Adding Rule-Based Model Transformation to Modelling Languages in MetaEdit+

Simon Van Mierlo
Hans Vangheluwe

Universiteit Antwerpen

If you can Model, you can Transform

To demonstrate that transformation languages can be modelled explicitly

- their concrete and abstract syntax
- their execution semantics (building on a transformation kernel)

… independent of implementation technology

- AToM³ and AToMPM
- **MetaEdit+**

Model transformations are useful.

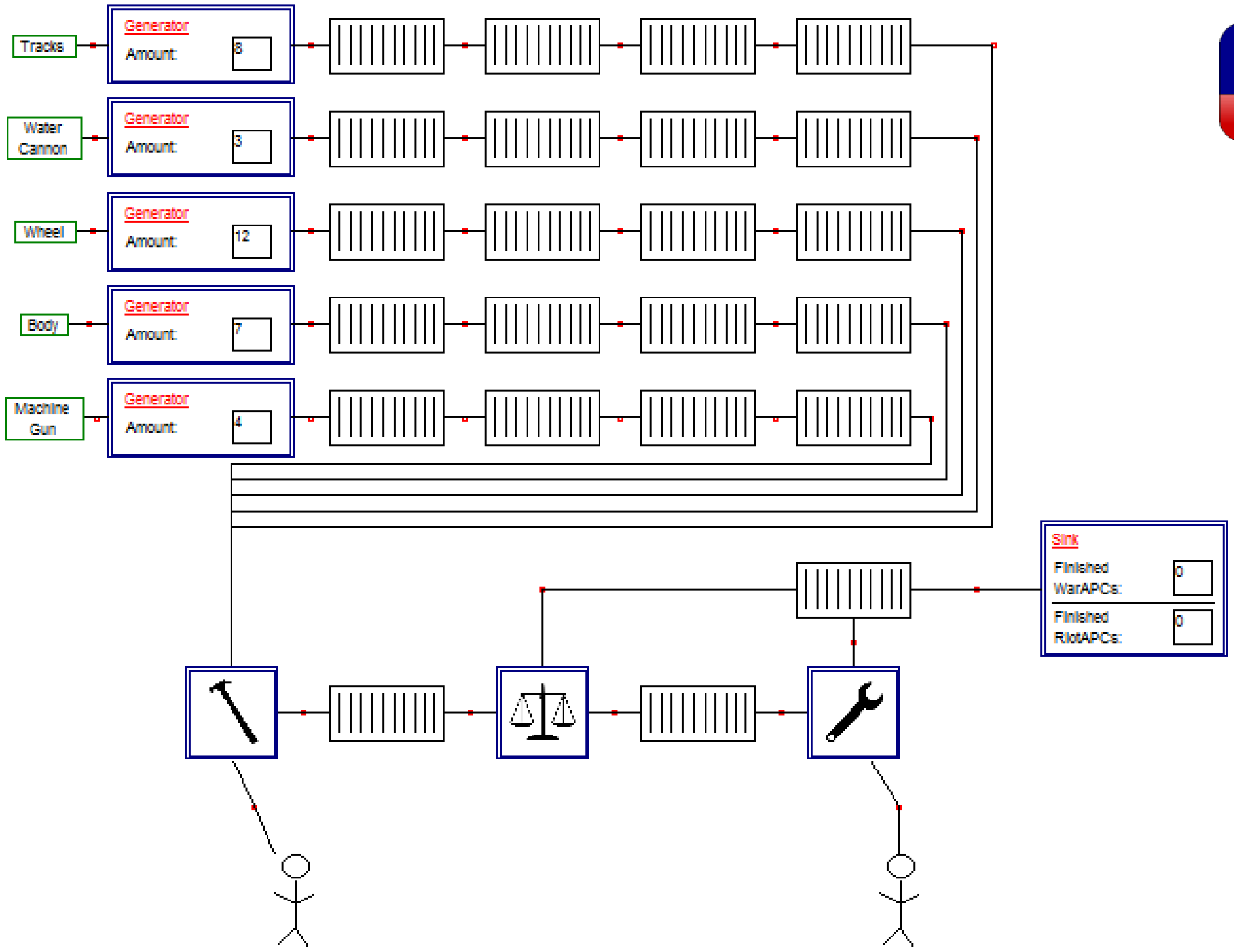
We focus on modelling operational semantics of a DSL using rule-based model transformations.

This is done in the commercial tool MetaEdit+.

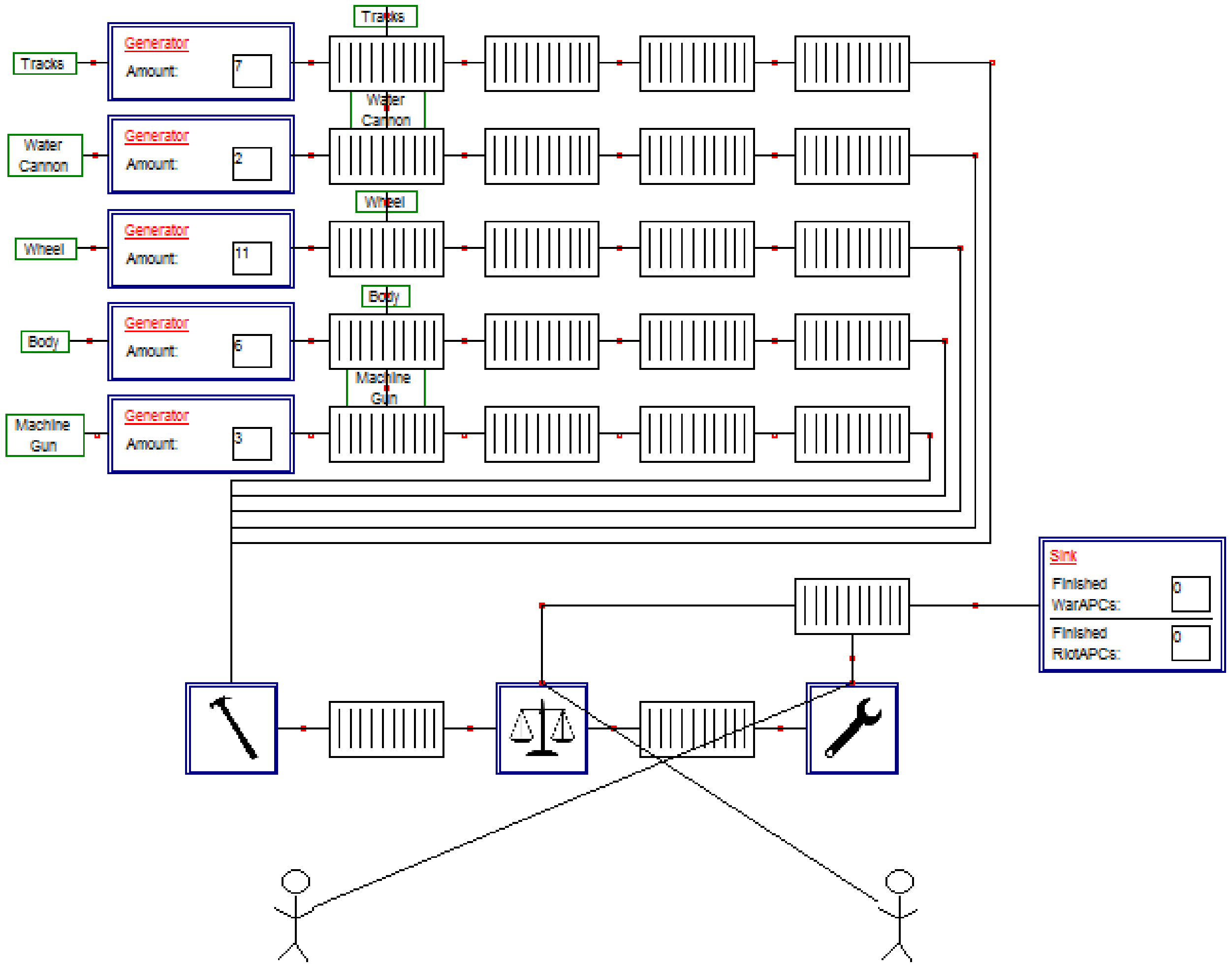
This tool has no model transformation support.

