COPE – Automating Model Migration in Response to Metamodel Evolution

Markus Herrmannsdoerfer
Technische Universität München
herrmama@in.tum.de

13th International Conference on Model Driven Engineering Languages and Systems (MoDELS)
4th October 2008
Oslo, Norway

Disclaimer

COPE only supports the coupled evolution of **EMF**-based metamodels and models.

Installation

Requirements

- minimum Java 1.5

Download from USB Stick

- Eclipse 3.5 SR2 Modeling Tools for your platform (Win, Mac, Linux)
- Folder for COPE containing update site and workspaces

Install on your machine

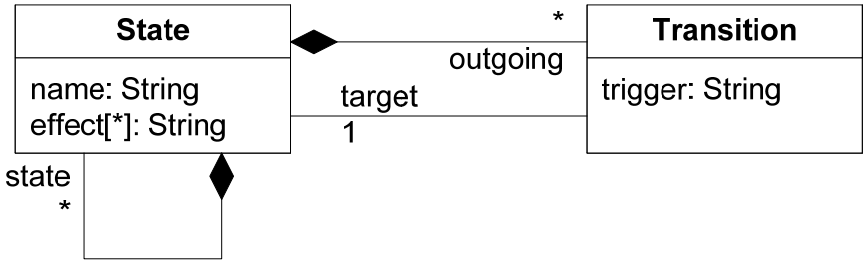
- Unzip Eclipse 3.5 SR2 Modeling Tools
- Add to eclipse.ini: "*-XX:MaxPermSize=256m*"
- Start executable "*eclipse*" in the extracted eclipse folder
- Choose the "*metamodelerWorkspace*" as workspace location
- Install COPE from the update site (Help -> Install new Software …)

Agenda

1. **Introduction**
   - principles behind COPE
2. **Guided Tool Usage**
   - simple example
3. **Independent Tool Usage**
   - more difficult task
   - evolve your own metamodel
4. **Wrap Up**
   - questions and feedback

