Towards Refactoring of Rule-Based, In-Place Model Transformation Systems

Gabriele Taentzer, Thorsten Arendt

Philipps Universität Marburg

Claudia Ermel

Technische Universität Berlin
Germany

Reiko Heckel

University of Leicester

Towards Refactoring of Rule-Based, In-Place Model Transformation Systems

Gabriele Taentzer,
Thorsten Arendt

Philipps Universität Marburg

Claudia Ermel

Technische Universität Berlin
Germany

Reiko Heckel

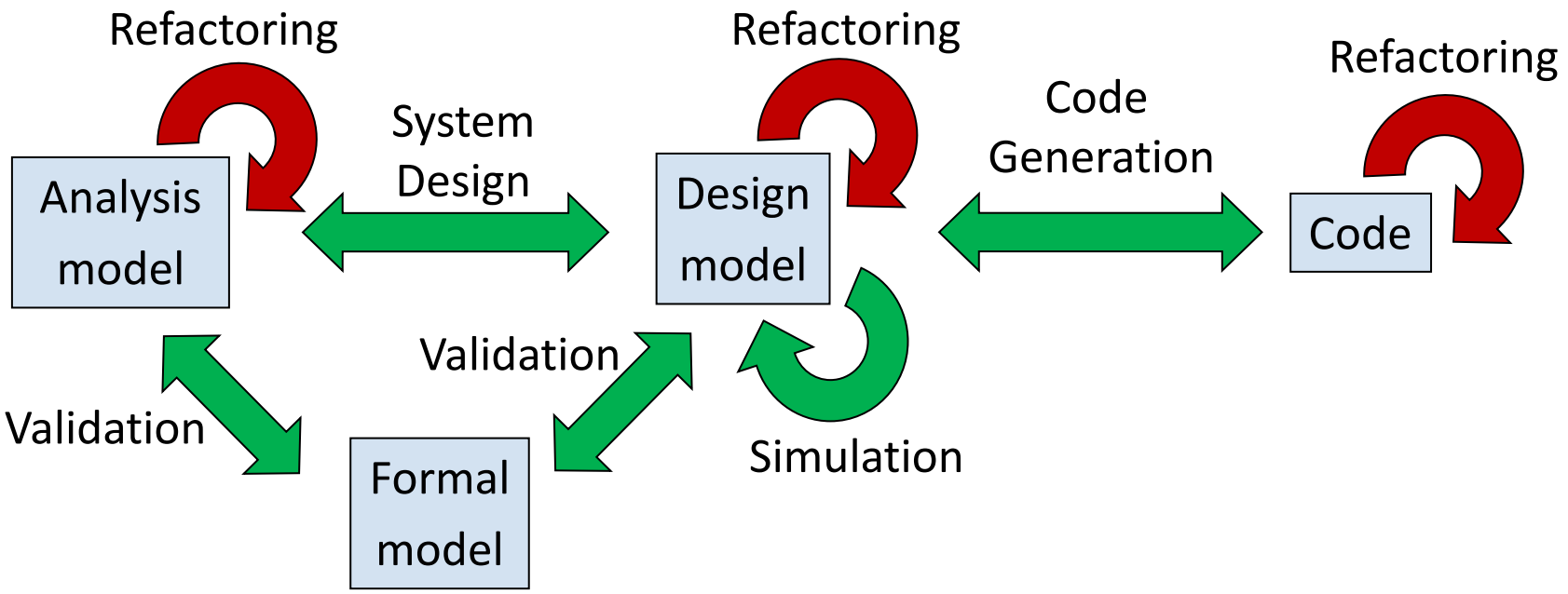
University of Leicester

Agenda

- **Motivation**
  - Improving the Quality of Model Transformations
- **Quality Aspects of Model Transformation Systems**
  - Selected Smells
- **Selected Refactorings**
  - Merge Rules Differing in Types Only
  - Extract Precondition
  - Unify Rules with Same Actions
- **Some Remarks on Semantics Preservation**
- **Implementation: Using the EMF Model Transformation Tool Henshin**
- **Conclusion and Future Work**