Contents

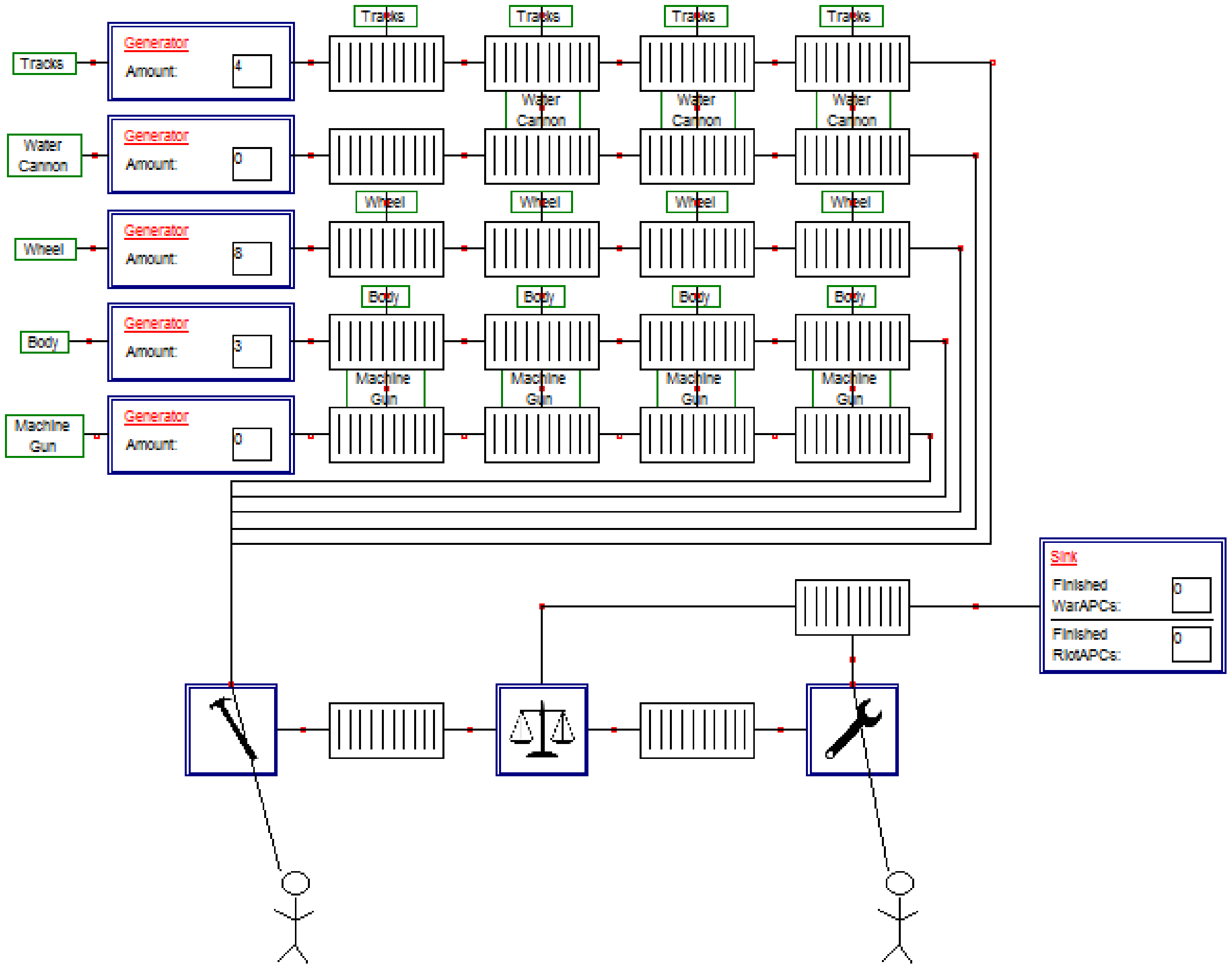
Example Case: Production System Language