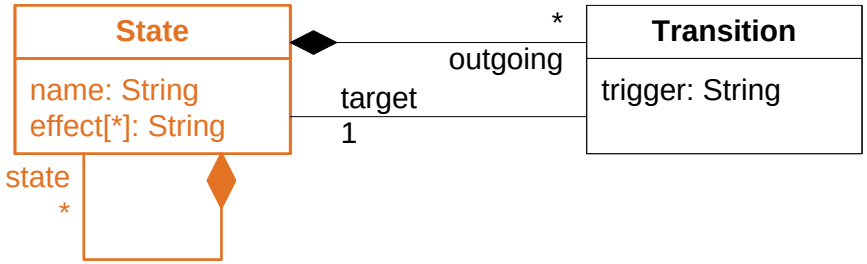
Problem – Running Example

Metamodel

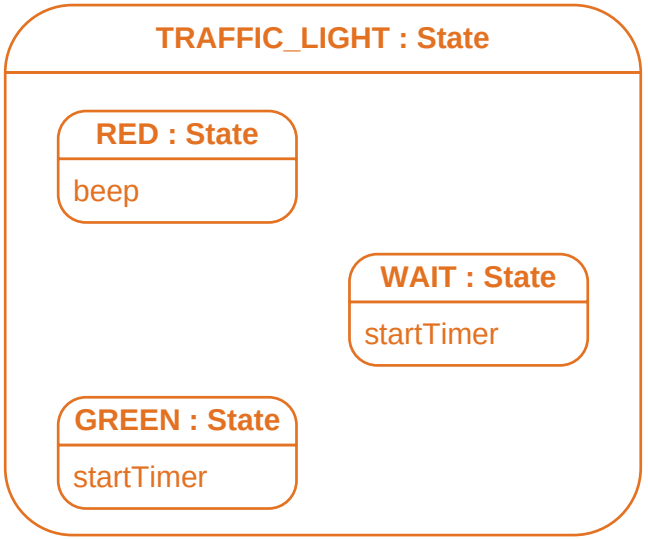


Problem – Running Example

Metamodel

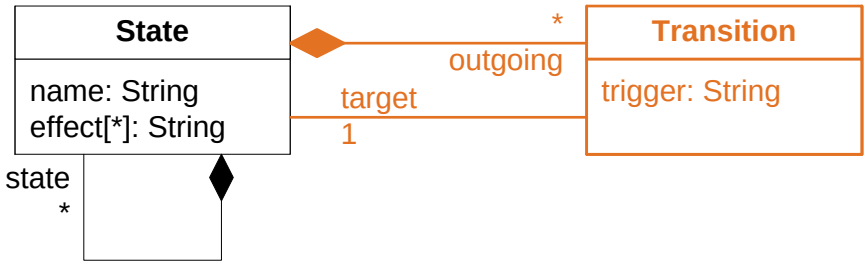


Model

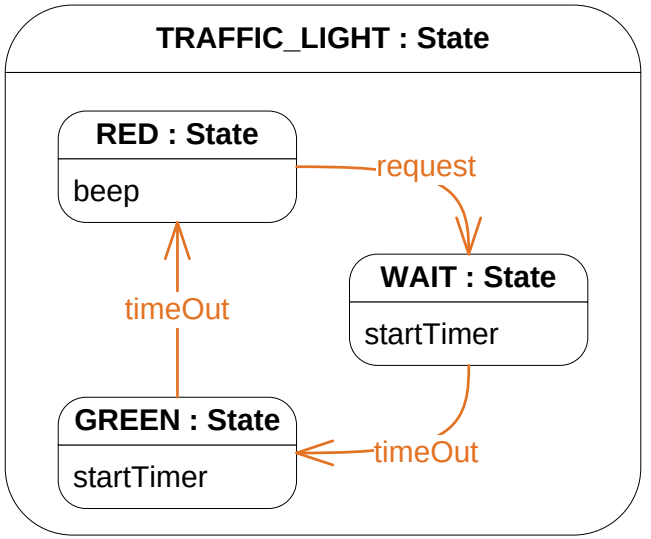


Problem – Running Example

Metamodel

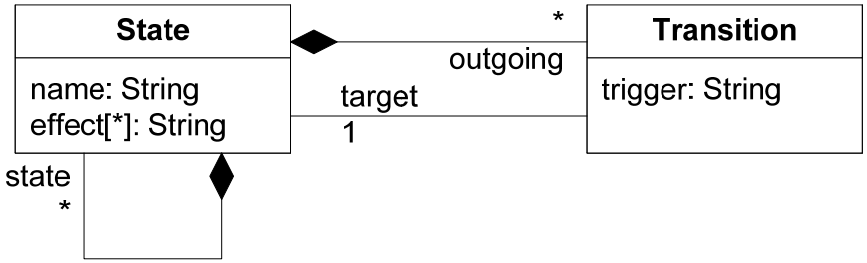


Model

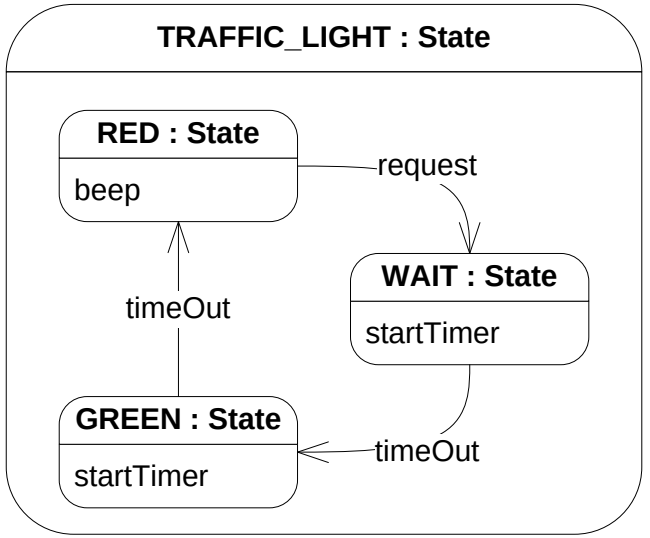


Problem – Running Example

Metamodel

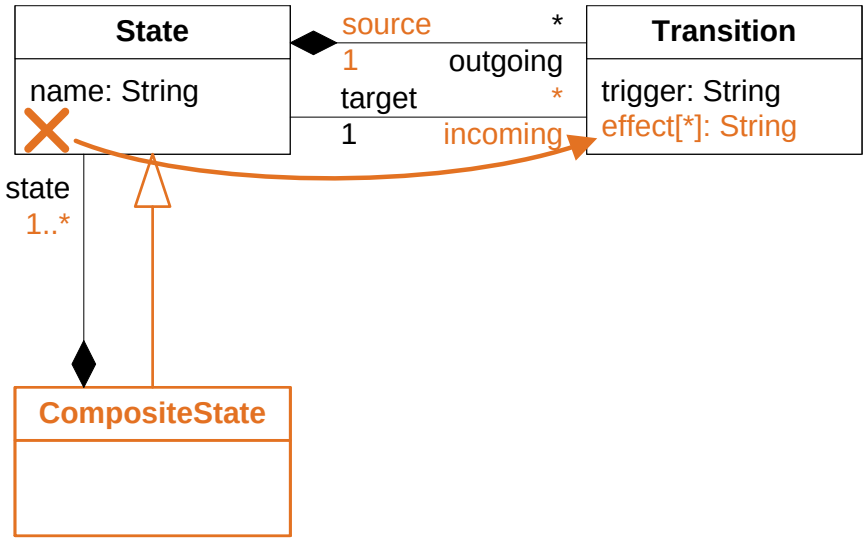


Model

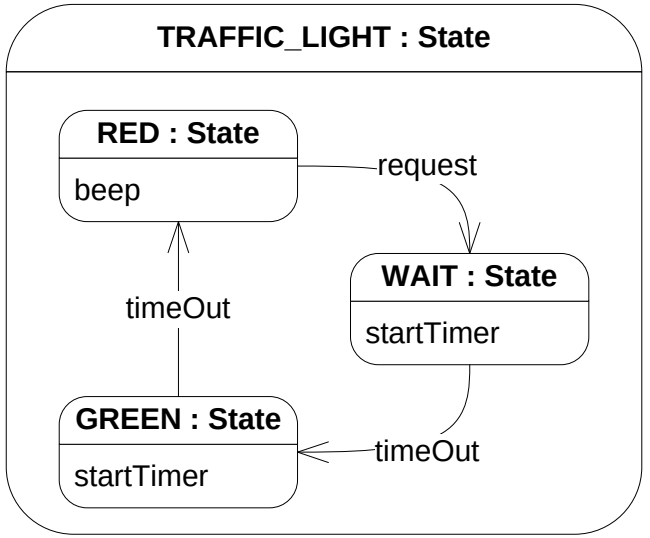


Problem – Metamodel Adaptation

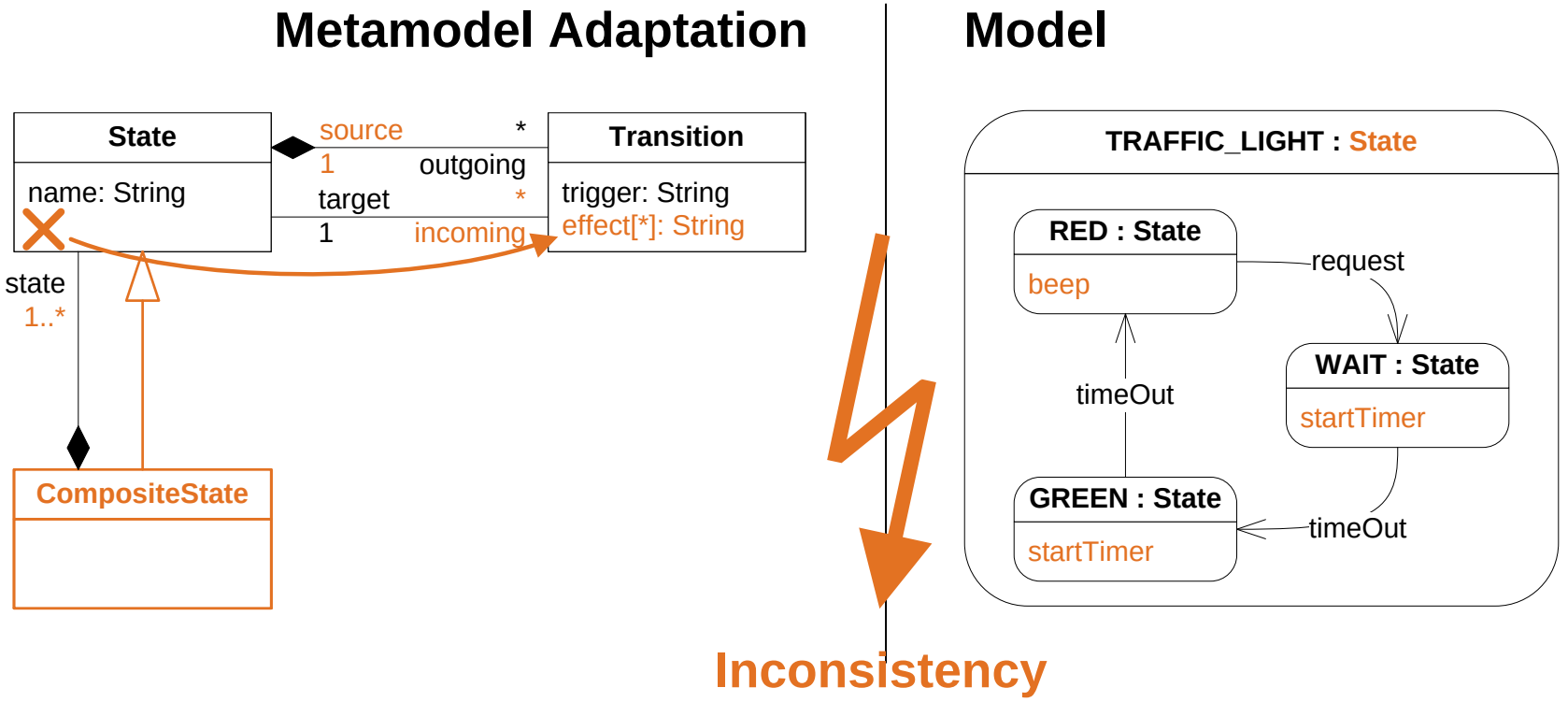
Metamodel Adaptation



Model

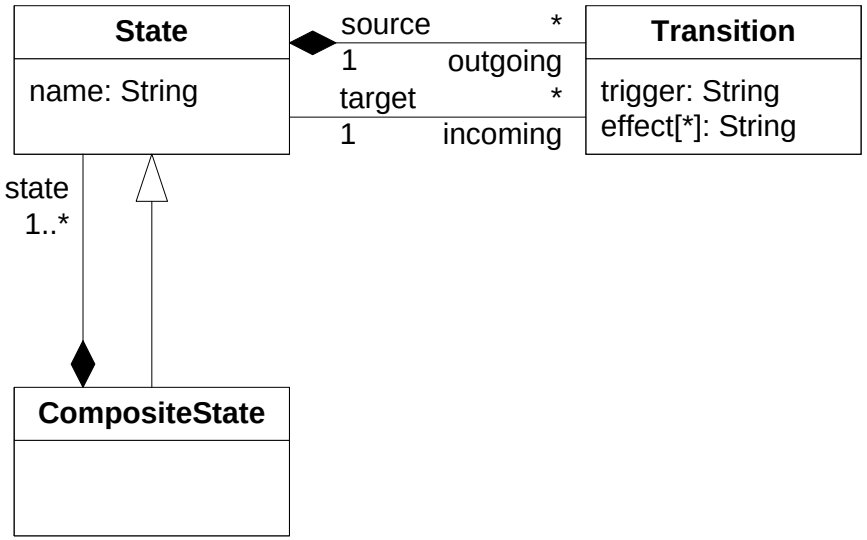


Problem – Metamodel Adaptation

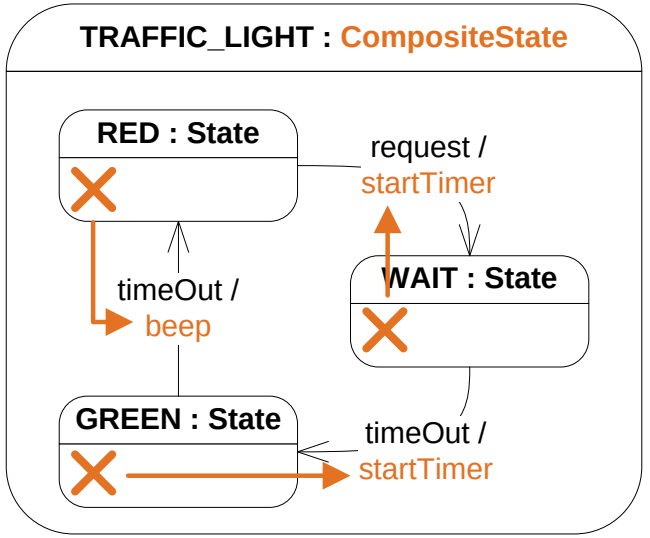


Problem – Model Migration

Metamodel



Model Migration



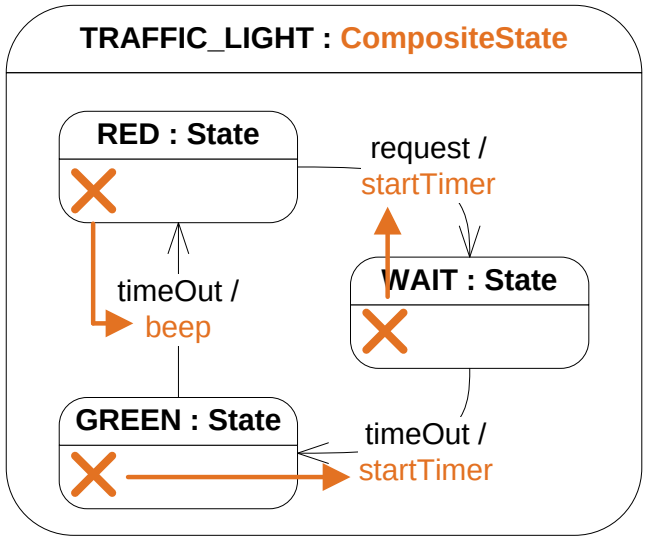
Problem – Model Migration

Manual Migration

- tedious
- error-prone

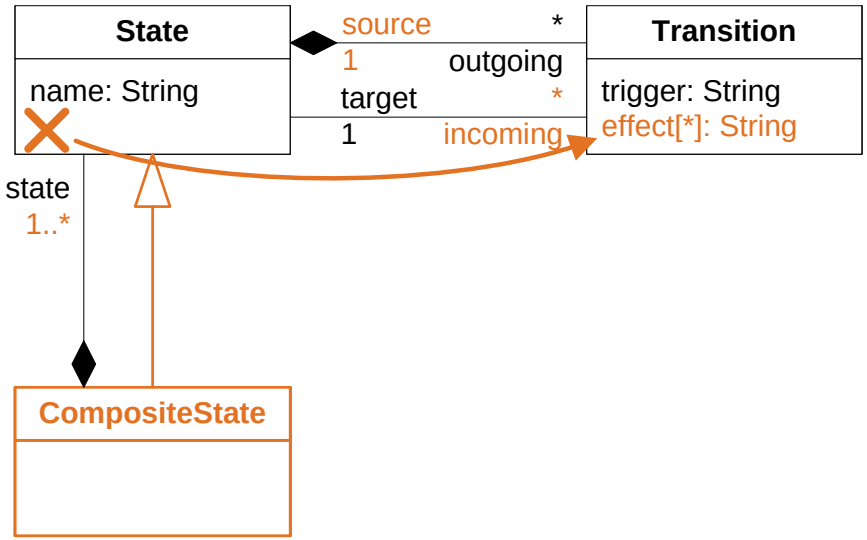
⇒ Tool support required

Model Migration

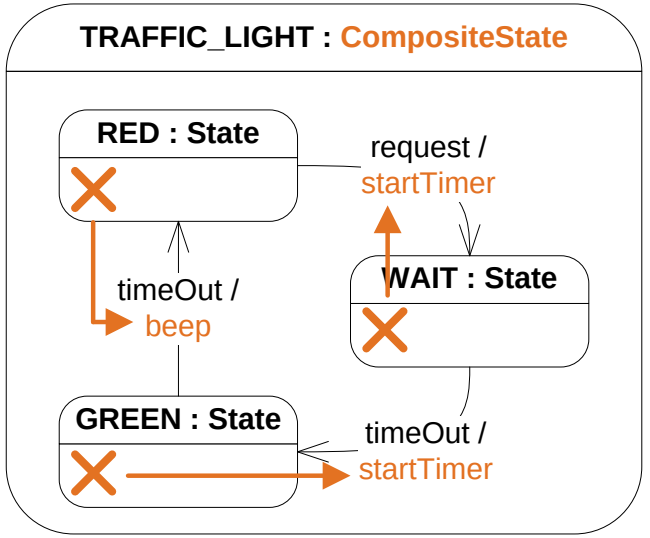


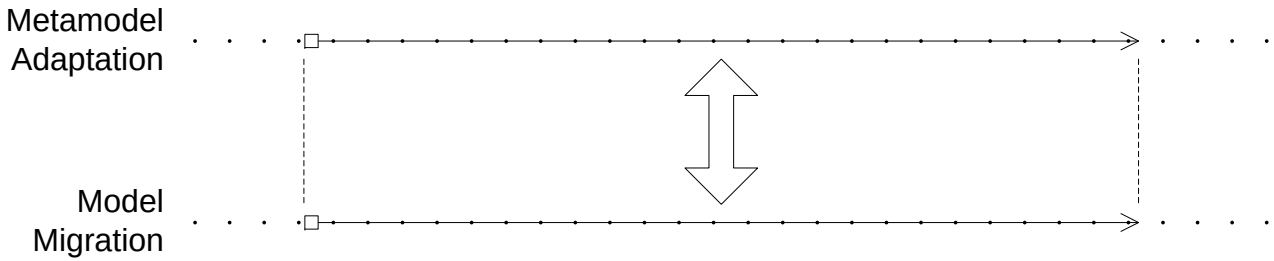
Problem – Coupled Evolution

Metamodel Adaptation



Model Migration





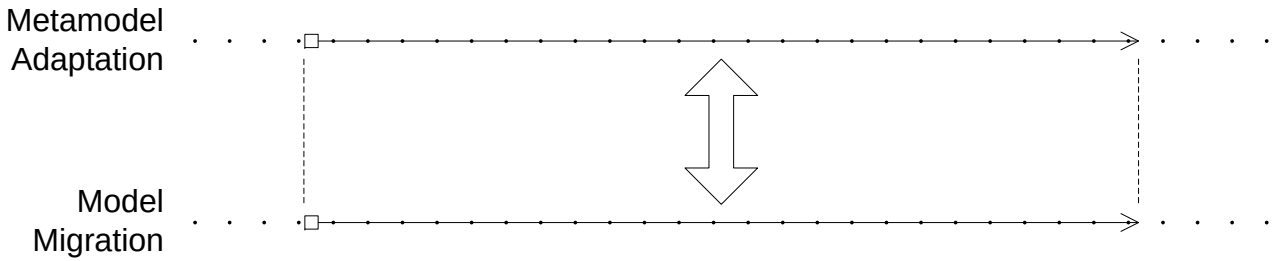
Solution – Operation-based Coupled Evolution

