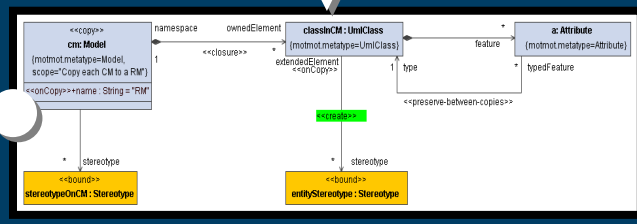
Universiteit Antwerpen

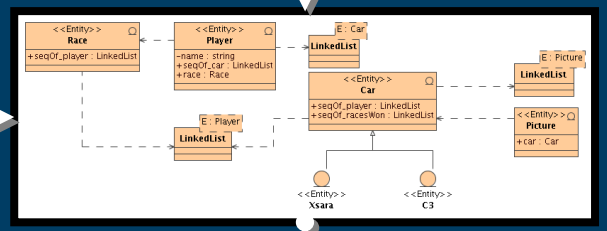
Towards Transformation Code…

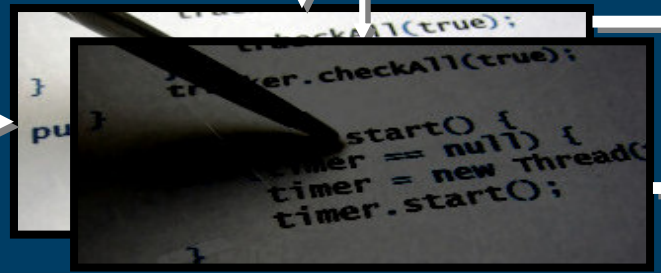












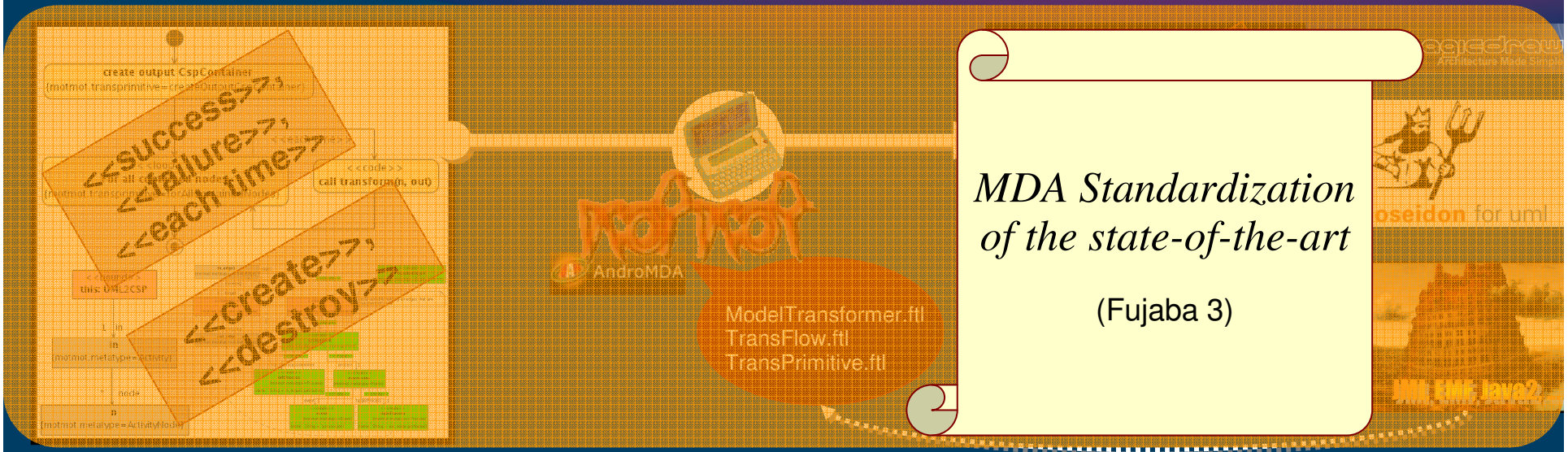
Higher Order Transformations

Case Studies:

1. Story Diagrams
2. **Copying**

Higher Order Transformations

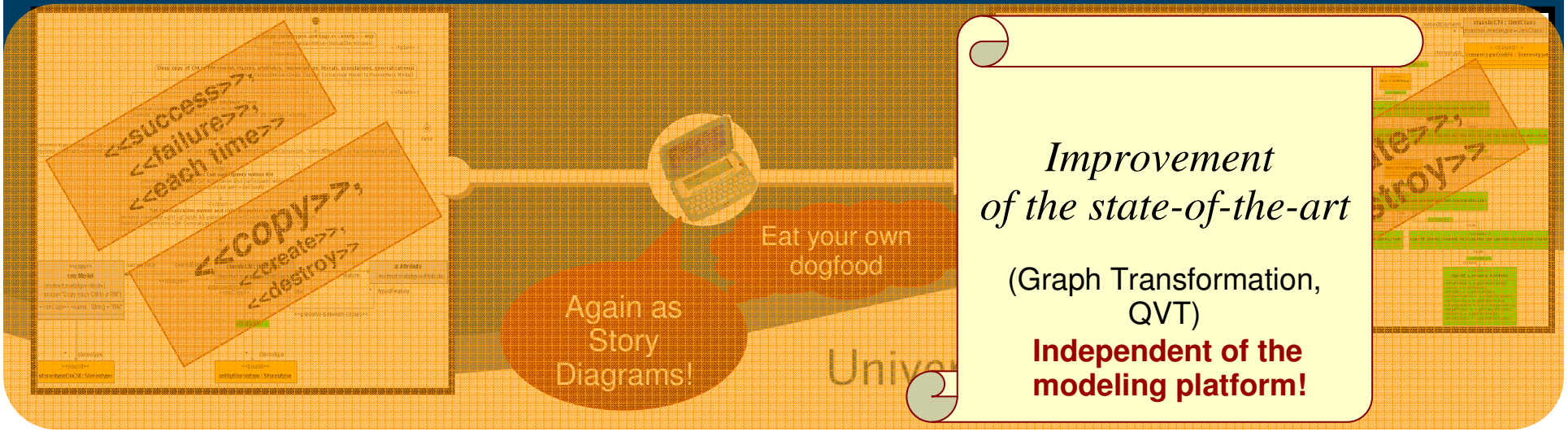
1. From Story Diagrams to JMI Transformation Code



MDA Standardization of the state-of-the-art

(Fujaba 3)

2. From the Copy language to plain Story Diagrams



Improvement of the state-of-the-art

(Graph Transformation, QVT)

Independent of the modeling platform!

Part 4

Current Focus

Universiteit Antwerpen

Hybrid Transformation Modeling

- Combine Implicit and Explicit Rule Scheduling
- Combine TGGs (bidirectional, incremental) with Story Diagrams (control flow)
- …

Summary

- Modeling languages:
  - Do not re-invent the wheel
  - Applies especially to Transformation Modeling!
    - **Views** for **free**!
- **Copying** problem tackled
  - Transformation **Modeling** (not Programming)
- **Higher Order** Transformations
  - Extensible implementation of <<copy>>
  - Easy to integrate with other modeling platforms
- **Challenge Me!**
  - Copy Operator
- **Success** Criteria
  - OUT: Evangelize Story Diagrams, Copy Operator
  - IN: Practical Experience with
    - » DEVS, Alloy
    - » MDE work for SoC