Example Case: Production System Language

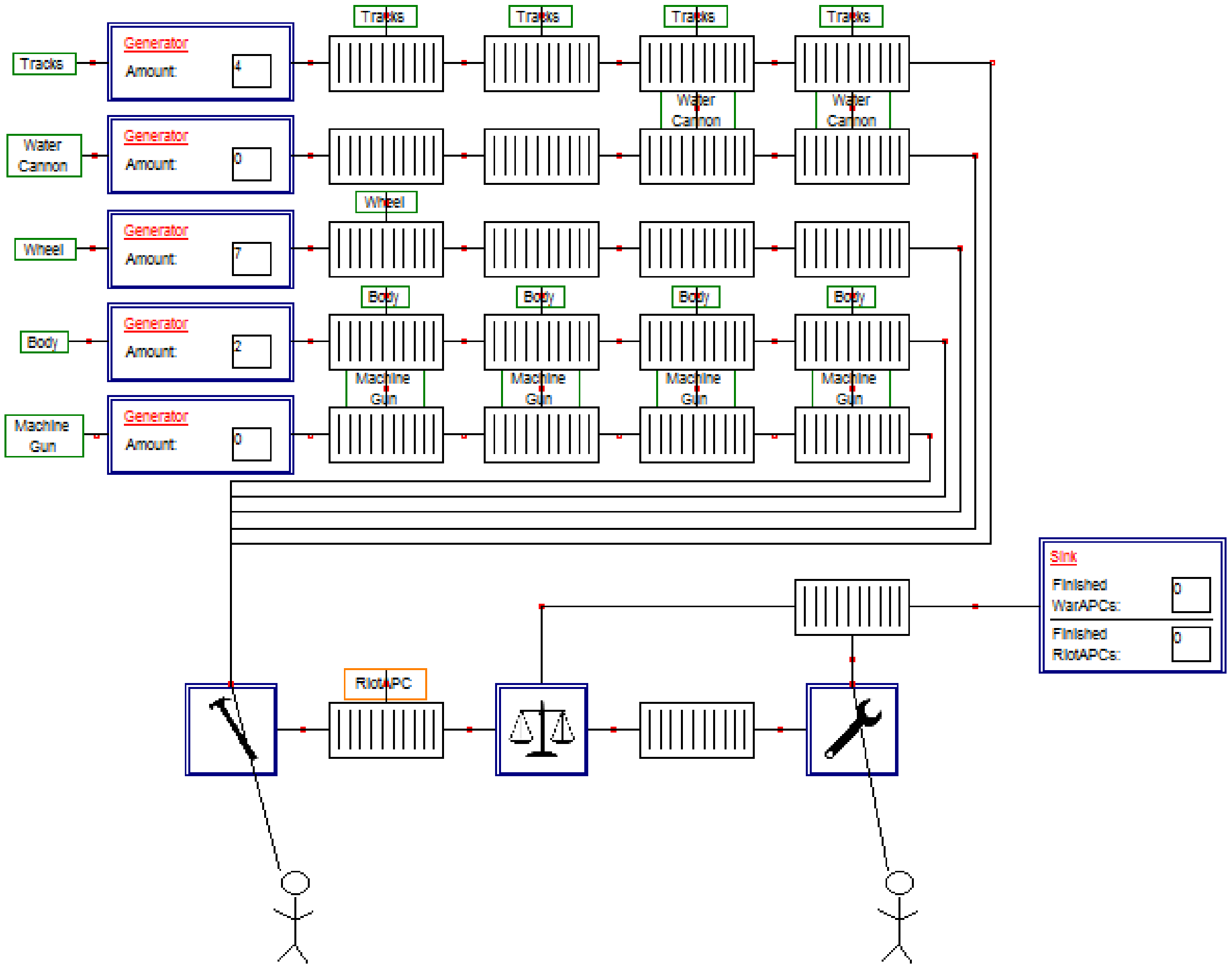


Example Case: Production System Language

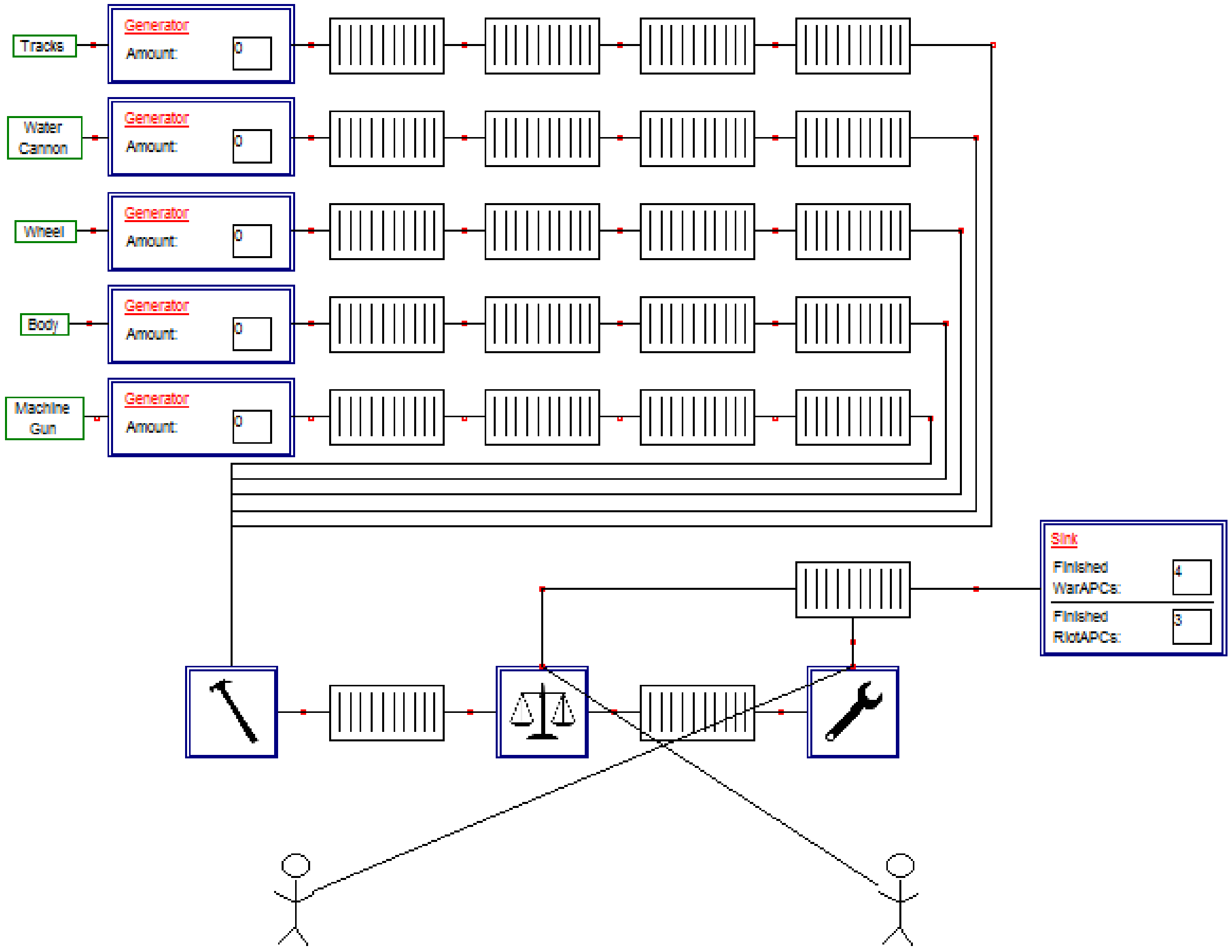


Universiteit Antwerpen

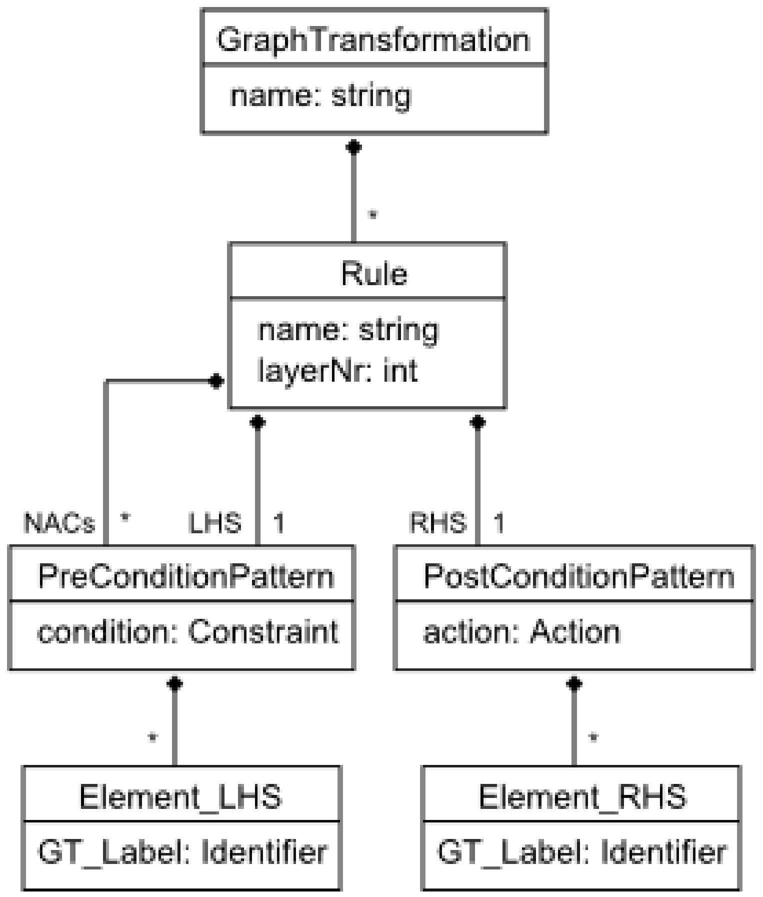
Example Case: Production System Language



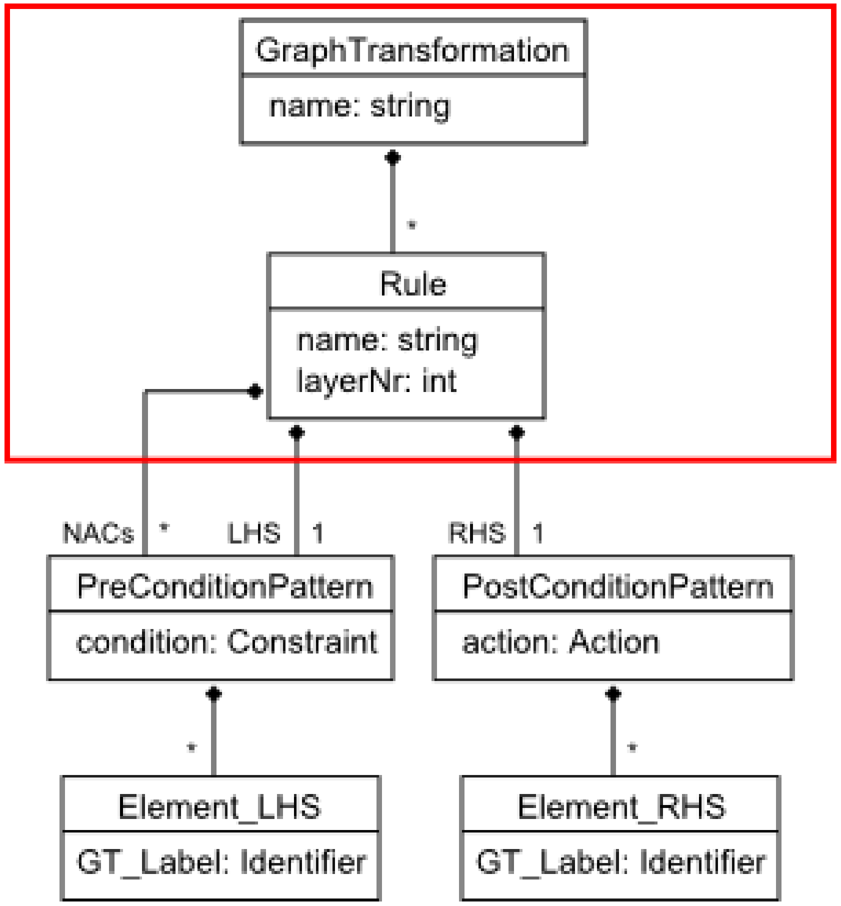
Example Case: Production System Language

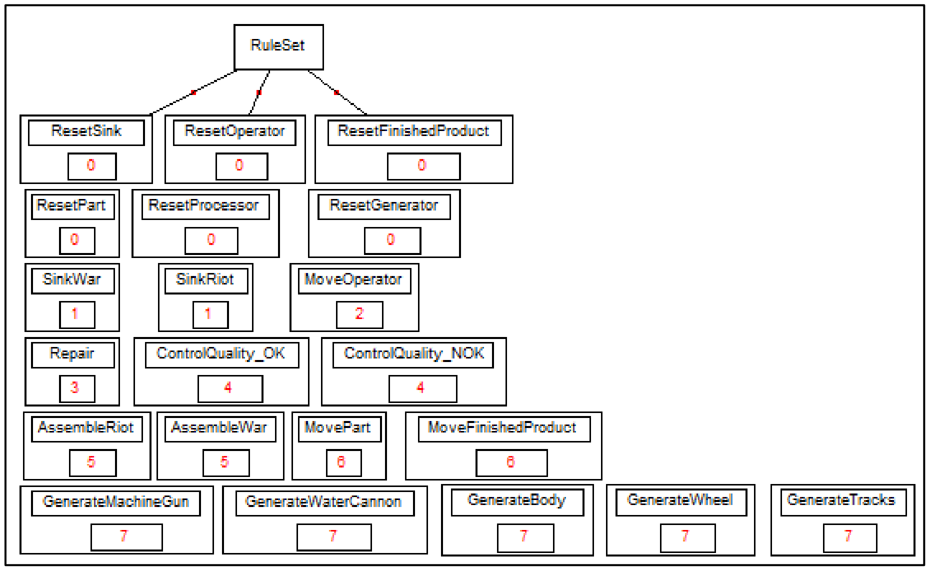


Explicitly Modelling the Transformation Language

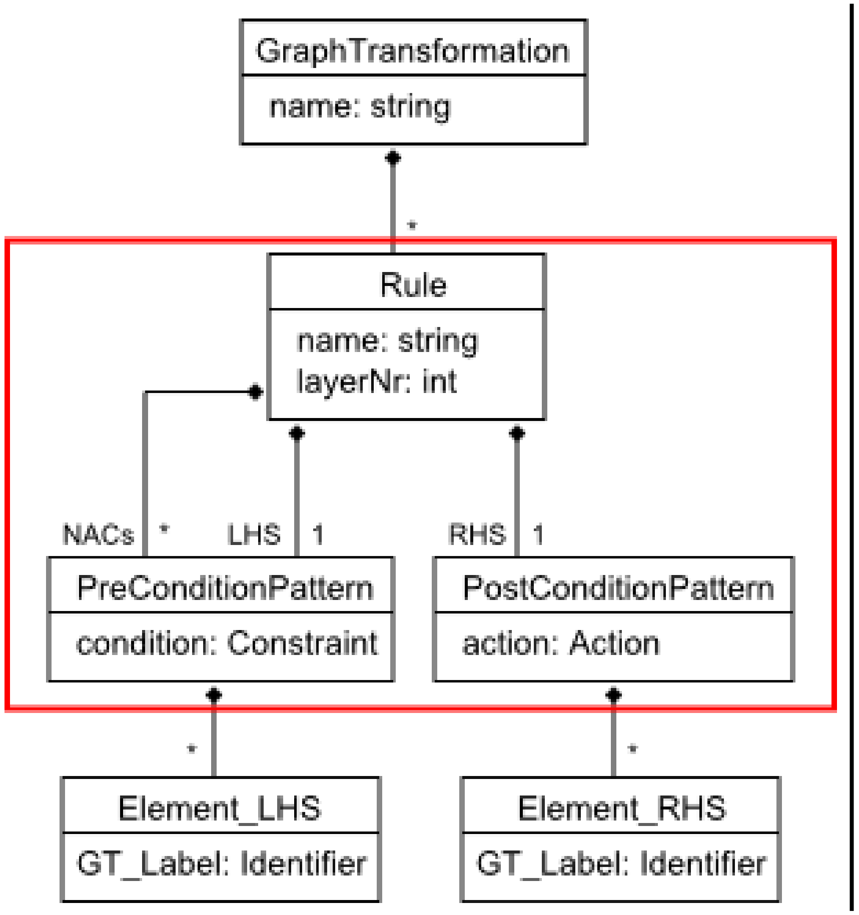


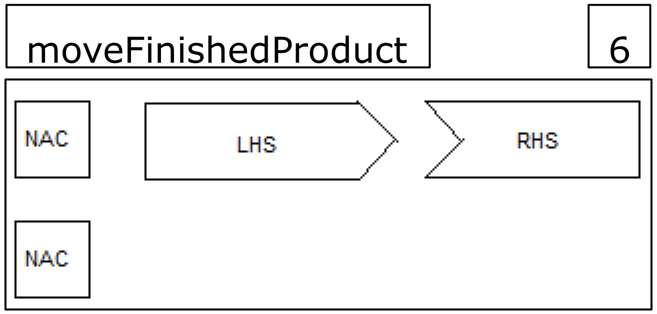
Explicitly Modelling the Transformation Language