COPE: Language and Tool to specify the Coupled Evolution

- Metamodel Adaptation
- Reconciling Model Migration

Main benefits

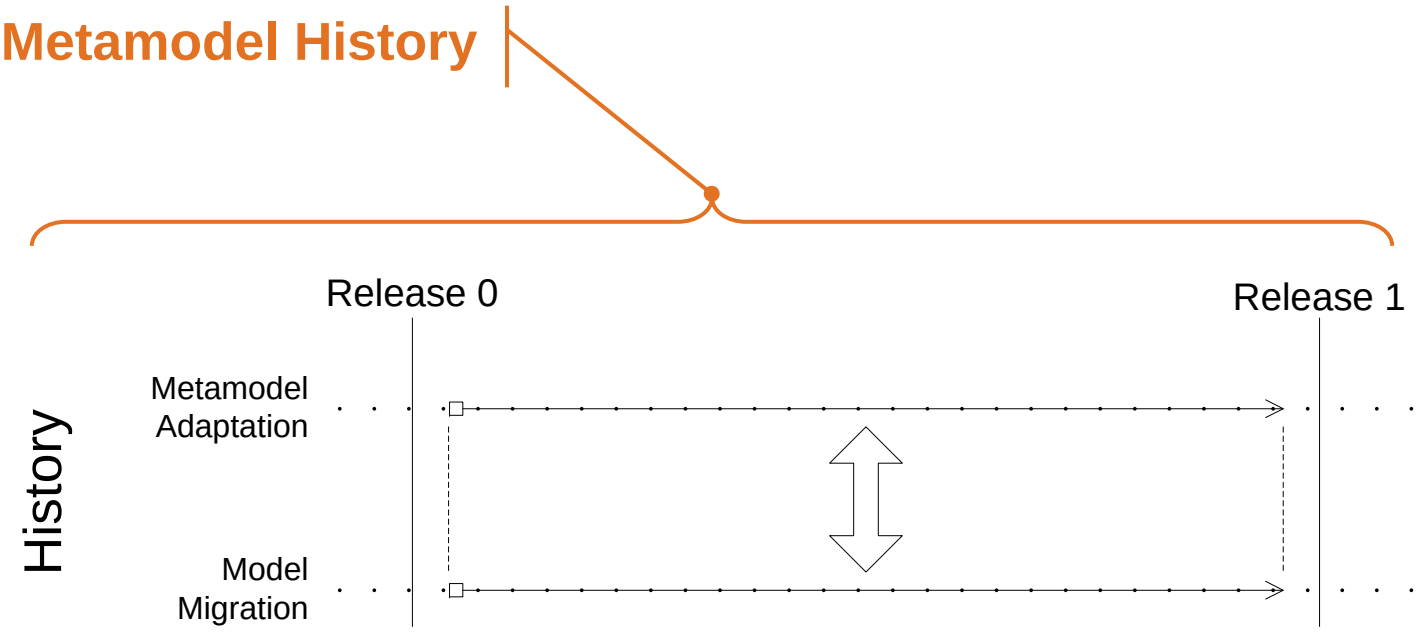
- Preserving the intended Model Migration
- Automating Model Migration as far as possible



Solution – 1st Principle: Recording the Coupled Evolution

Explicit Metamodel History

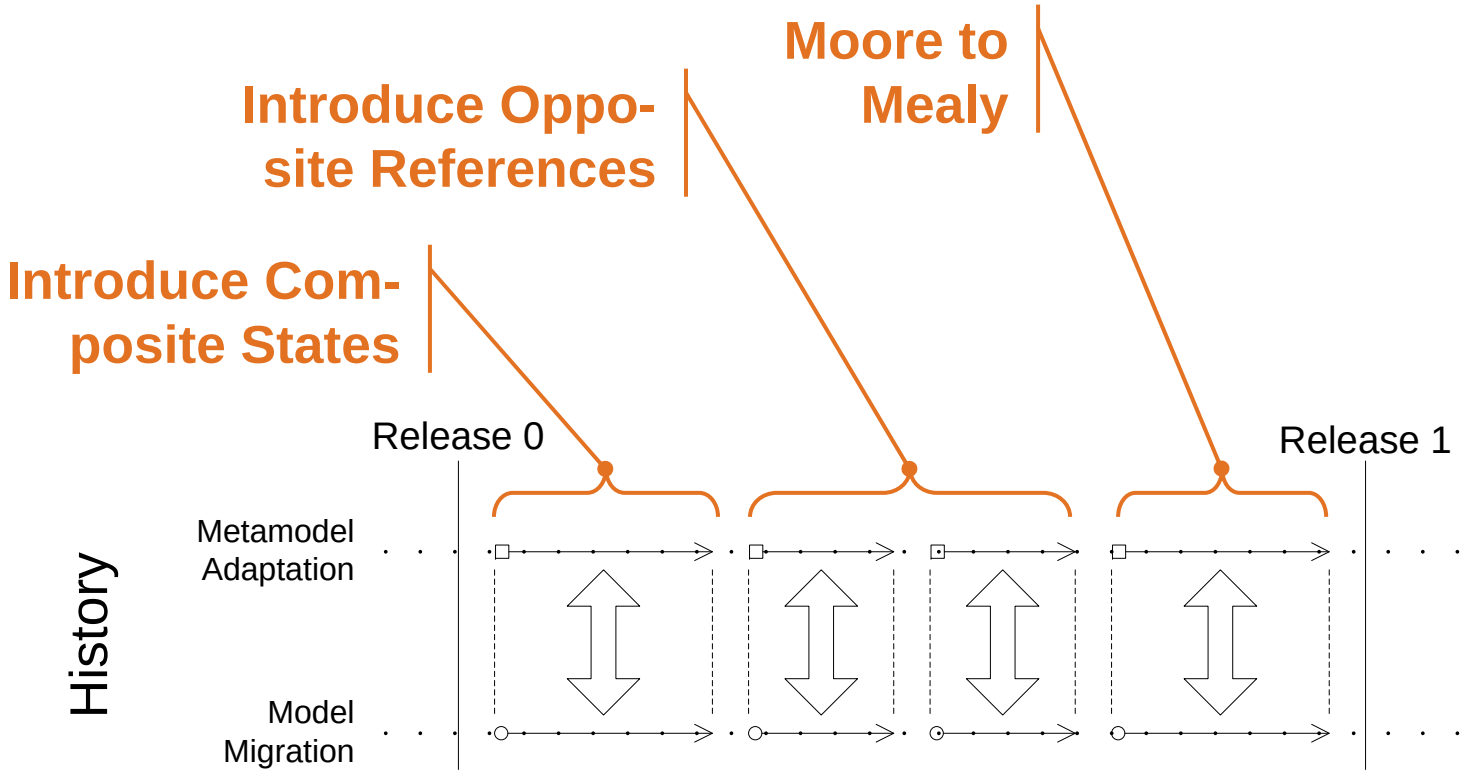
- Recording the intended Model Migration together with the Metamodel Adaptation
- Migration of models not under control of the metamodel developers



Solution – 2nd Principle: Incremental Coupled Evolution

Modularization of Coupled Evolution into Coupled Operations

- Support for typical evolution process
- Scalability through divide and conquer



Solution – 3rd Principle: Specifying only the Difference

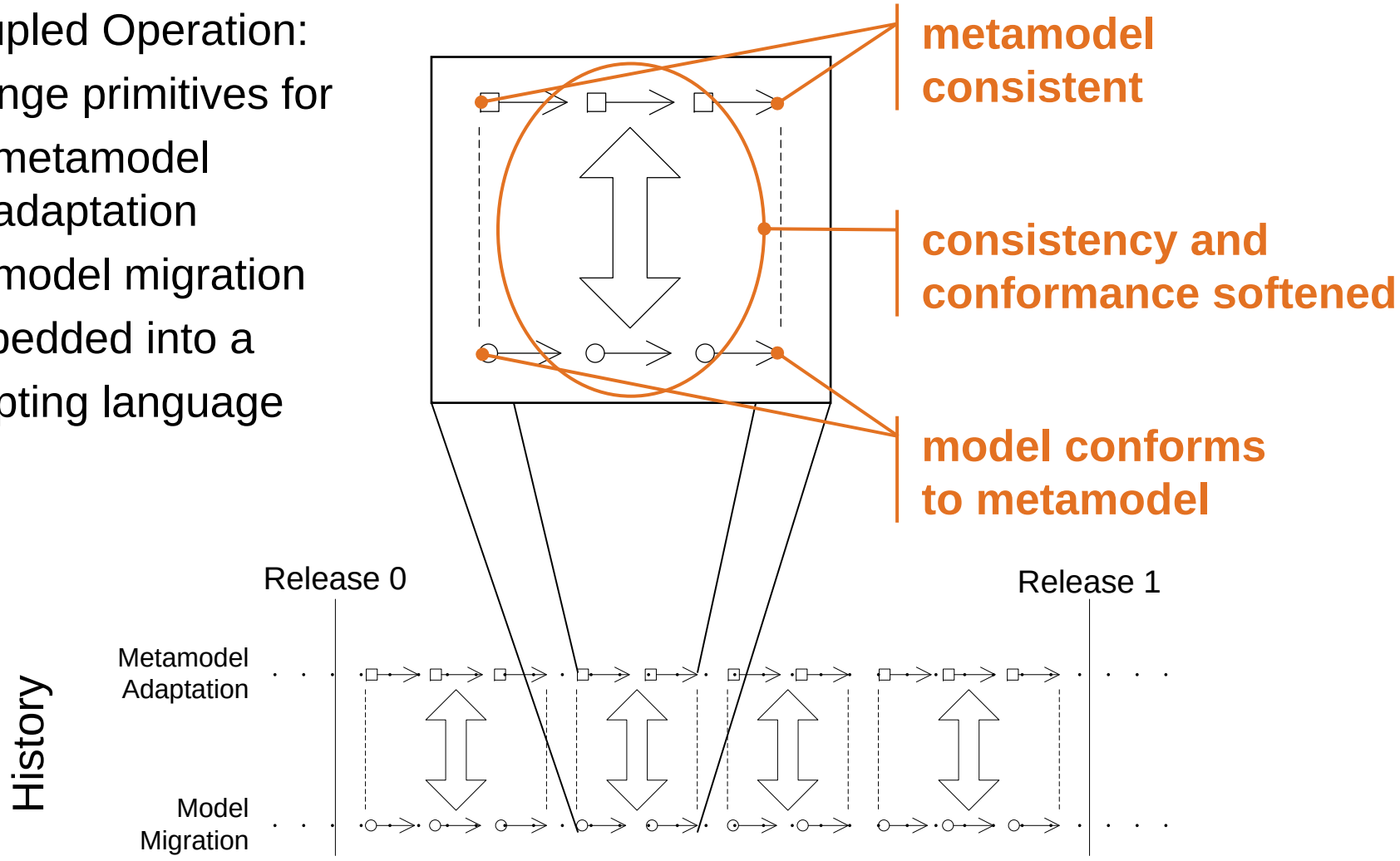
Coupled Operation:

change primitives for

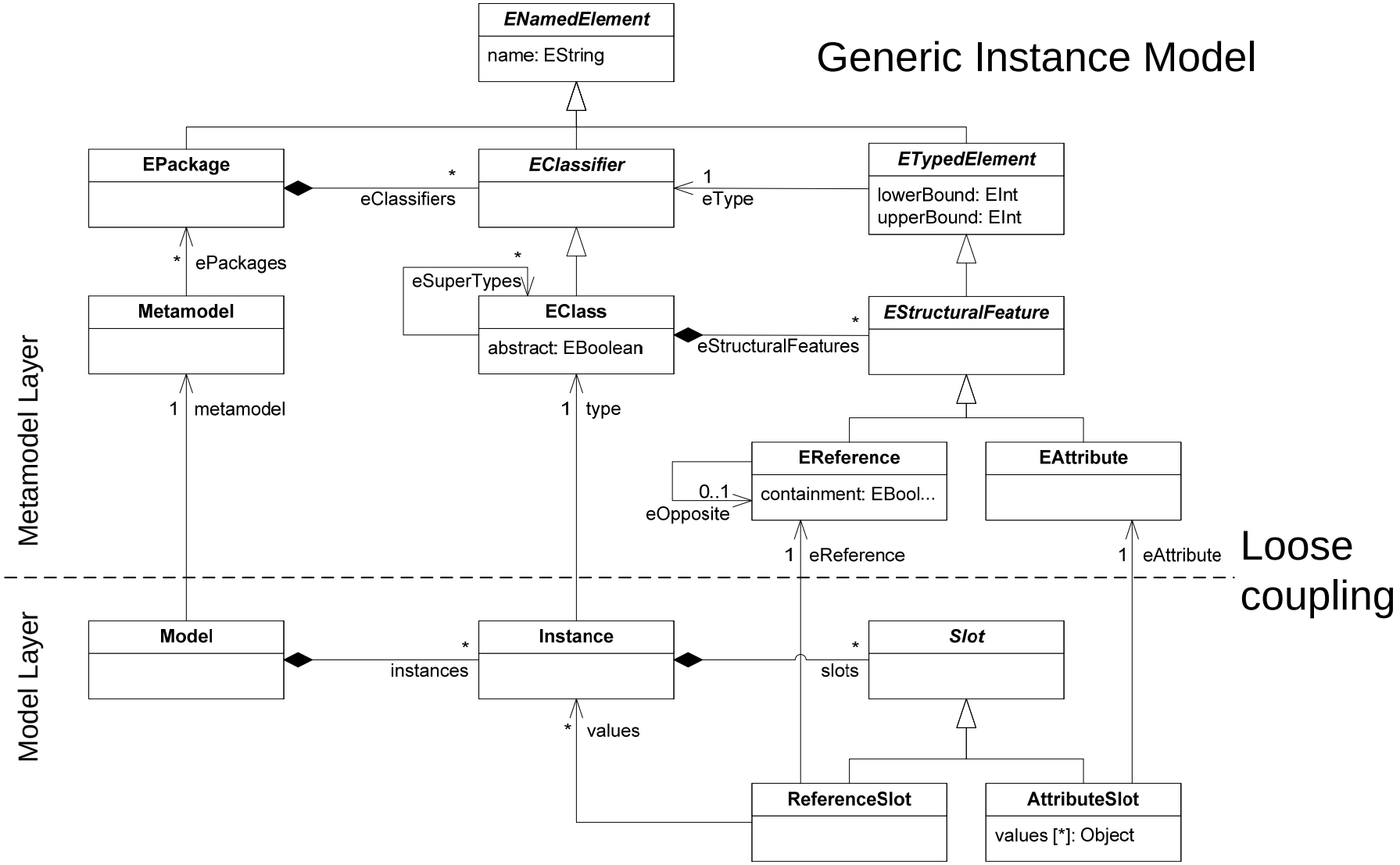
- metamodel adaptation
- model migration

embedded into a

scripting language



Solution – 3rd Principle: Specifying only the Difference



Solution – 3rd Principle: Specifying only the Difference

Primitives for Model Migration

- **querying**
  - «class».instances / allInstances
  - «instance».get(«feature») / «instance».«featureName»
  - «instance».getInverse(«reference»)
- **modification**
  - «class».newInstance() / «instance».delete()
  - «instance».migrate(«class»)
  - «instance».set(«feature», «value») / «instance». «featureName» = «value»
  - «instance».unset(«feature»)

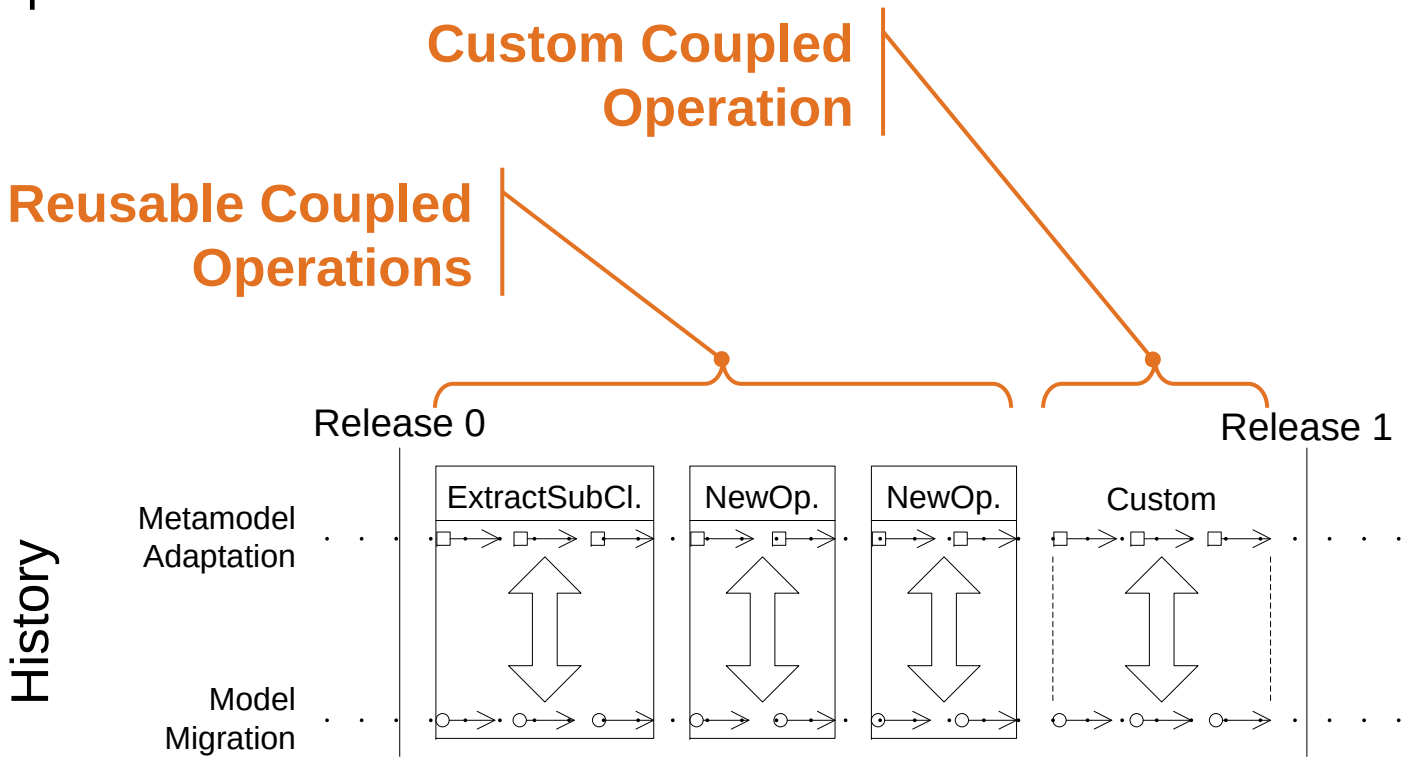
cope_language.pdf

http://cope.in.tum.de/pmwiki.php?n=Primitives.Main

Solution – 4th Principle: Reuse and Expressiveness

Requirements for Coupled Evolution [Models 2008]

- Reuse of recurring migrations ⇒ Reusable Coupled Operation
- Expressiveness for complex migrations ⇒ Custom Coupled Operation



Solution – 4th Principle: Reuse and Expressiveness

Library of Reusable Coupled Operations (68)

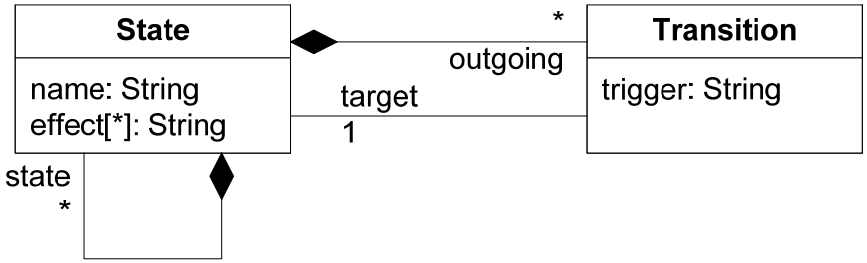
- **Structural** Primitives (12): Create / Delete Metamodel Elements
- **Non-Structural** Primitives (9): Modify Metamodel Elements
- **Specialization / Generalization** Operations (9): Specialize / Generalize Metamodel Elements
- **Inheritance** Operations (11): Move Along Inheritance Hierarchy
- **Delegation** Operations (11): Move Along Association Structure
- **Replacement** Operations (9): Switch between equivalent Metamodel Elements
- **Merge / Split** Operations (7): Merge / Split Metamodel Elements of the same Type

cope_library.pdf

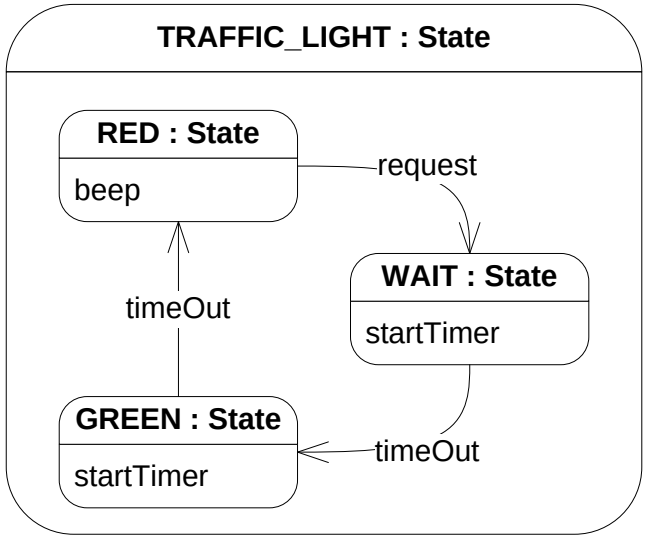
http://cope.in.tum.de/pmwiki.php?n=Operations.Main