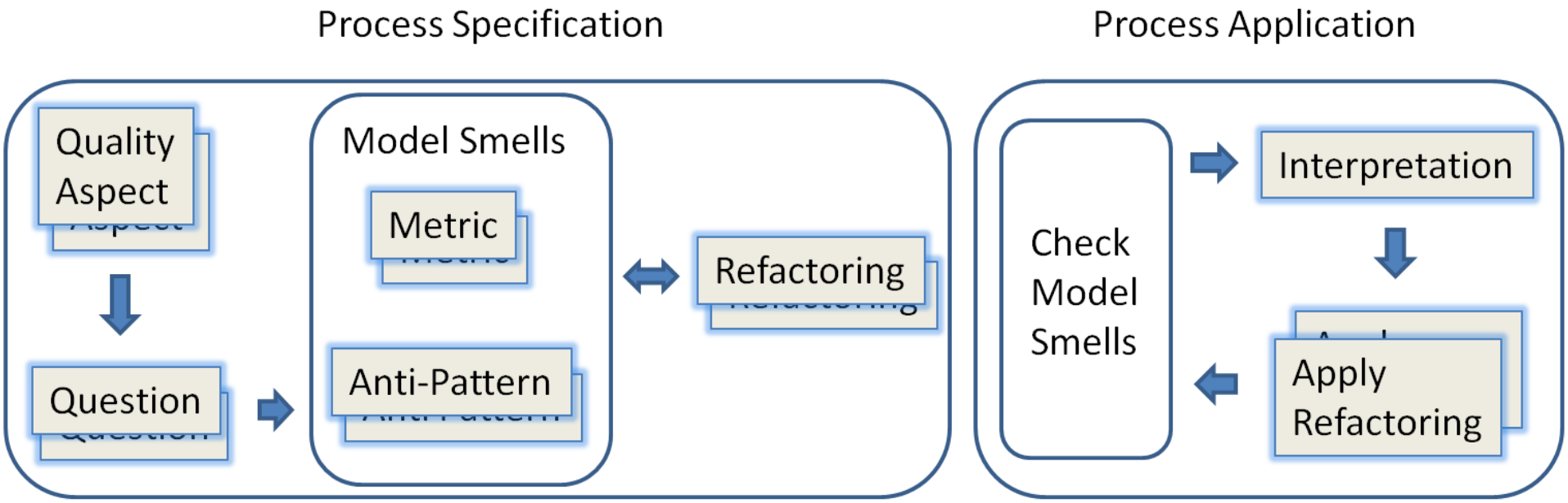
Motivation: Model Transformations in MDD

Model Transformations and Refactorings: Central Technologies in MDD



- Challenge: Evolution, maintainability and quality assurance of model transformations
- Solution Idea:
  - Consider rule-based, in-place model transformation systems
  - Use Rule-Based Transformation for Refactoring of Model Trafos

A Quality Assurance Process for Models



Aim: Apply the quality assurance process to rule-based model transformation specifications

Quality Aspects of Model Trafo Systems

Correctness

- Concerning syntax and semantics

Conciseness

- Presentation at the “right“ level of abstraction

Changeability

- Rapid and continuous evolution possible

Comprehensibility

- Understandable by intended users

Selected Smells of Model Trafo Systems (1)

Large Rule

- Detection: number of elements (objects, relations, pre-conditions, actions, …)
- Affected quality aspects: conciseness, comprehensibility

Redundant Attributes and References

- Detection: No. of equal attributes and references
- Affected quality aspect: conciseness, comprehensibility, changeability

Redundant Rules

- Detection: No. of rule pairs differing in types only
- Affected quality aspect: conciseness, comprehensibility, changeability

Selected Smells of Model Trafo Systems (2)

Unused Object Type

- Detection: number of rules using a specific object type
- Affected quality aspect: correctness, conciseness, completeness

Delete and Created the Same Object

- Detection: find corresponding patterns in rules
- Affected quality aspect: conciseness, comprehensibility

Rules with Common Subrules

- Detection: find corresponding subpatterns in rules
- Affected quality aspect: conciseness, changeability, comprehensibility

Refactoring of Model Trafo Systems

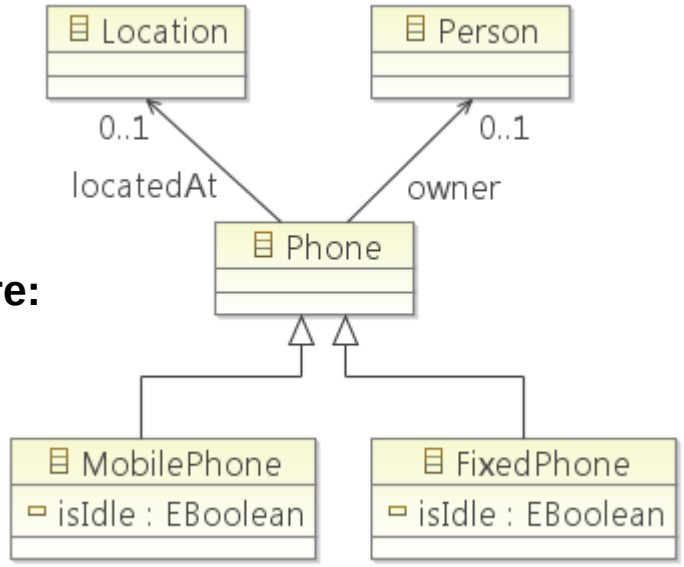
Refactoring changes a model transformation system such that its structure is improved while its semantics is preserved.