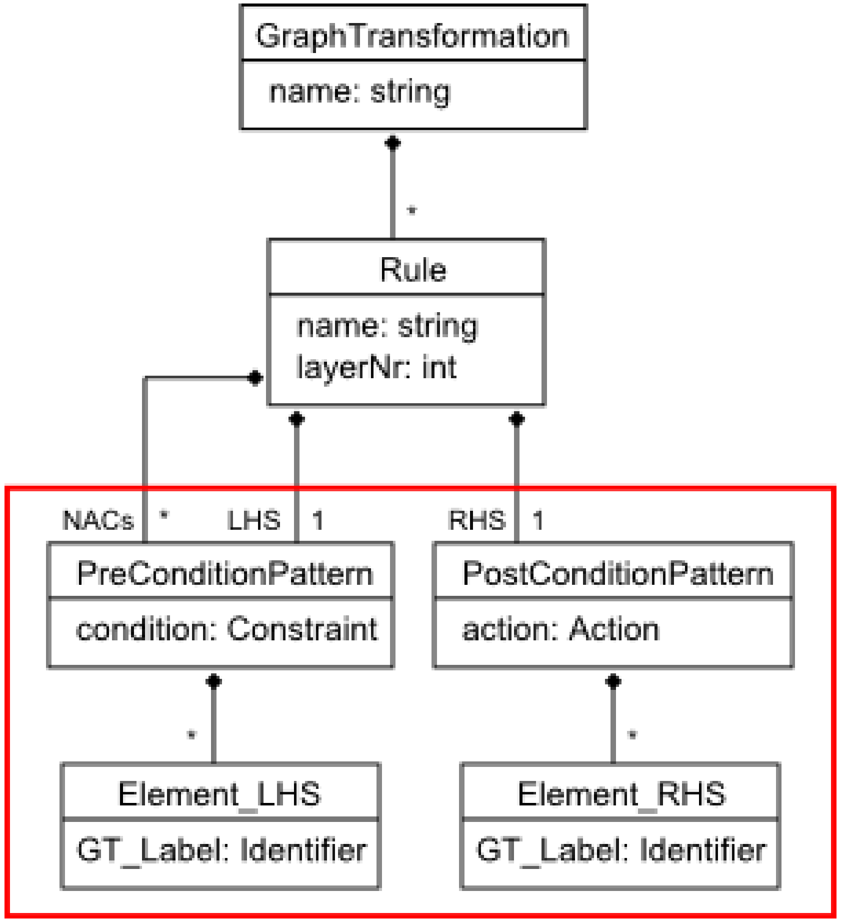


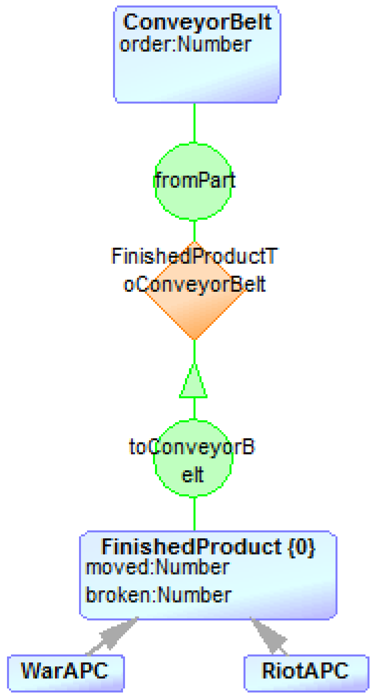
Explicitly Modelling the Transformation Language





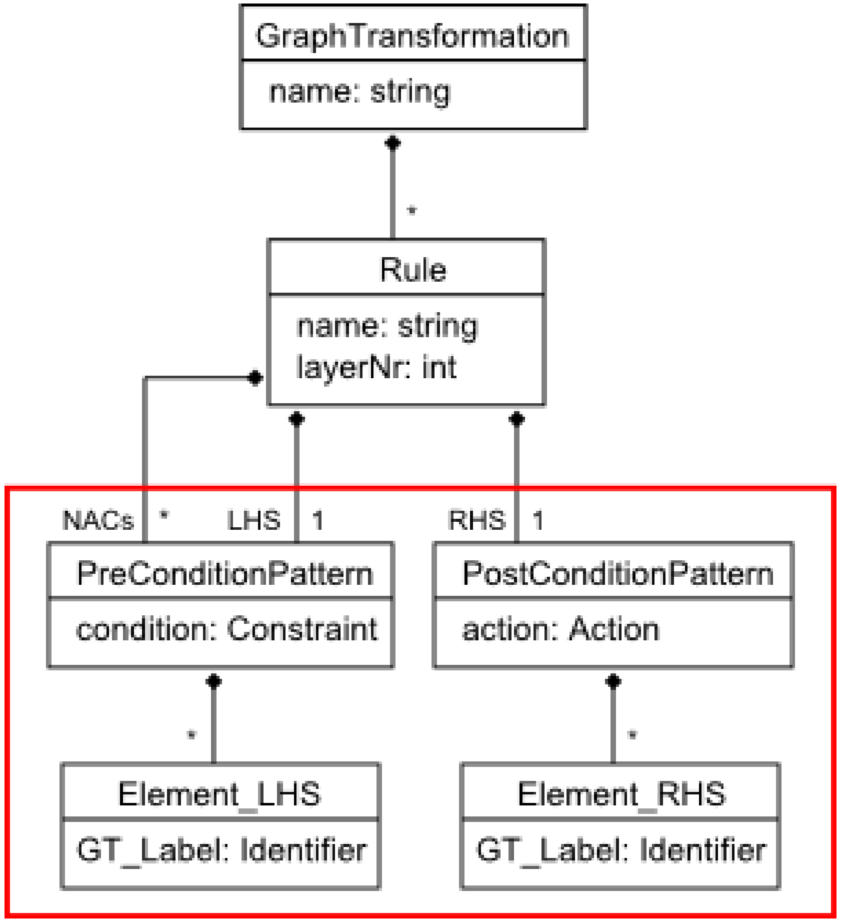
Explicitly Modelling the Transformation Language

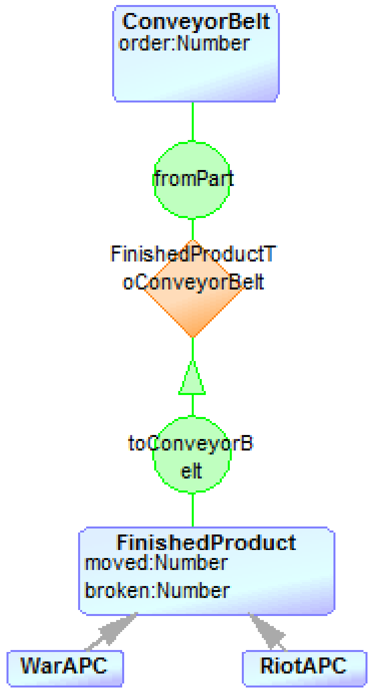




Kühne, T., Mezei, G., Syriani, E., Vangheluwe, H., Wimmer, M., 2010. Explicit transformation modeling. In: Ghosh, S. (Ed.), Models in Software Engineering. Vol. 6002 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 240-255