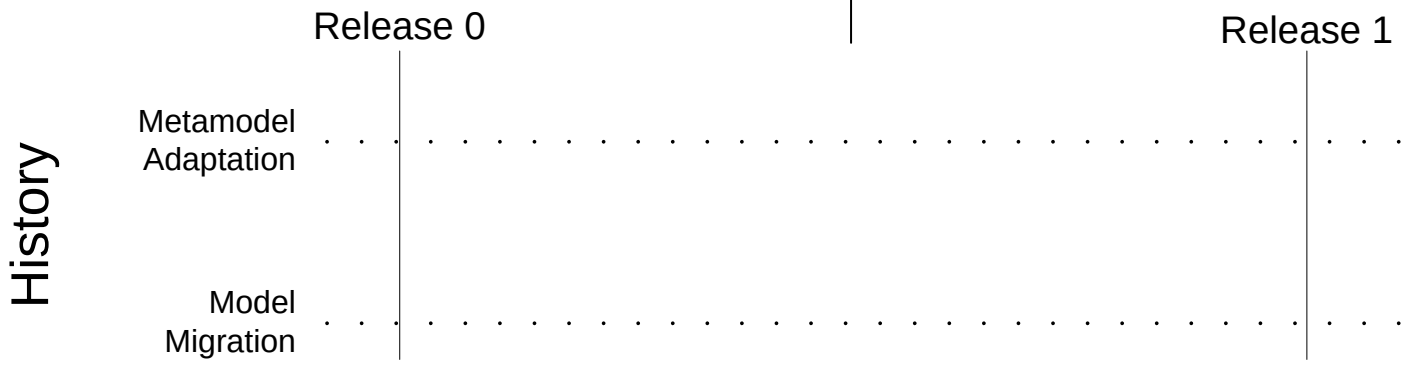
Solution – Running Example

Metamodel



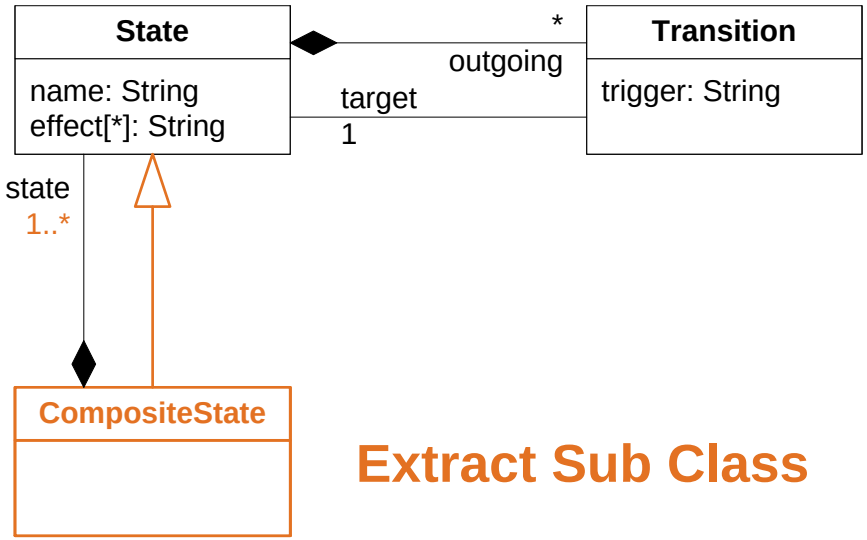
Model



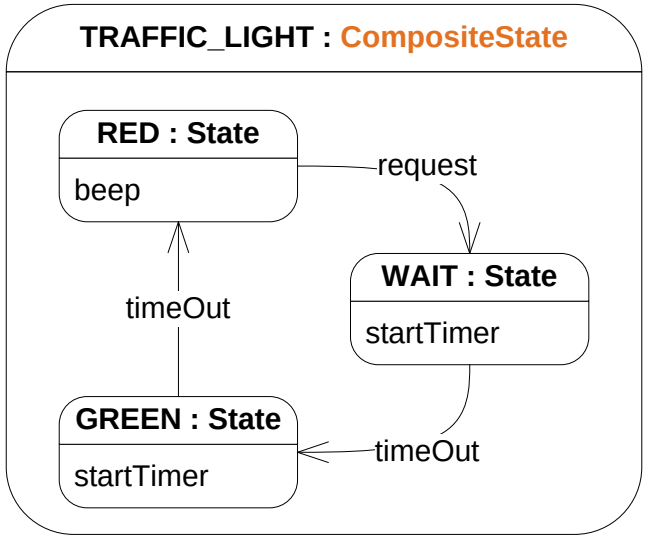


Solution – Introduce Composite States

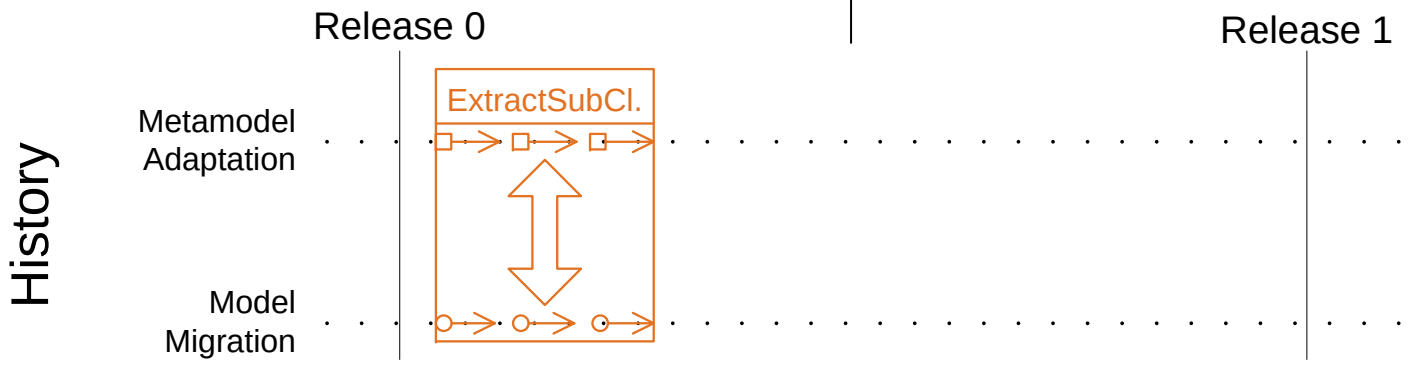
Metamodel Adaptation



Model Migration

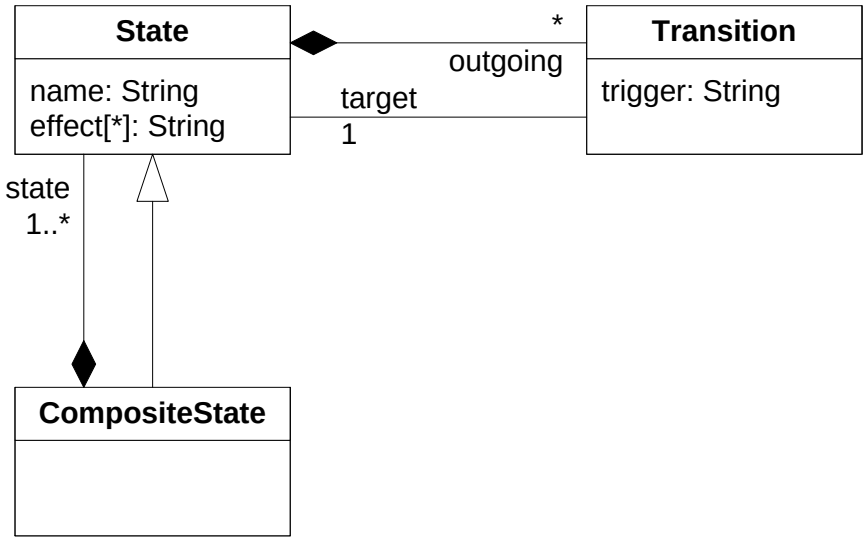


Extract Sub Class

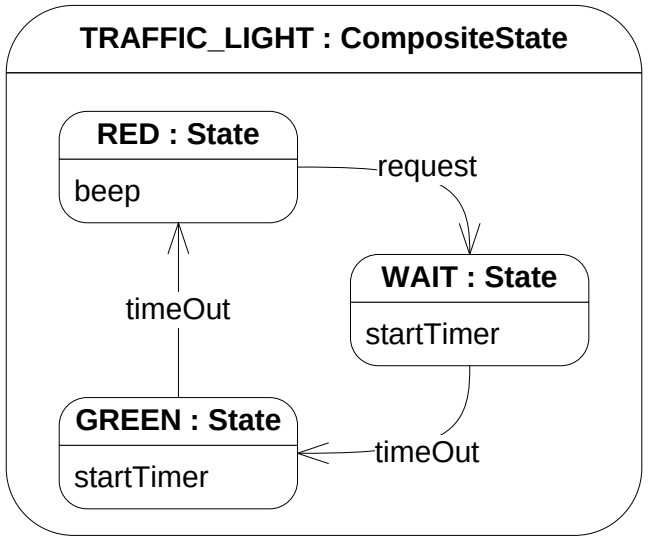


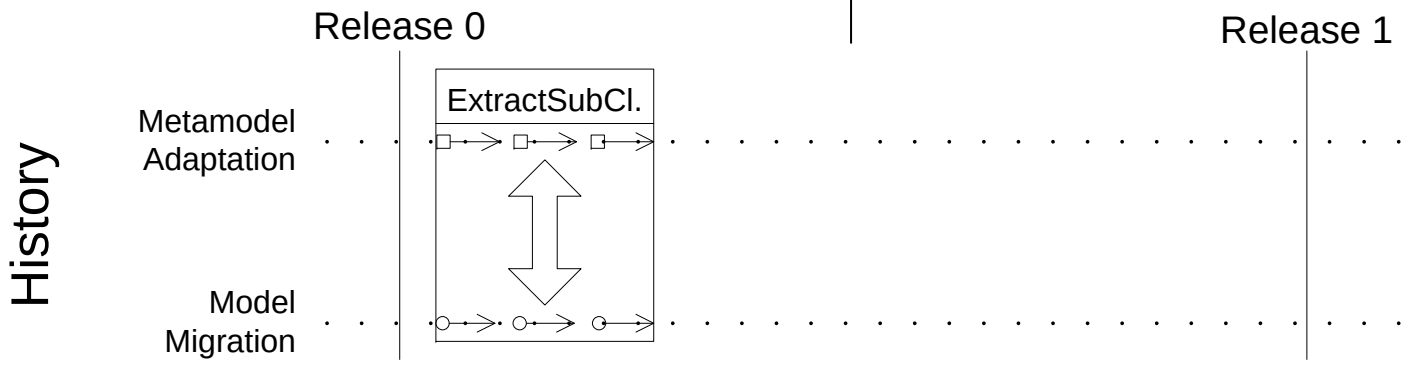
Solution – Introduce Composite States

Metamodel



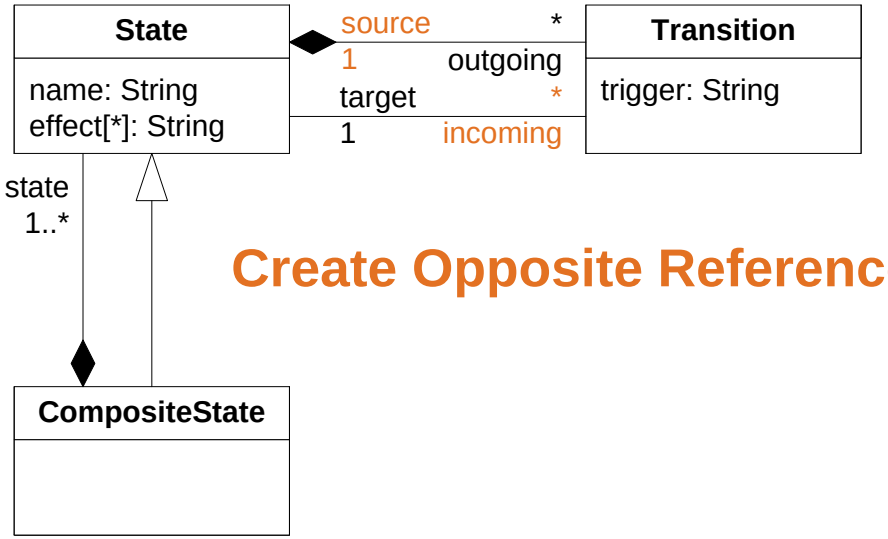
Model





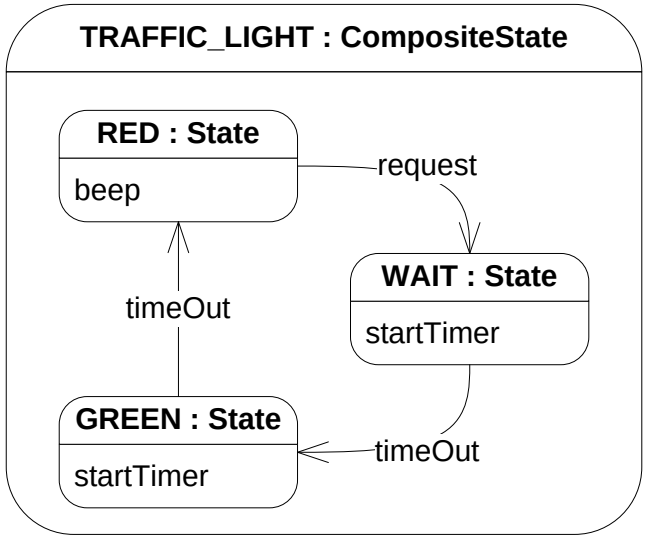
Solution – Introduce Opposite References