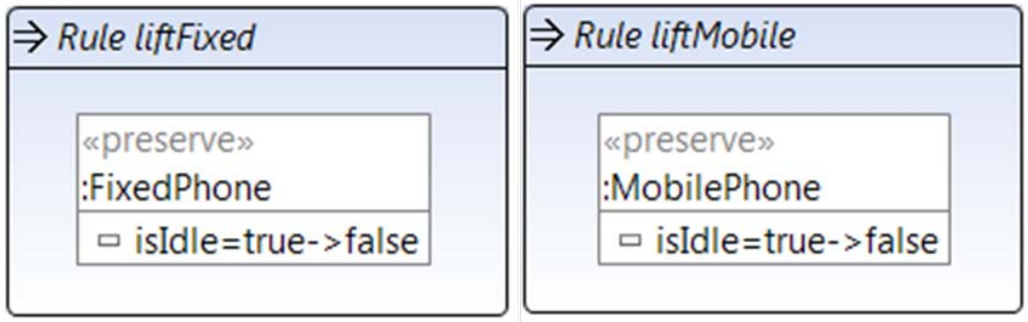
Selected Refactorings, improving

- *typing:* pull up attribute, push down attribute, extract supertype, remove supertype, extract node type, loop-edges-to Boolean-attributes
- *rules:* extract pre-condition, split rule into set of action rules, merge rules differing in types only, move versus delete and create, unify rules with same actions
- feature usage: inline pre-condition, Boolean-attribute-to-loop-edges,

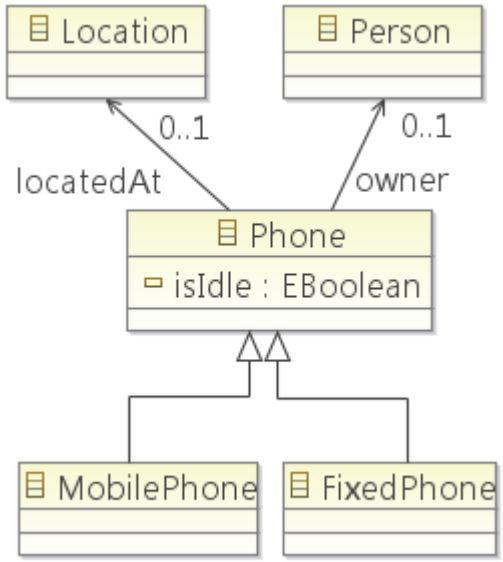
Refactoring „*Merge Rules Differing in Types Only*“

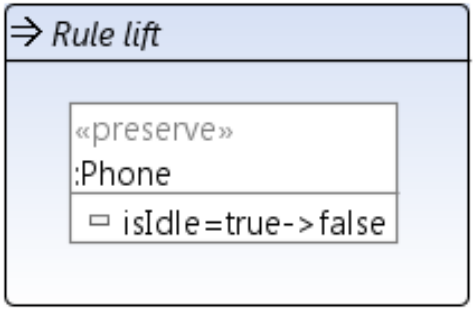
Before:

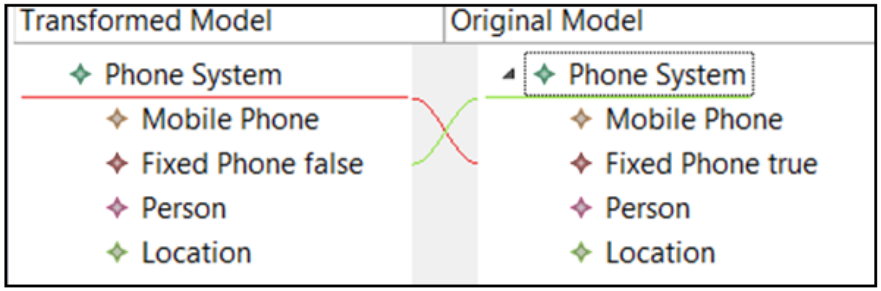




After:







Refactoring *„Merge Rules Differing in Types Only“*

Input parameters: Names of rules to be merged

Pre-condition: Rules differ in one object type only; the set of found object types contains all subclasses of a common superclass

Strategy:

- Identify all varying object types with a common superclass
- Construct new rule by taking one existing rule and replacing the identified subclass by the identified superclass; rename rule
- Delete all remaining original rules

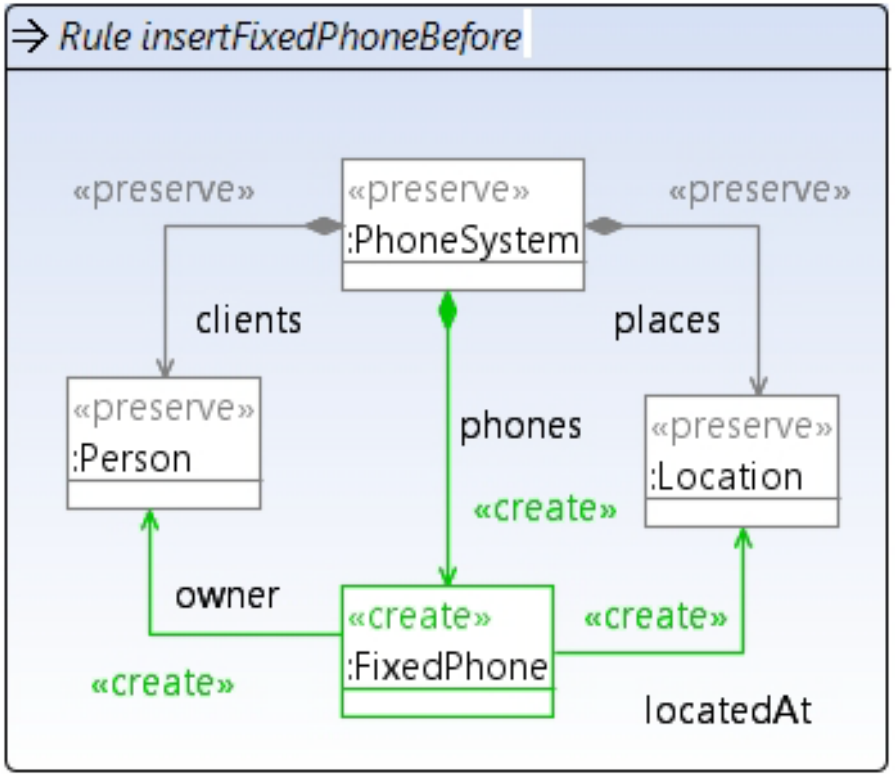
Post-condition:

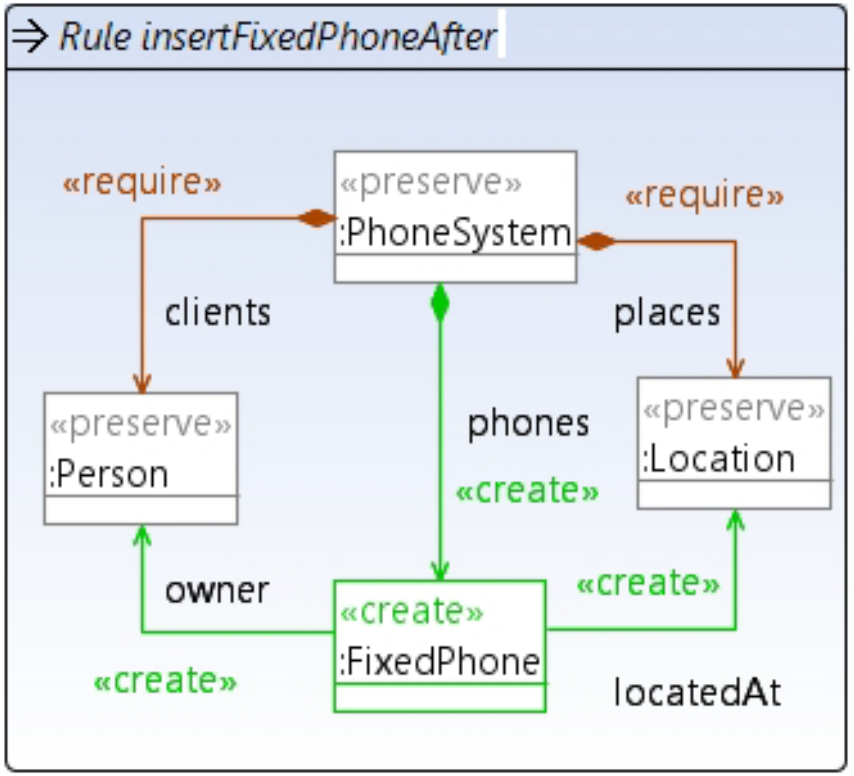
- Original rules are replaced by one new rule using superclass type

Quality improvement: conciseness, comprehensibility

Semantics: is preserved

Refactoring „*Extract Pre-condition*“





Refactoring „*Extract Pre-condition*“

Input parameters: Rule name

Pre-condition: none

Strategy:

- Determine preserved part
- Create new PAC and put preserved part in
- Reduce the rule‘s preserved part

Post-condition:

- Preserved part is minimal

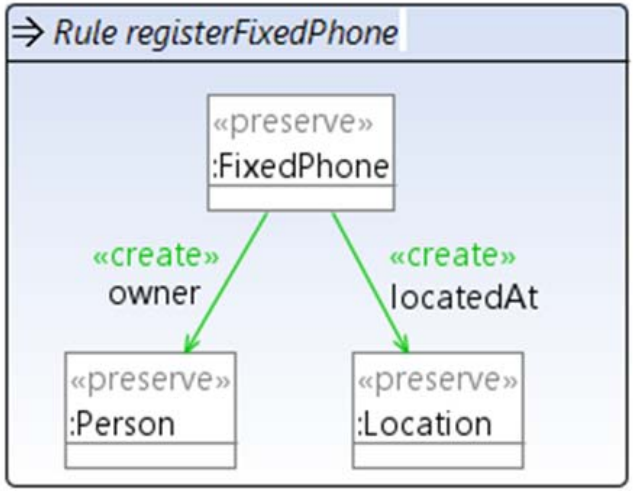
Quality improvement:

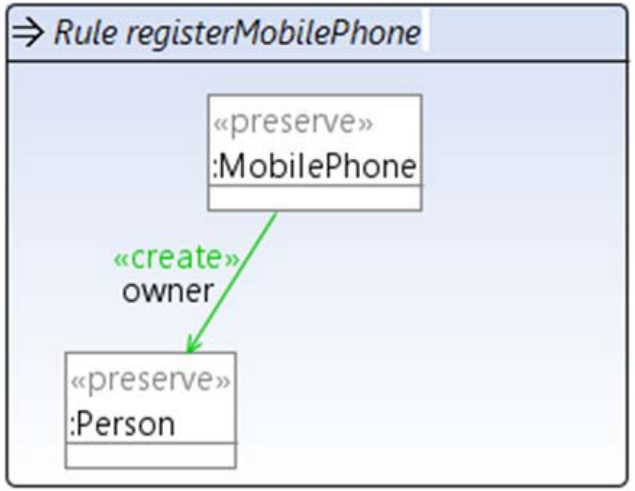
- conciseness, comprehensibility

Semantics: is preserved

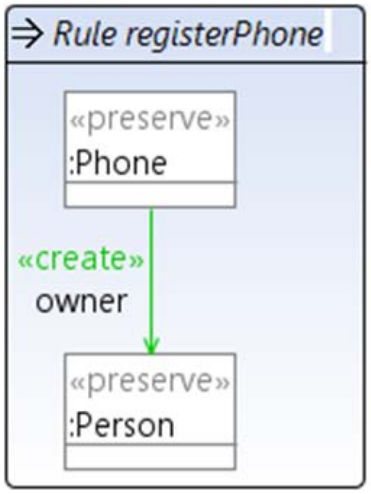
Refactoring „*Unify Rules with Same Actions*“

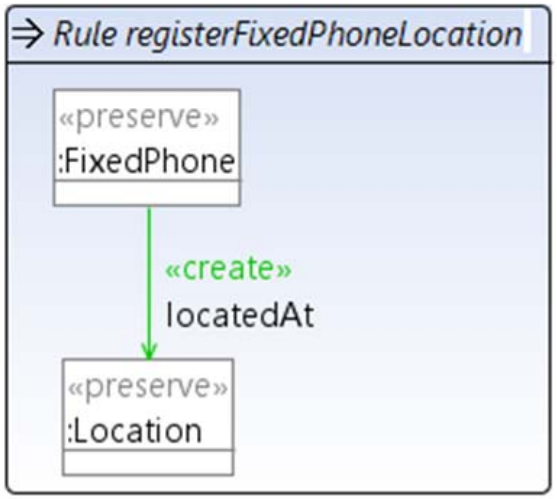
Before:





After:





Refactoring „ *Unify Rules with Same Actions*“

Input parameters: Set of rule names

Pre-condition: none

Strategy:

- Identify set of actions being shared by the set of input rules
- Create new rule scheme with common action (kernel rule) and remaining actions (remainder rules)
- Apply kernel rule before remainder rules

Post-condition:

- new kernel rule (common actions) and remainder rules

Quality improvement:

- conciseness

Semantics: more trafo rule sequences (more interleaving of rules)

Some Remarks on Preservation of Semantics

Refactoring $r : MTS_1 \rightarrow MTS_2$ induces mapping of models (graphs).

Depending on the kind of semantics, e.g.

- generated language $L(MTS)$,
- transformation relation $MT(MTS)$ over input-output models,
- operational semantics $LTS(MTS)$,

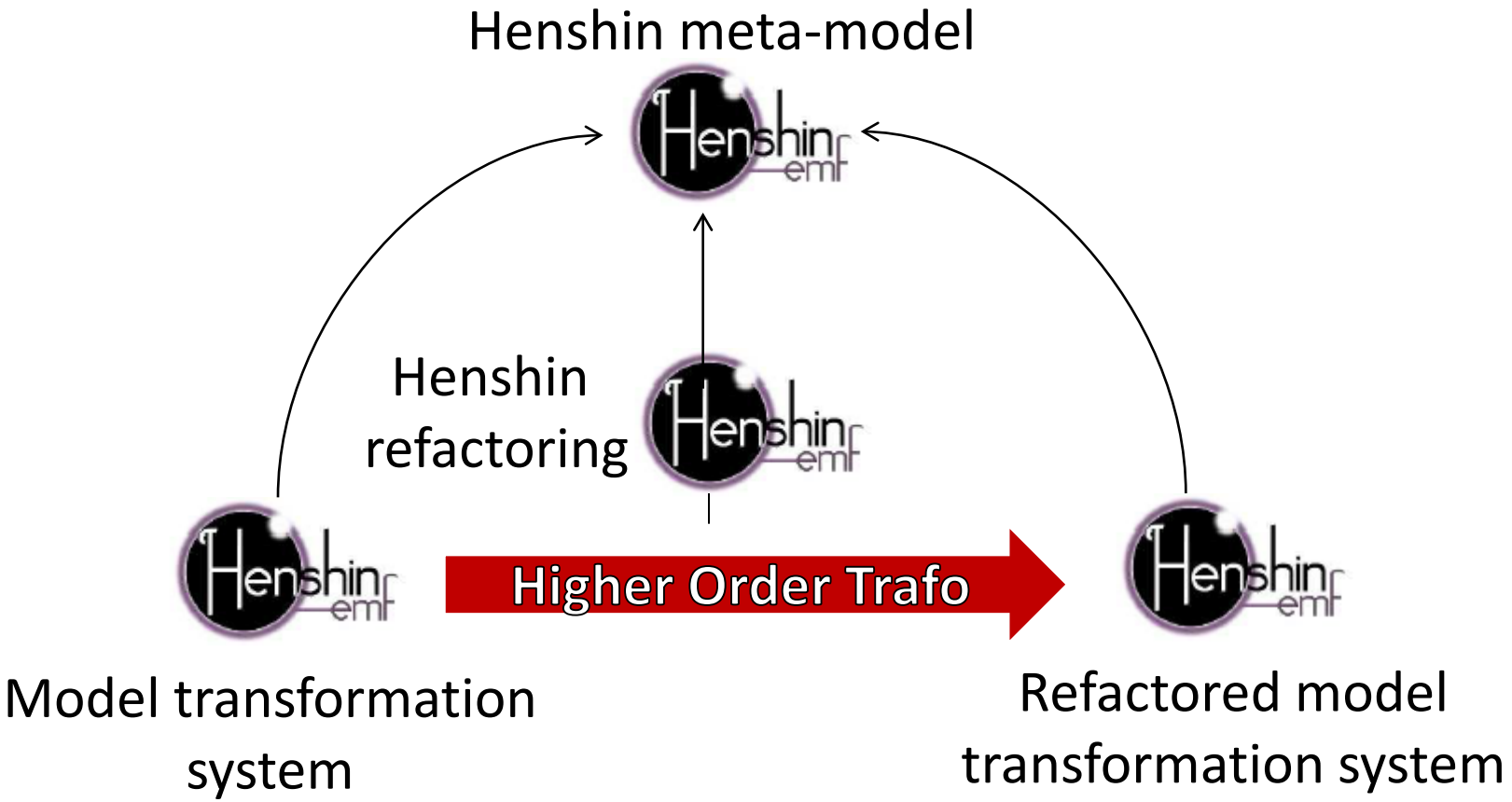
preserving them means different things.