Explicitly Modelling the Transformation Language

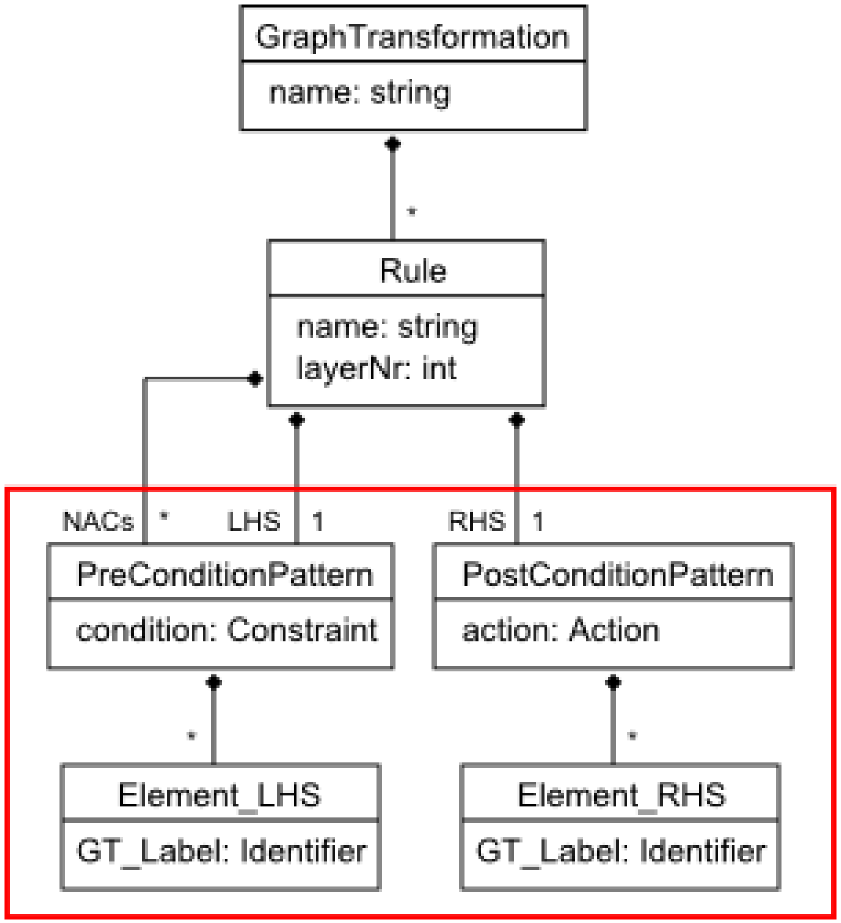


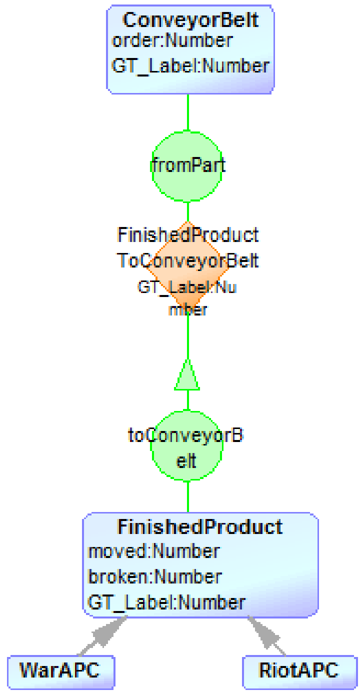


Relax

Kühne, T., Mezei, G., Syriani, E., Vangheluwe, H., Wimmer, M., 2010. Explicit transformation modeling. In: Ghosh, S. (Ed.), Models in Software Engineering. Vol. 6002 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 240-255

Explicitly Modelling the Transformation Language

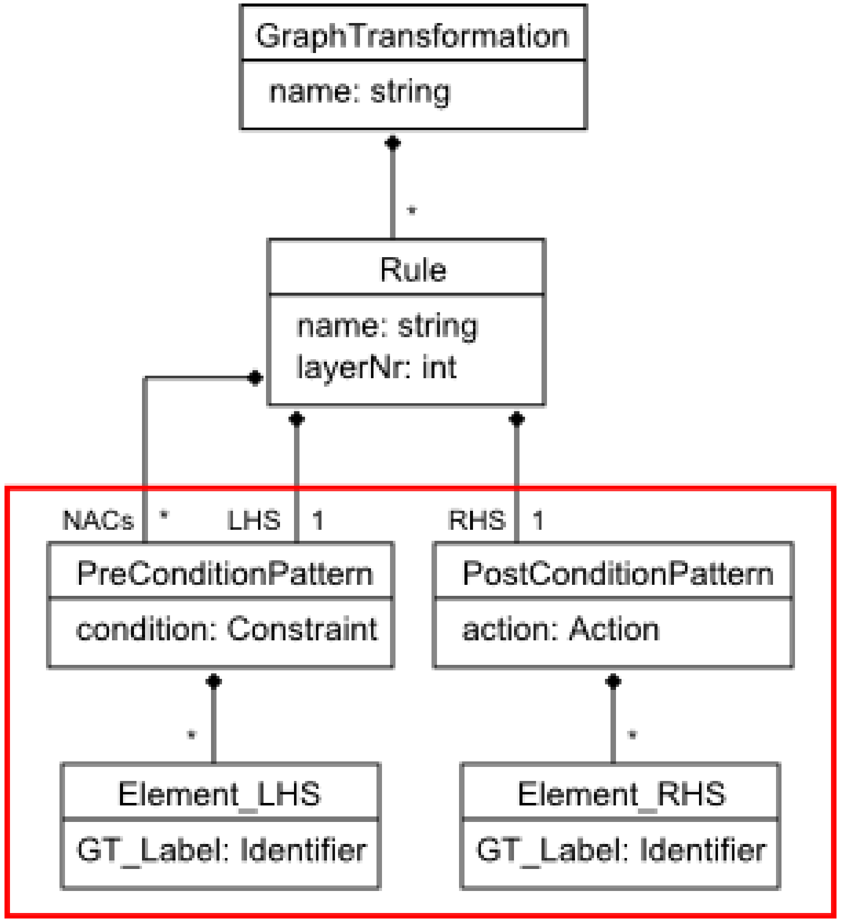


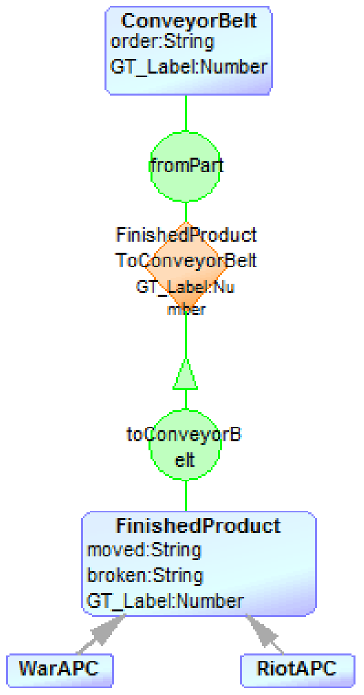


Augment

Kühne, T., Mezei, G., Syriani, E., Vangheluwe, H., Wimmer, M., 2010. Explicit transformation modeling. In: Ghosh, S. (Ed.), Models in Software Engineering. Vol. 6002 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 240-255