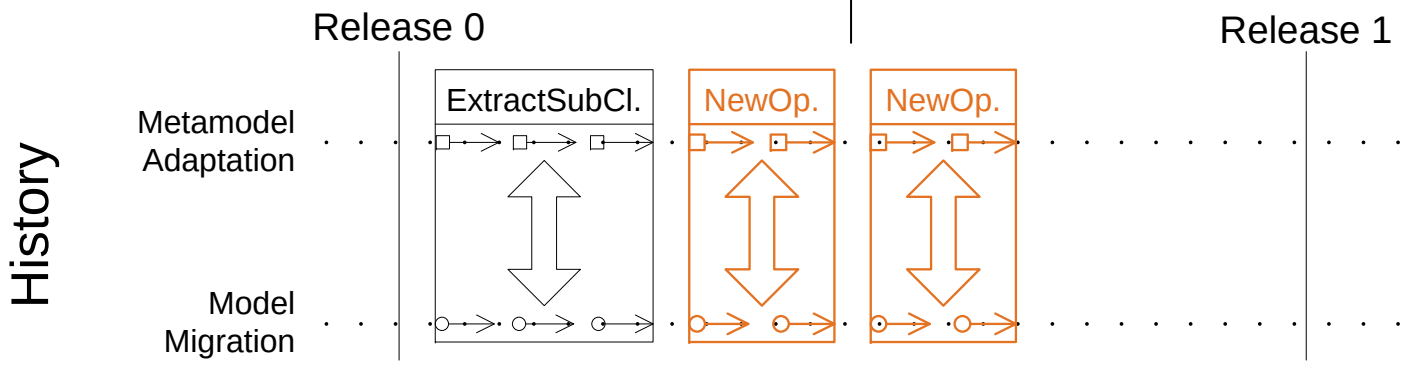
Metamodel Adaptation



Create Opposite Reference

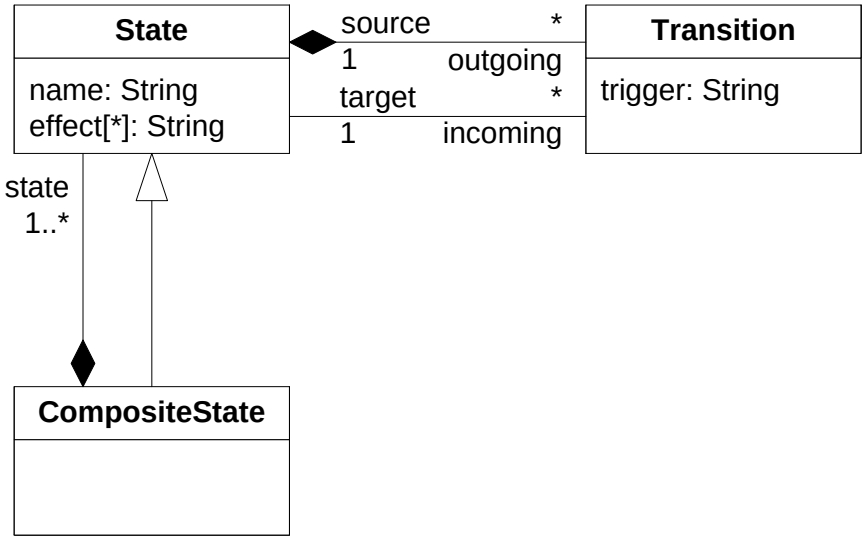
Model Migration



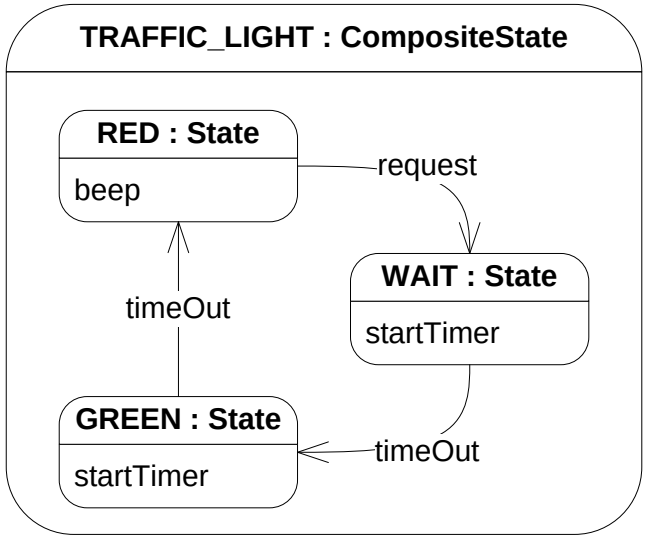


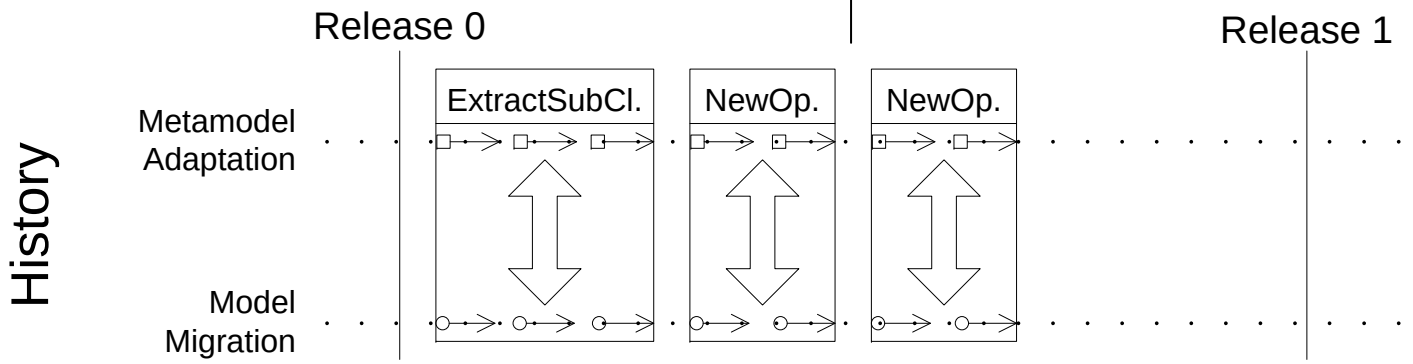
Solution – Introduce Opposite References