For example, *r* preserves the

- generated language if $r(L(MTS_1)) = L(MTS_2)$
- transformation relation if $r\,(MT(MTS_1)) = MT(MTS_2)$
- operational semantics if r is suitable observational equivalence relating $LTS(MTS_1)$ and $LTS(MTS_2)$

Prototypical Implementation

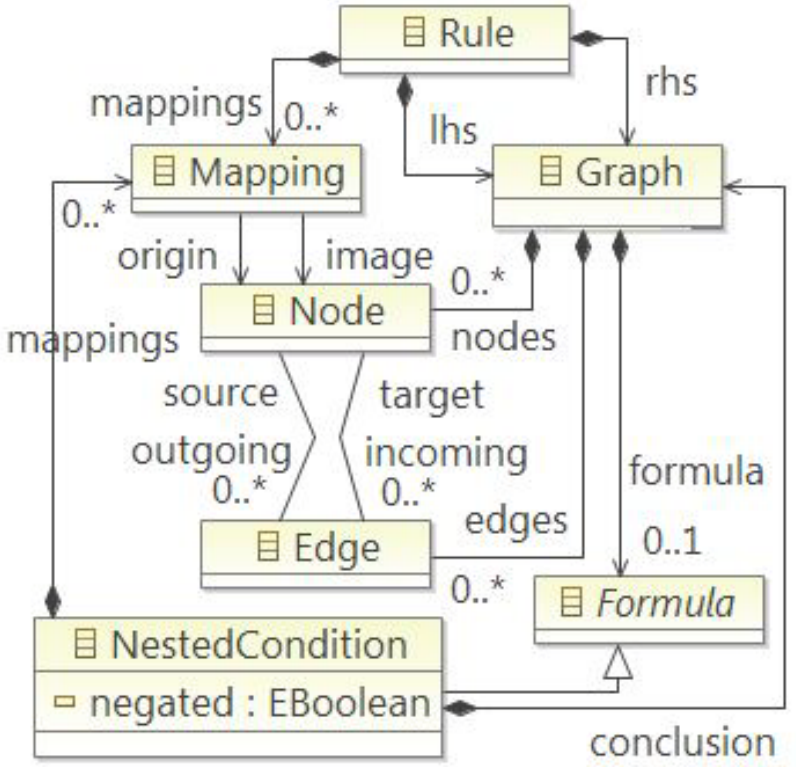
Model transformation language based on graph transformation concepts for the Eclipse Modeling Framework



Higher Order Transformations in Henshin

Henshin Features

- *in-place:* models are transformed directly
- *rule-based:* transformation rules are defined visually
- *transformation model in EMF*:

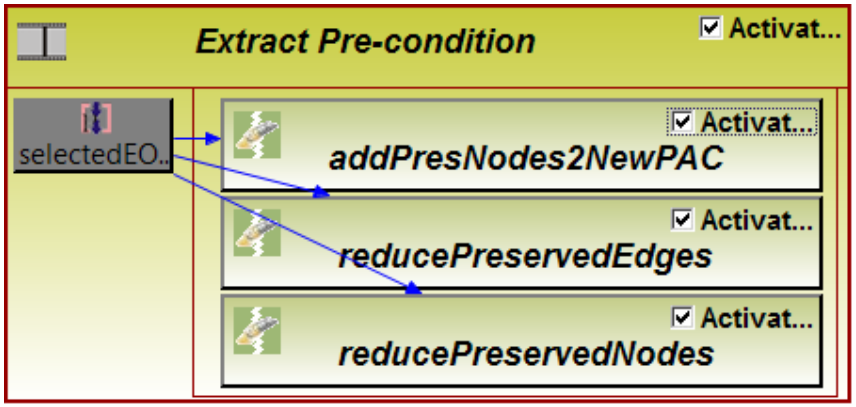


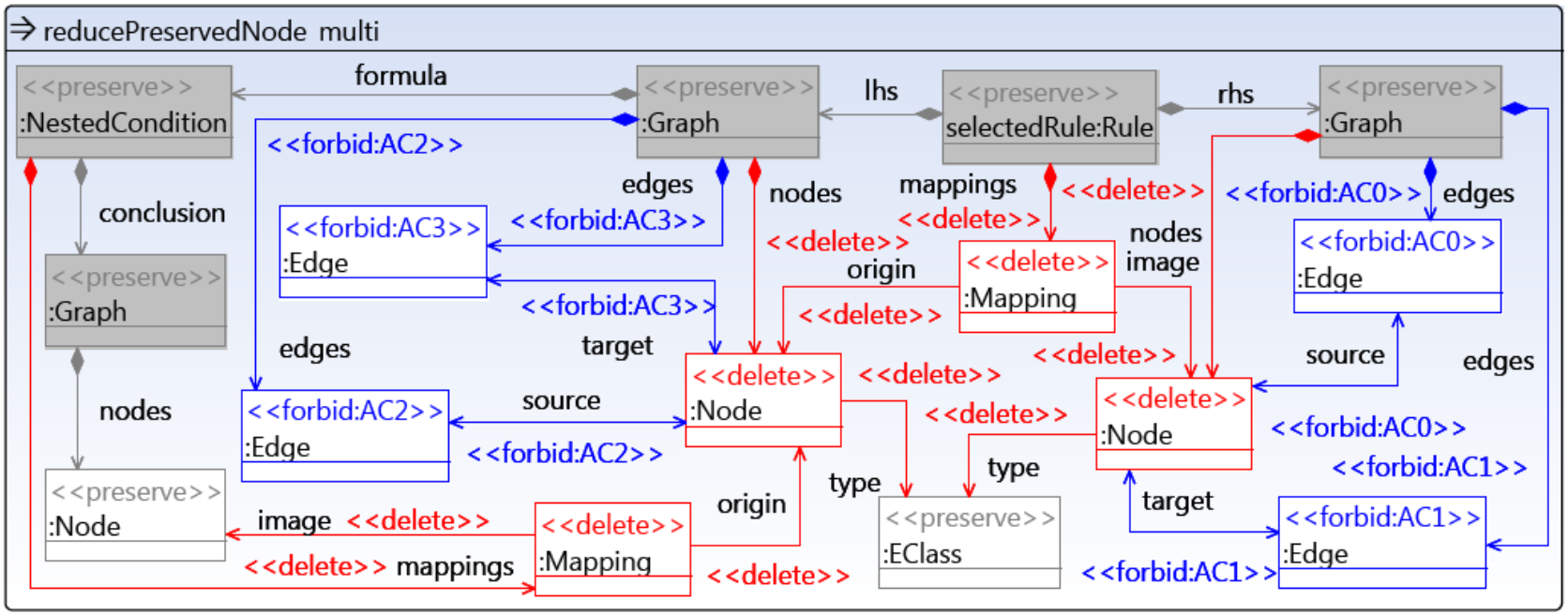
- refactorings: higher-order transformation on the Henshin model

Example: Extract Pre-condition

Control Structures on Rules

(here: rule sequence)





Integration of refactorings in Henshin editors by EMF Refactor

Conclusion

New topic: Refactoring of model transformation systems

Modeling experiences are made explicit.