Explicitly Modelling the Transformation Language

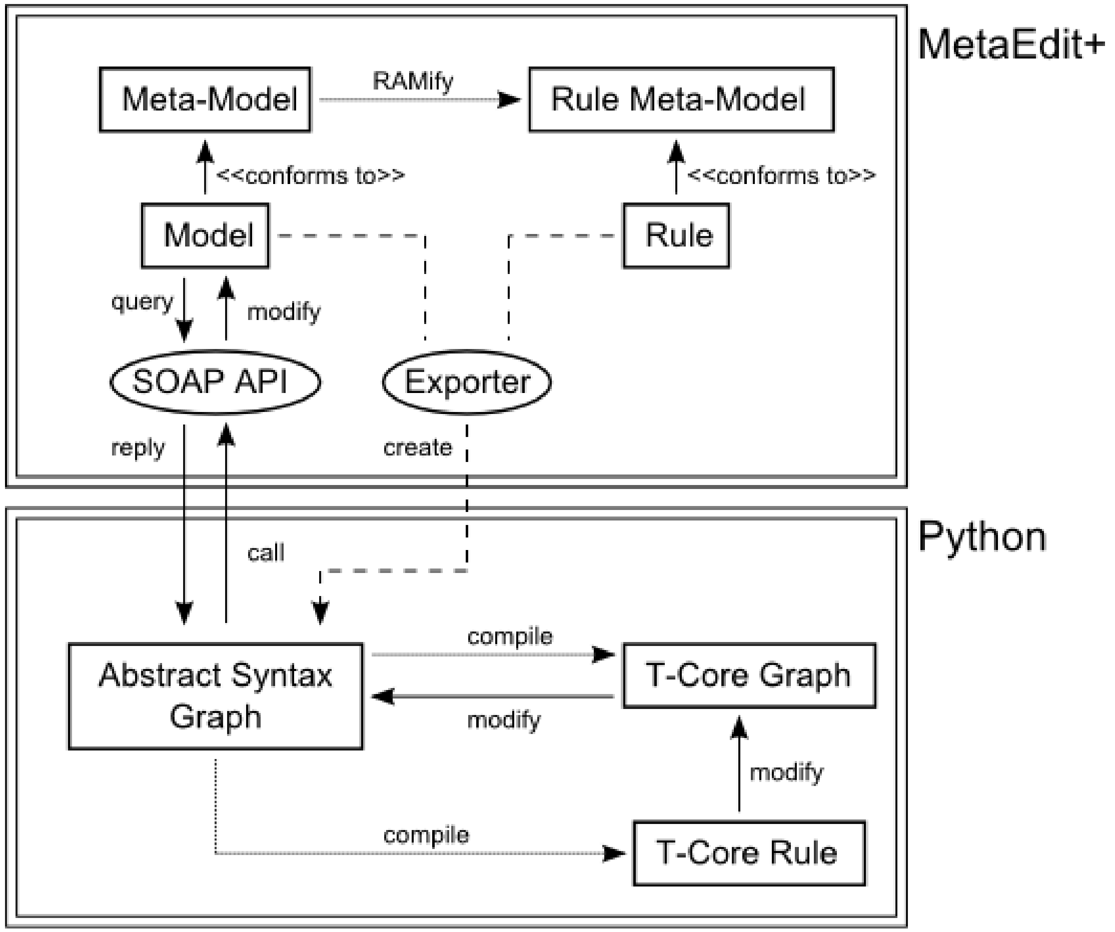




Modify

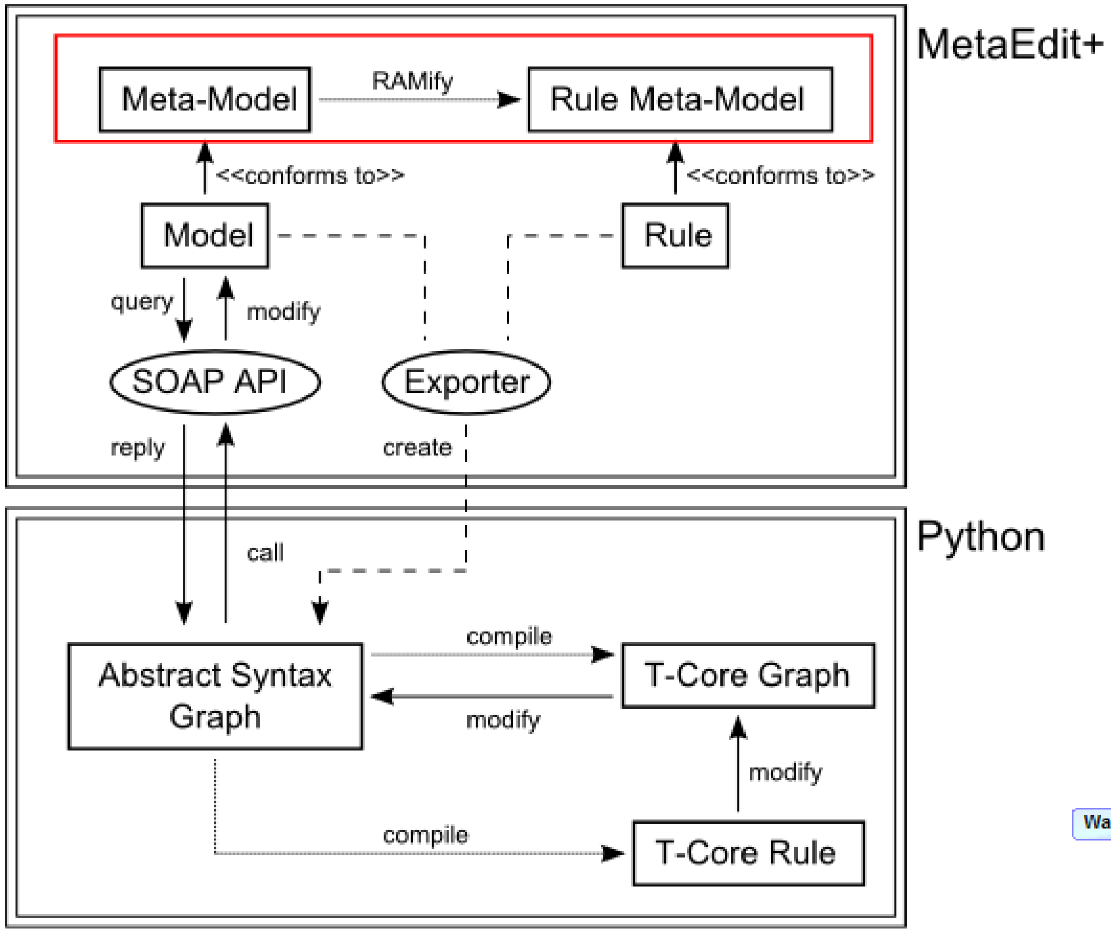
Kühne, T., Mezei, G., Syriani, E., Vangheluwe, H., Wimmer, M., 2010. Explicit transformation modeling. In: Ghosh, S. (Ed.), Models in Software Engineering. Vol. 6002 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 240-255