Metamodel



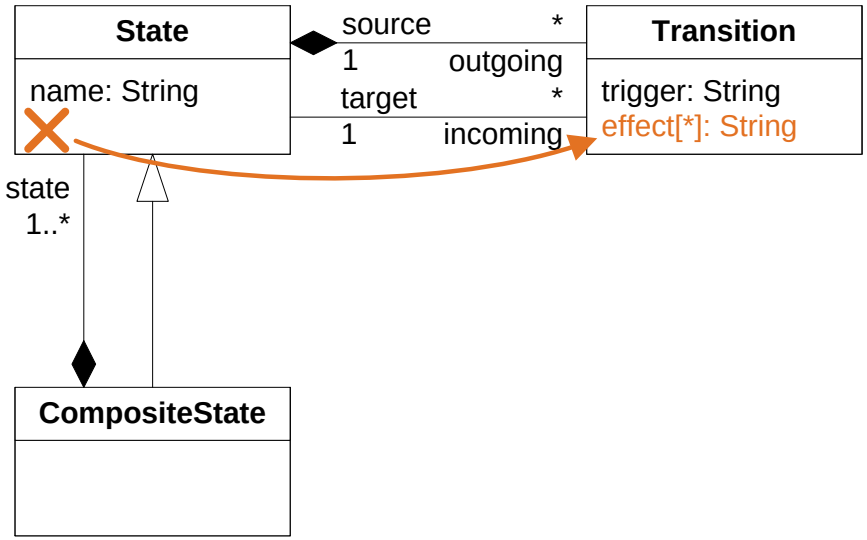
Model



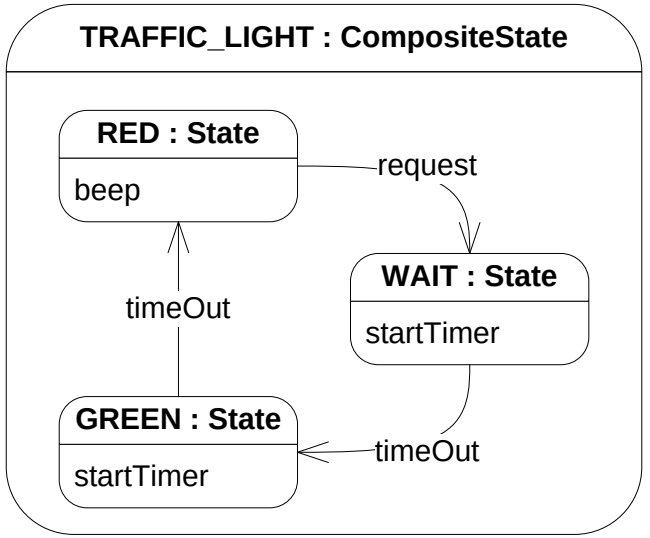


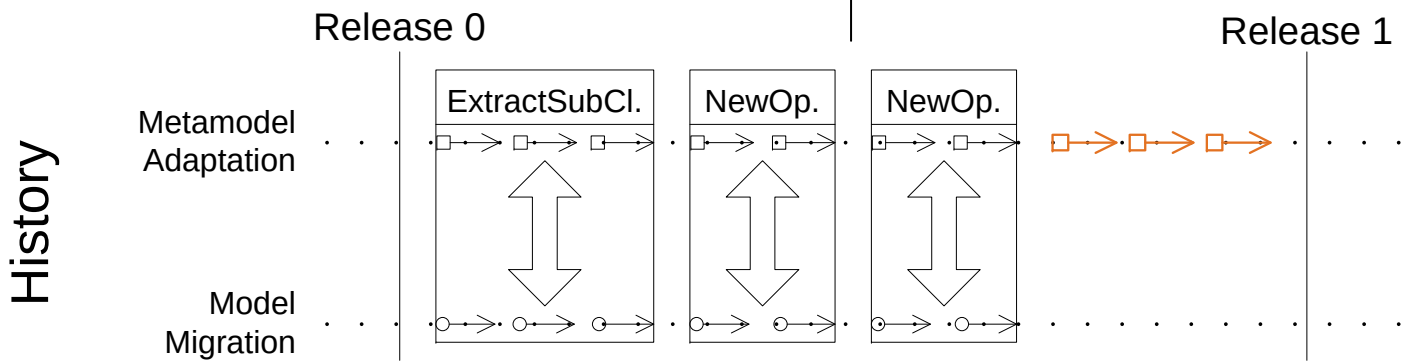
Solution – Moore to Mealy

Metamodel Adaptation



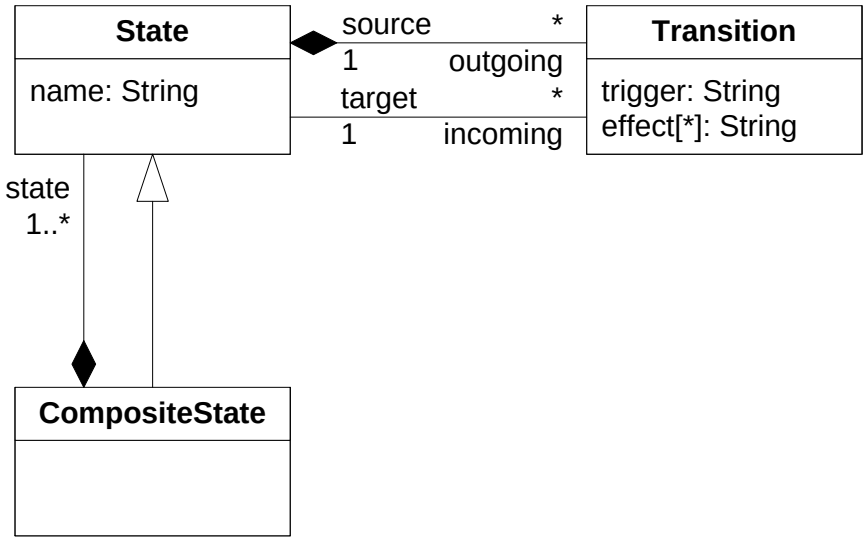
Model



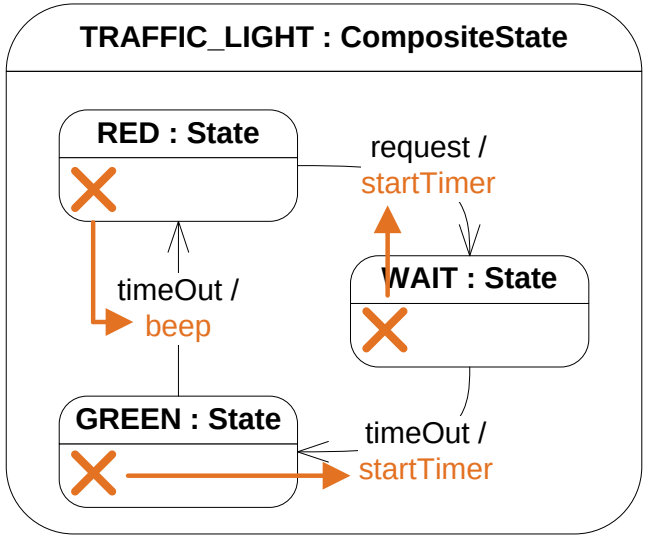


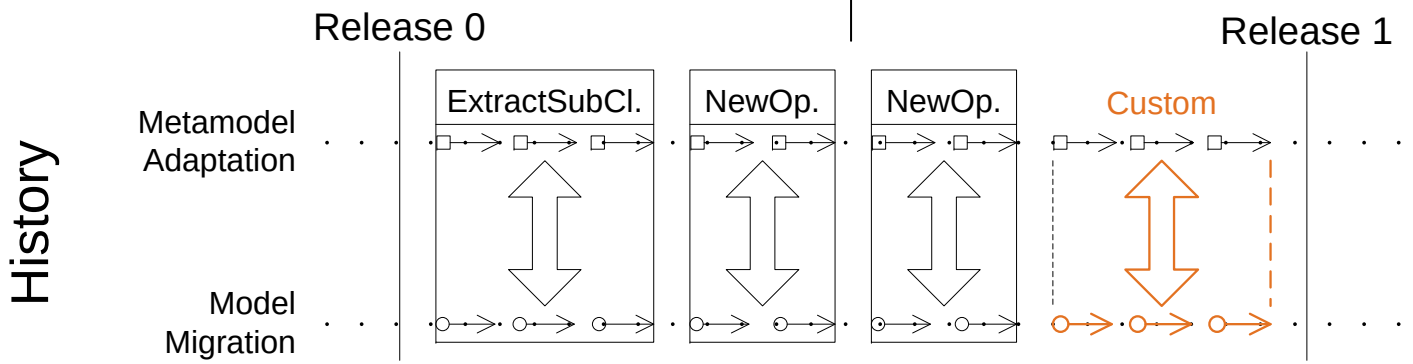
Solution – Moore to Mealy

Metamodel



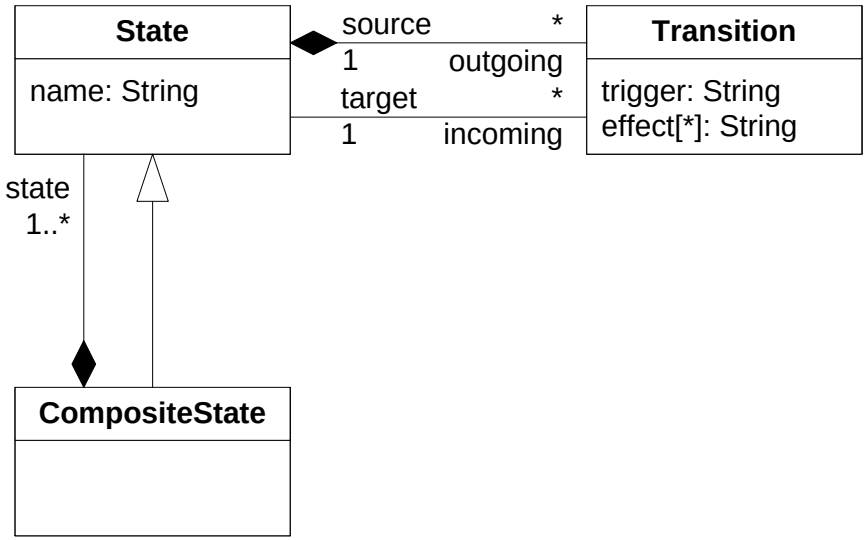
Model Migration



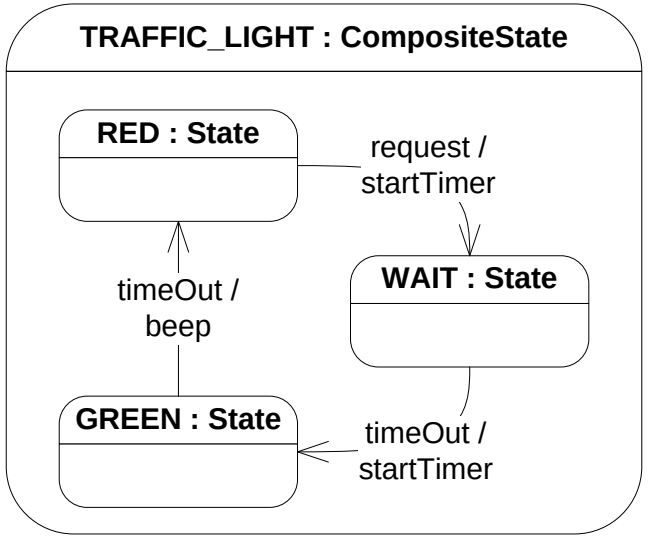


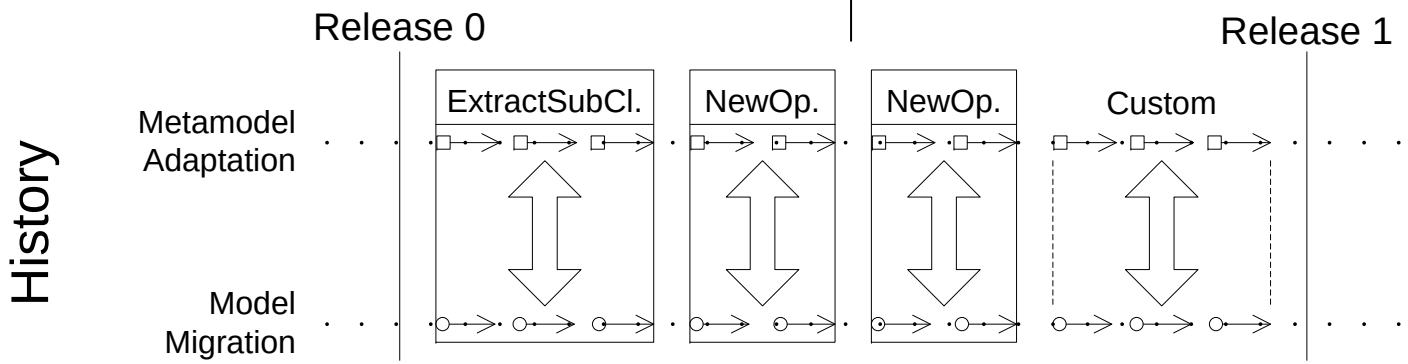
Solution – Moore to Mealy

Metamodel

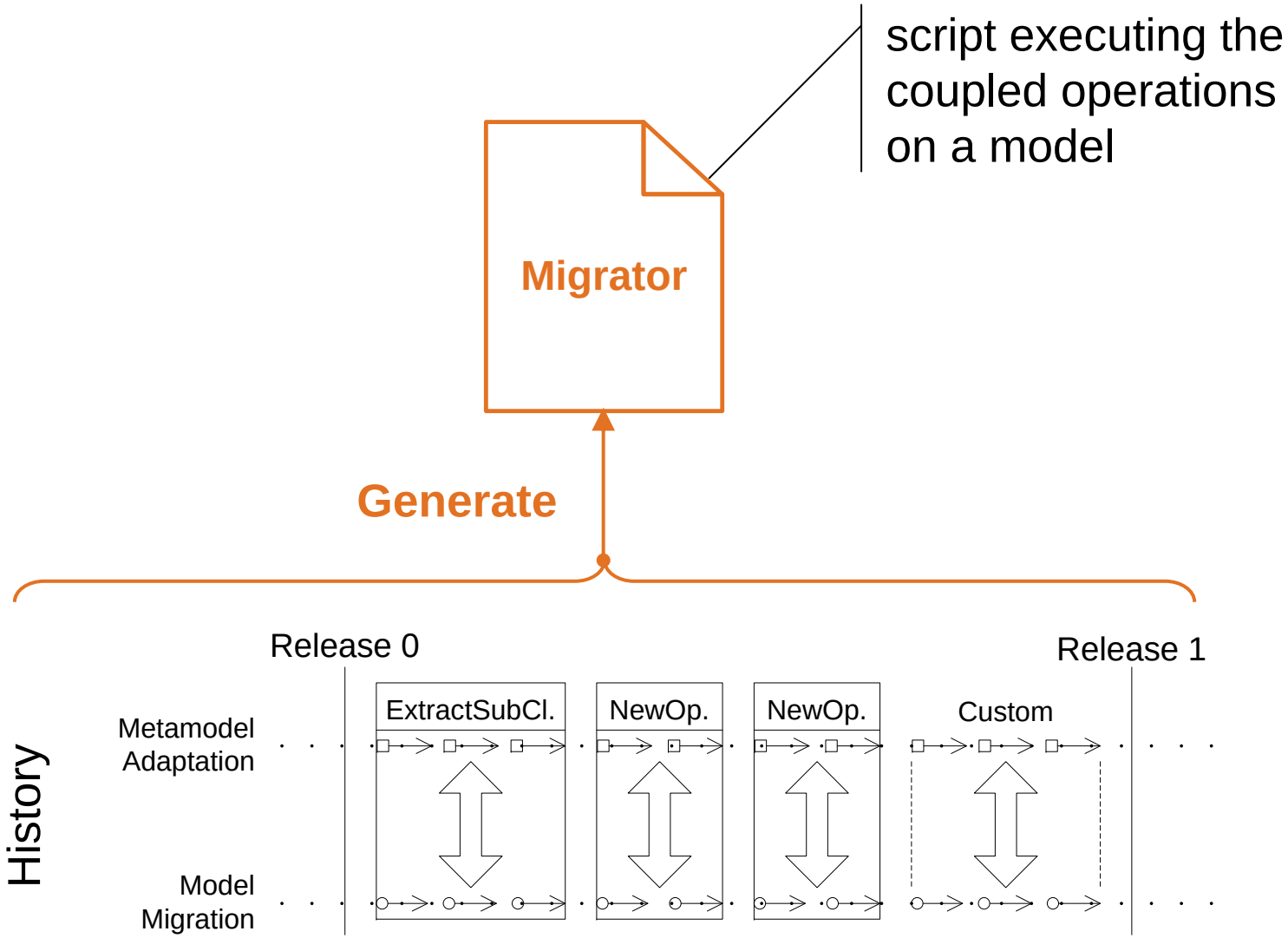


Model



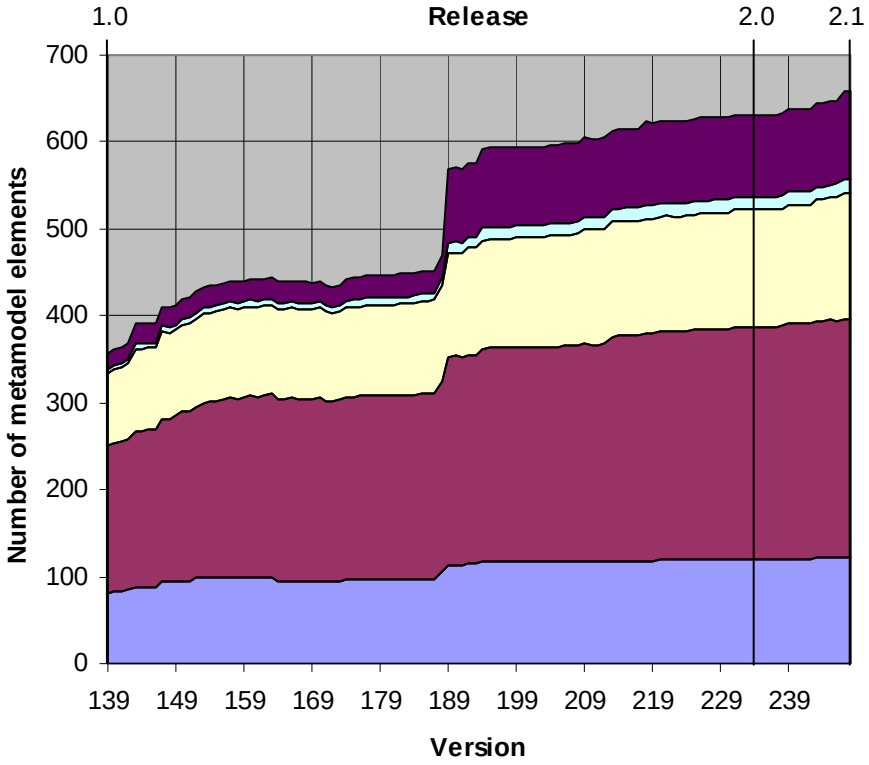


Solution – Model Migration

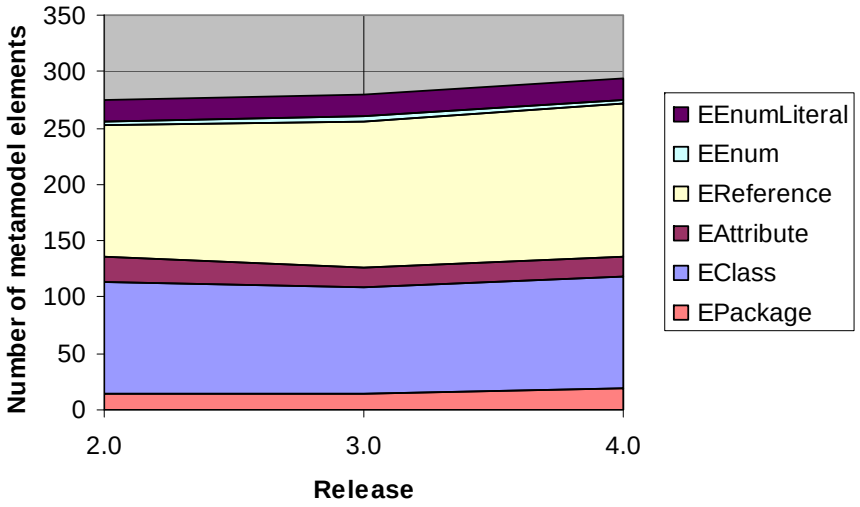


Case Study: Reverse Engineering of Metamodel History

GMF Generator Model
(Graphical Modeling Framework)

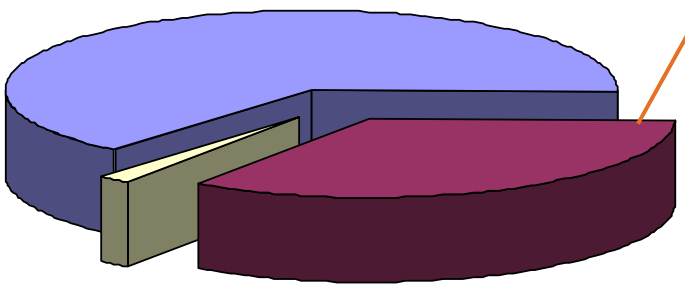


PCM
(Palladio Component Model)



Case Study: Results

GMF Generator Model
(Graphical Modeling Framework)



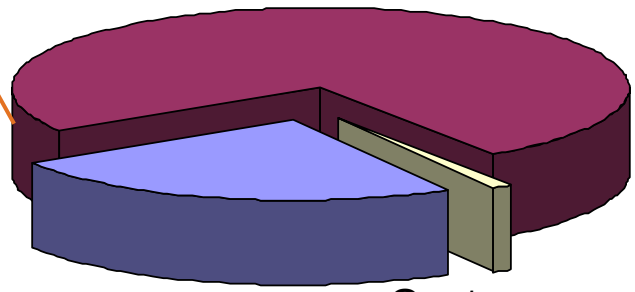
Metamodel Extension
148 (64%)

Custom Coupled Operation
4 (2%)

Reusable Coupled Operation
79 (34%)

High potential for reuse

PCM
(Palladio Component Model)



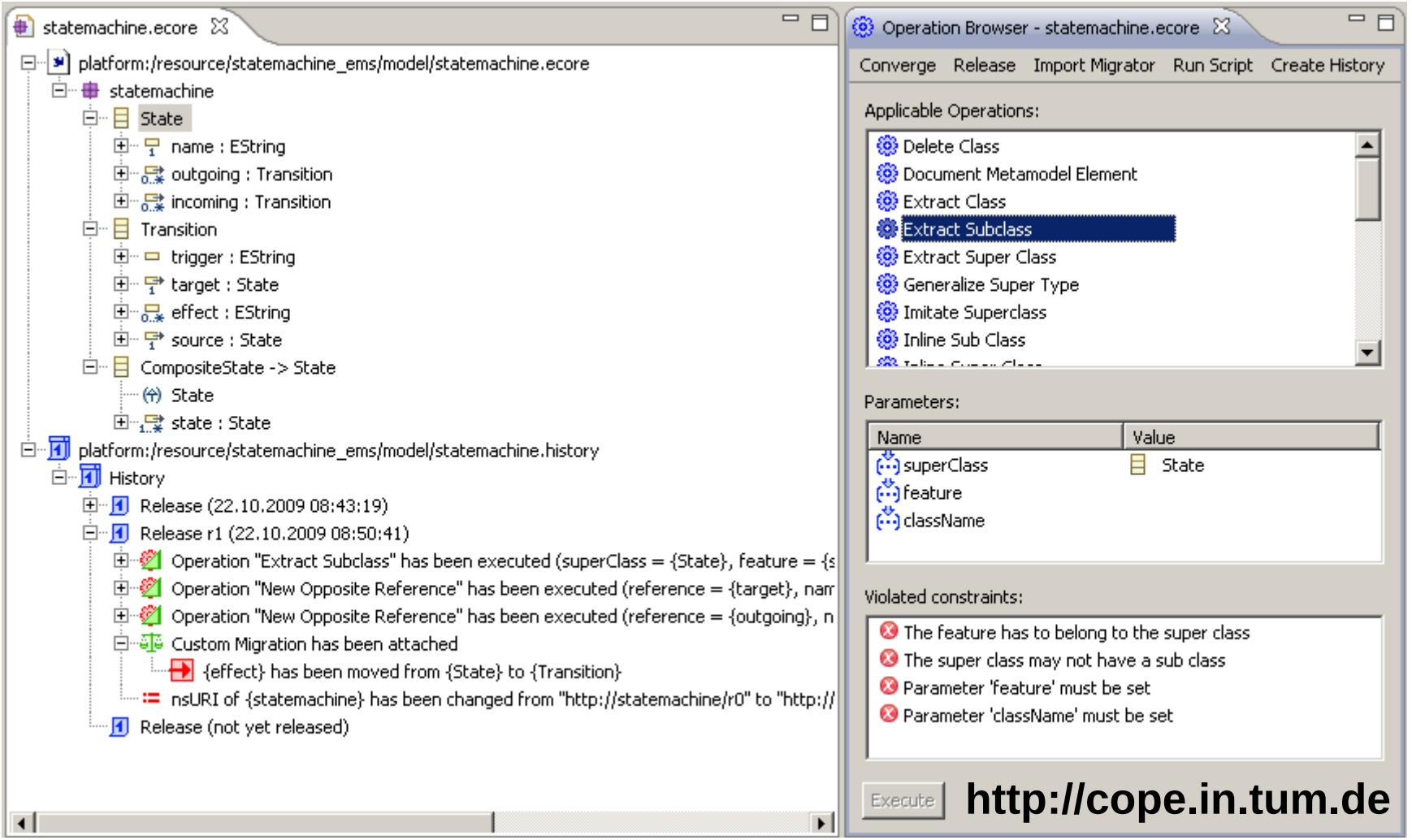
Reusable Coupled Operation
76 (74%)

Metamodel Extension
25 (25%)

Custom Coupled Operaction
1 (1%)

Expressiveness required for a few changes

Seamless Integration into EMF Metamodel Editor



Agenda

1. **Introduction**
   - principles behind COPE
2. **Guided Tool Usage**
   - simple example
3. **Independent Tool Usage**
   - more difficult task
   - evolve your own metamodel
4. **Wrap Up**
   - questions and feedback

Guided Tool Usage

- **Recording** the Coupled Evolution
  - Reusable Coupled Operations
  - Custom Coupled Operation
  - Migrator Generation
- **Inspecting** the Coupled Evolution
  - Reconstruction
  - History Differencing
- **Refactoring** the Coupled Evolution
  - Undoing Changes
  - Replacing Changes