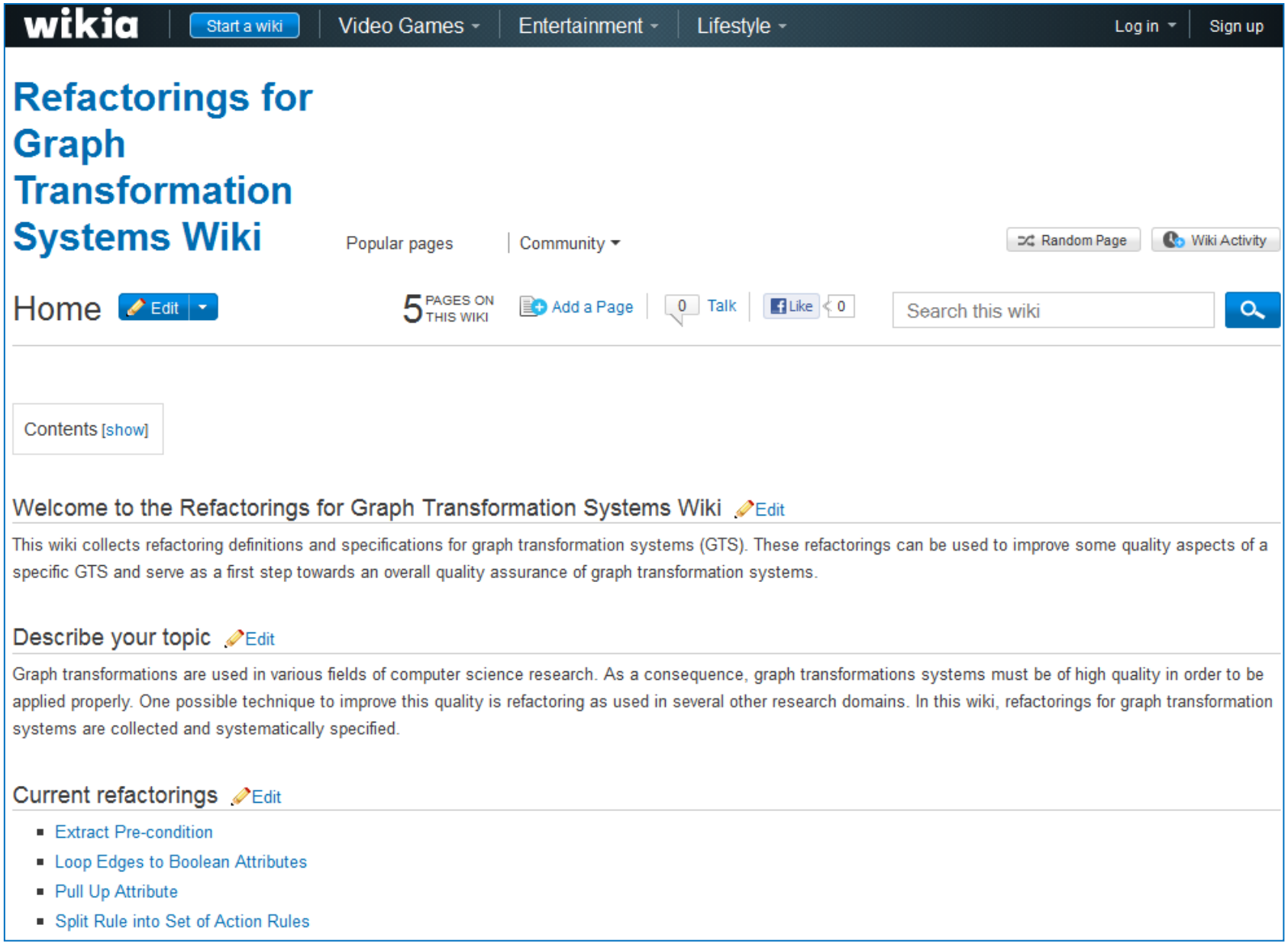
Selection of interesting refactorings as starting point

What next?

Startet joint activity to build up a Wiki with interesting smells and refactorings (for graph transformations so far)

A glimpse at the graph transformation refactoring wiki

http://refactorings4gts.wikia.com/



A glimpse at the graph transformation refactoring wiki

http://refactorings4gts.wikia.com/

wikia | Start a wiki | Video Games | Entertainment | Lifestyle | Log in | Sign up

Refactorings for Graph Transformation Systems Wiki

Popular pages | Community | Random Page | Wiki Activity

Extract Pre-condition

Edit | 0 Comments | Like 0

Contents [show]

Description Edit

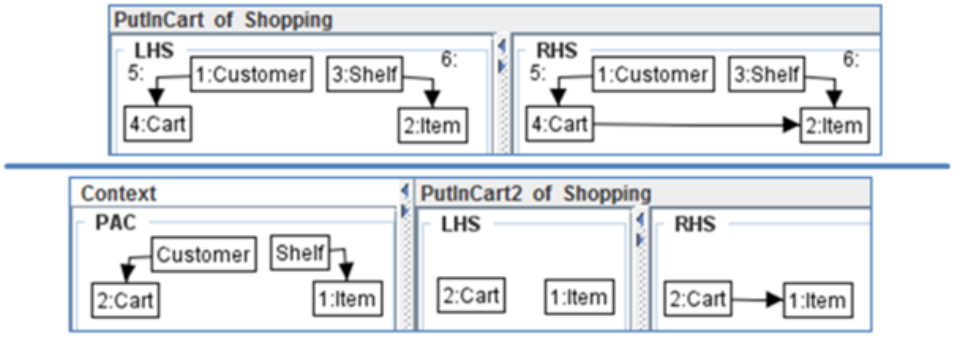
This refactoring reduces the preserved part of a rule and extracts it as positive application condition.

Input parameters Edit

- name of the rule (respectively the rule itself)

Example Edit

A customer takes an item out of the shelf. The rule mainly consists of context which has to be determined. We extract this context into a positive application condition which makes the rule considerably smaller (see figure).



Search this wiki

5 PAGES ON THIS WIKI | Add a Page

Advertisement | Your ad here

Latest Photos

7 PHOTOS ON THIS WIKI | Add a Photo

See all photos >

Recent Wiki Activity

See more >

Conclusion

New topic: Refactoring of model transformation systems

Modeling experiences are made explicit

Selection of interesting refactorings as starting point

What next?

Started joint activity to build up a Wiki with interesting smells and refactorings (for graph transformations so far)

Cover further model transformation features: join forces with Wimmer et al. (including also constraints, e.g. OCL)

THANK YOU FOR YOUR ATTENTION

M. Wimmer, S. Martinez, F. Jouault, J. Cabot: **A Catalogue of Refactorings for Model-to-Model Transformations**. Journal of Object Technology, August 2012