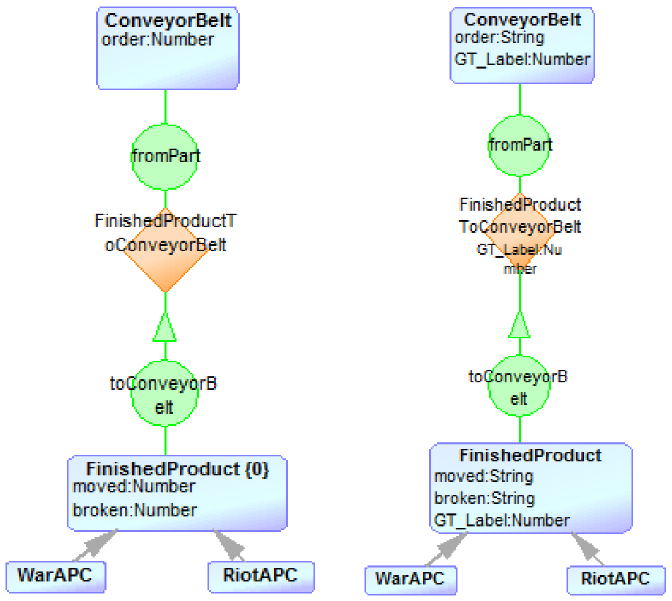
Architecture



Universiteit Antwerpen

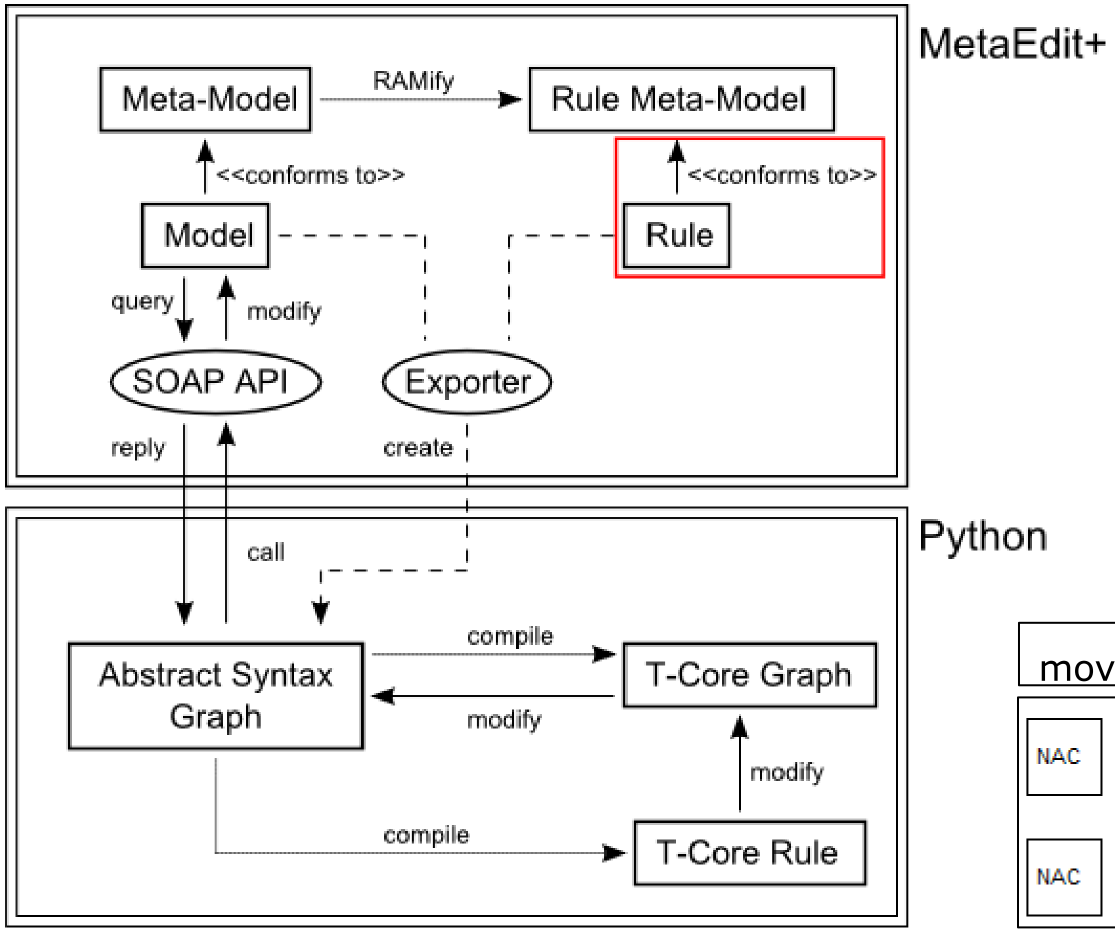
Architecture: Visual Rule Editor

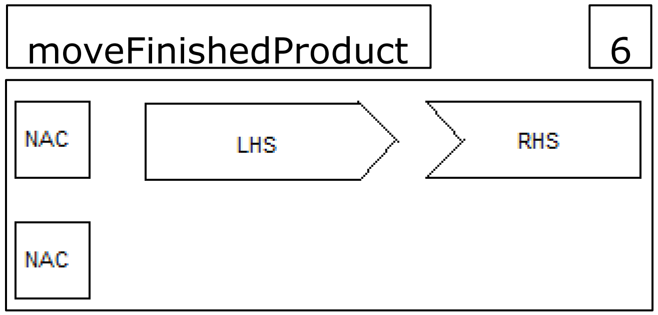




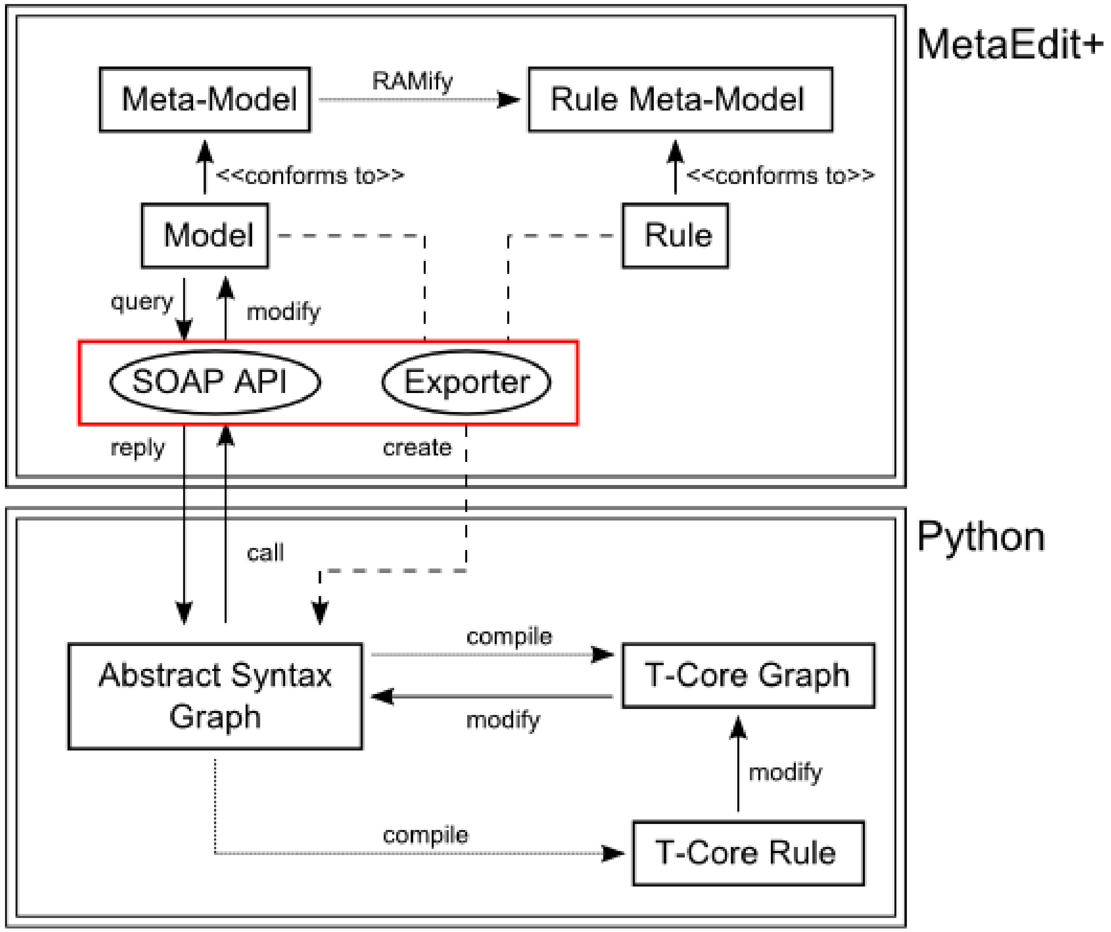
Universiteit Antwerpen

Architecture: Visual Rule Editor



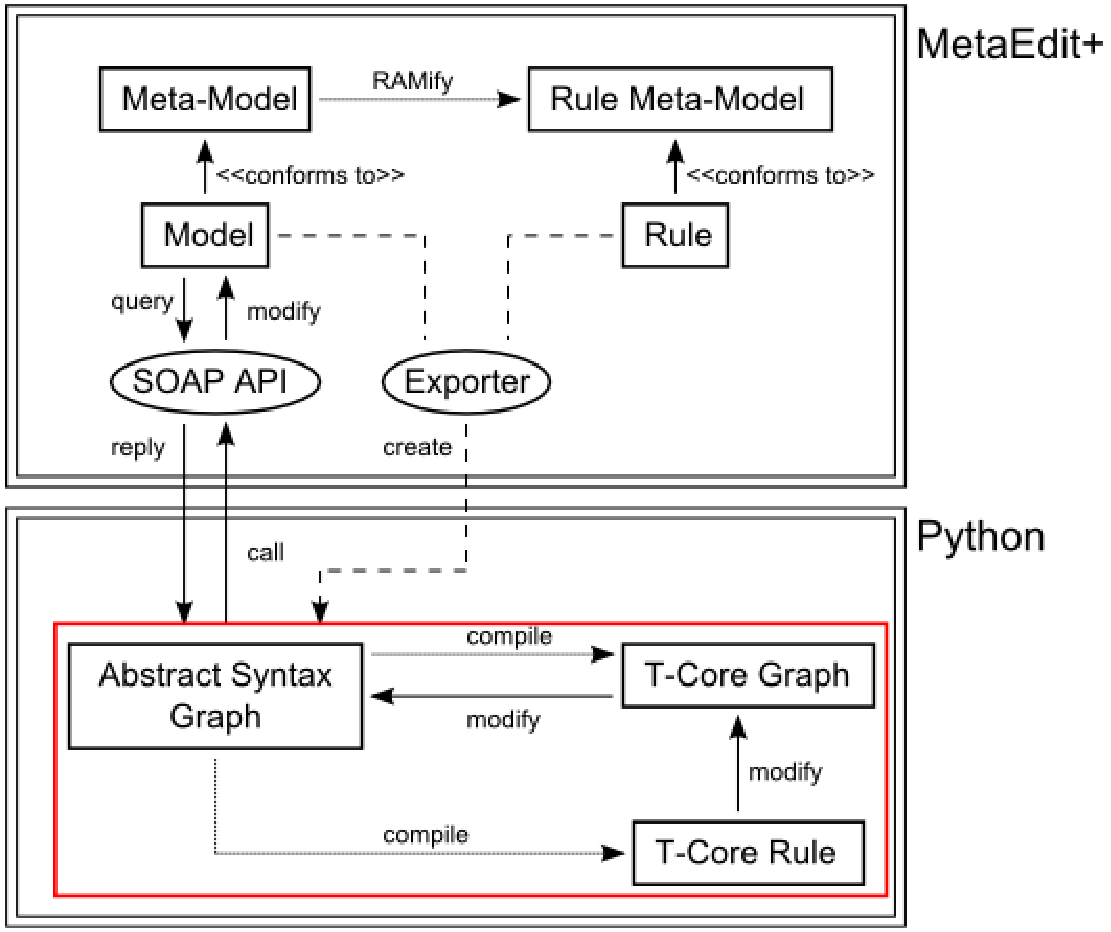


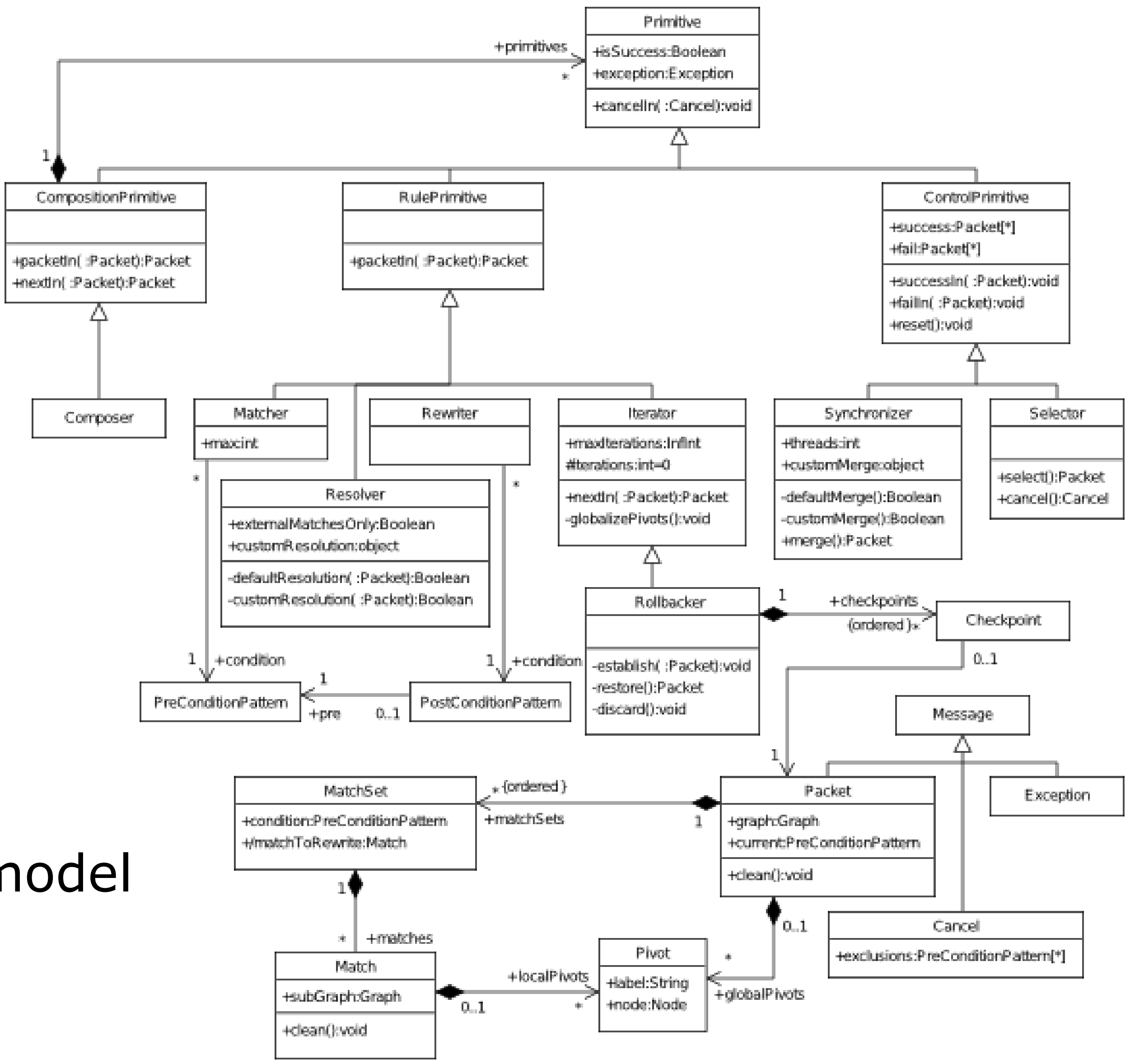
Architecture: MetaEdit+ Components



Universiteit Antwerpen

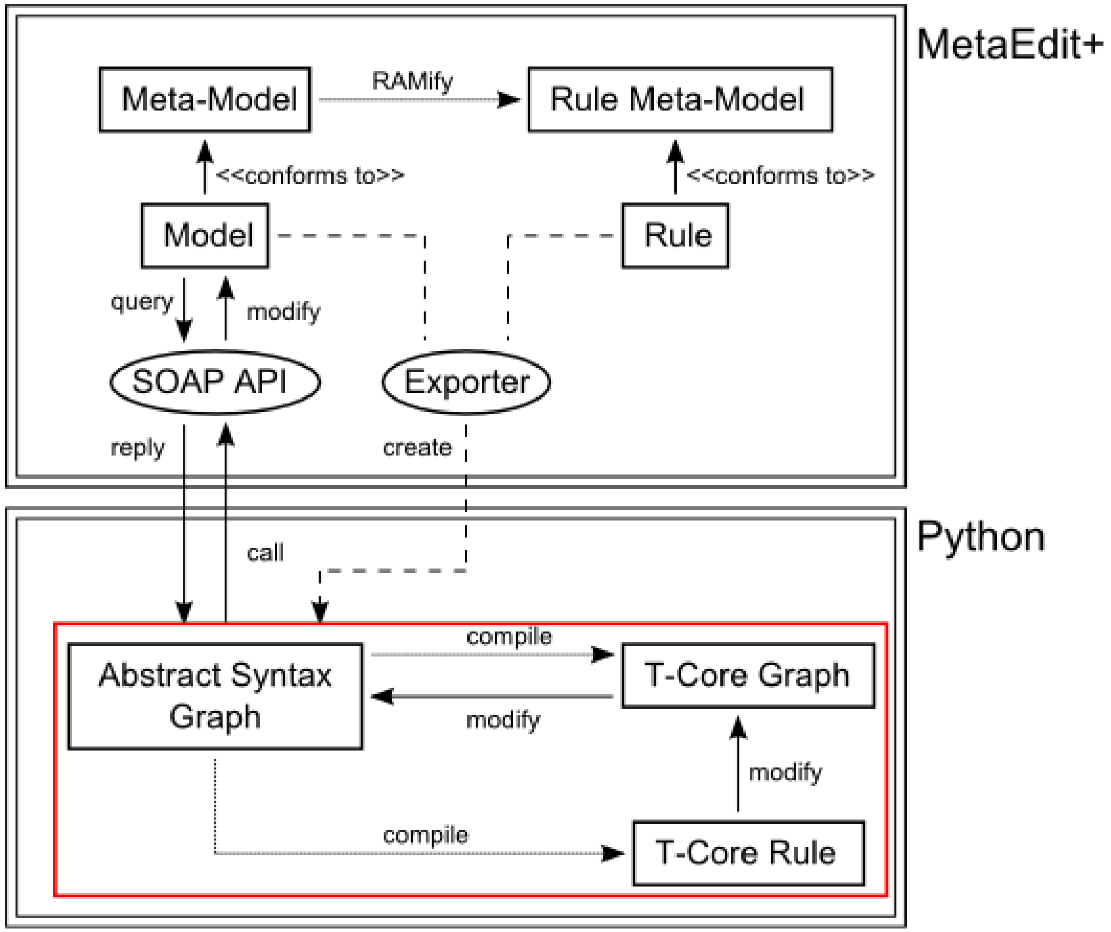
Architecture: Python Backend





T-Core Metamodel

Architecture: Python Backend



Demo

Conclusions

Demonstrated that transformation languages can be modelled explictly

- their concrete and abstract syntax