  - Reordering Changes
- **Recovering** the Coupled Evolution
  - Metamodel Convergence

cope_features.pdf

Installation

Requirements

- minimum Java 1.5

Download from USB Stick

- Eclipse 3.5 SR2 Modeling Tools for your platform (Win, Mac, Linux)
- Folder for COPE containing update site and workspaces

Install on your machine

- Unzip Eclipse 3.5 SR2 Modeling Tools
- Add to eclipse.ini: "*-XX:MaxPermSize=256m*"
- Start executable "*eclipse*" in the extracted eclipse folder
- Choose the "*metamodelerWorkspace*" as workspace location
- Install COPE from the update site (Help -> Install new Software …)

Agenda

1. **Introduction**
   - principles behind COPE
2. **Guided Tool Usage**
   - simple example
3. **Independent Tool Usage**
   - more difficult task
   - evolve your own metamodel
4. **Wrap Up**
   - questions and feedback

Possible Tasks

Easy: Petrinet example

- no custom coupled operation

Medium: Filesystem example

- few reusable coupled operations
- 1 custom coupled operation

Difficult: UML Activities example

- many reusable coupled operations
- 1 custom coupled operation

Alternative: Evolve your own metamodel

What to do

1. Choose a task
2. Reconstruct the first release of the task using the history in the workspace
3. Create a history and apply the operations to converge to the second release
4. Test the migration on the provided model

Agenda

1. **Introduction**
   - principles behind COPE
2. **Guided Tool Usage**
   - simple example
3. **Independent Tool Usage**
   - more difficult task
   - evolve your own metamodel
4. **Wrap Up**
   - questions and feedback

Conclusion

Benefits of COPE

- Preserve the intended Model Migration
- Automate Model Migration as far as possible

Ongoing Work

- Make COPE available through EMFT project Edapt
- Find contributors for EMFT project Edapt

Future Work

- Specify Model Migration in Java
- Separate history for each Metamodel

http://cope.in.tum.de