- their execution semantics (building on a transformation kernel)

… independent of implementation technology

- AToM³ and AToMPM
- **MetaEdit+**

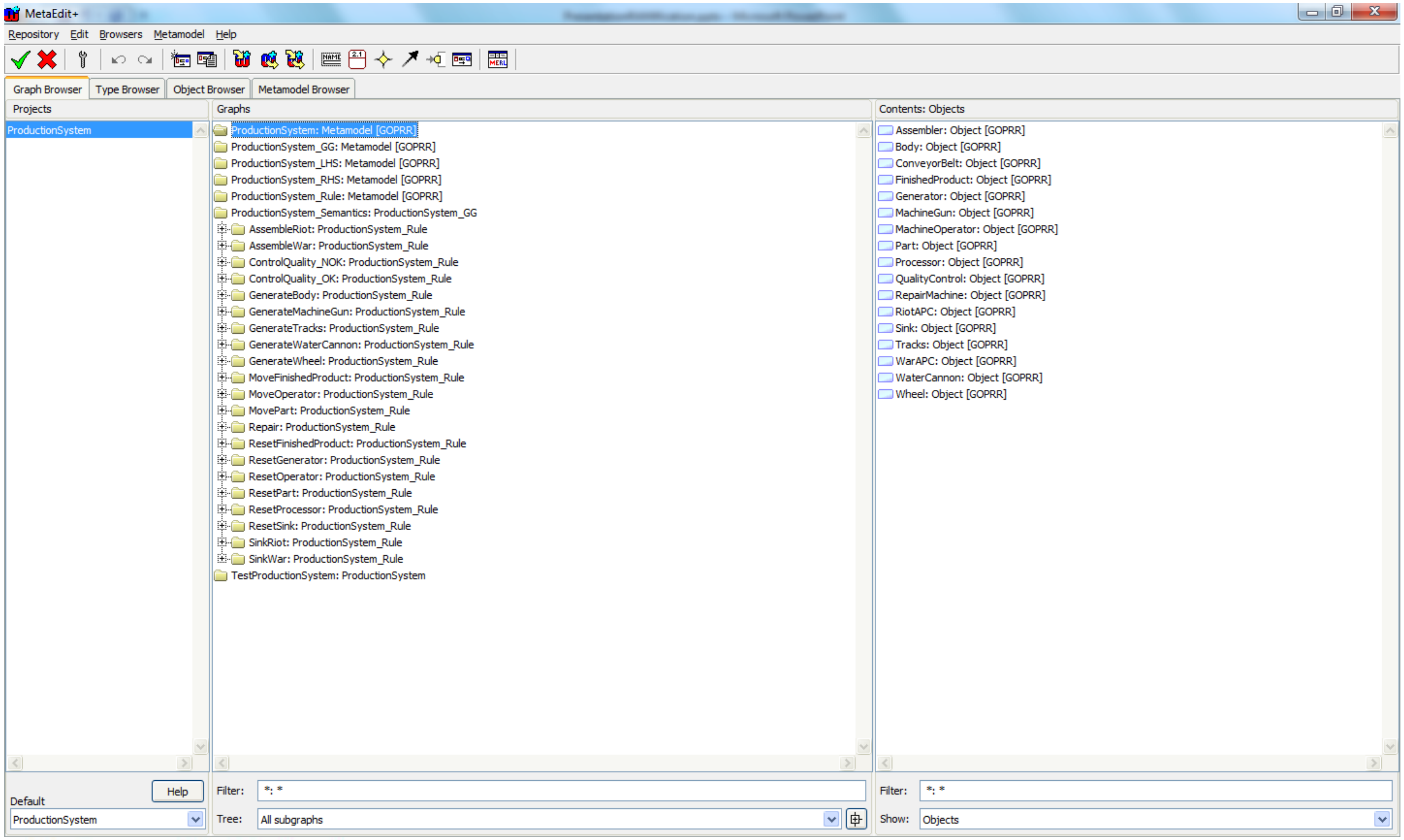
Future Work

- Explicit modelling of CS in transformation rules
- RAMification process for CS – Mapping – AS
- Automatic RAMification
  - Already implemented in AToM³ and AToMPM
  - Now also possible in MetaEdit+
- Other Environments
  - Eclipse/GMF
  - GrGen.NET

Questions?

simon.vanmierlo@student.ua.ac.be

Appendix: Screengrabs


MetaEdit+
Repository Edit Browsers Metamodel Help
Graph Browser Type Browser Object Browser Metamodel Browser

Projects

- ProductionSystem

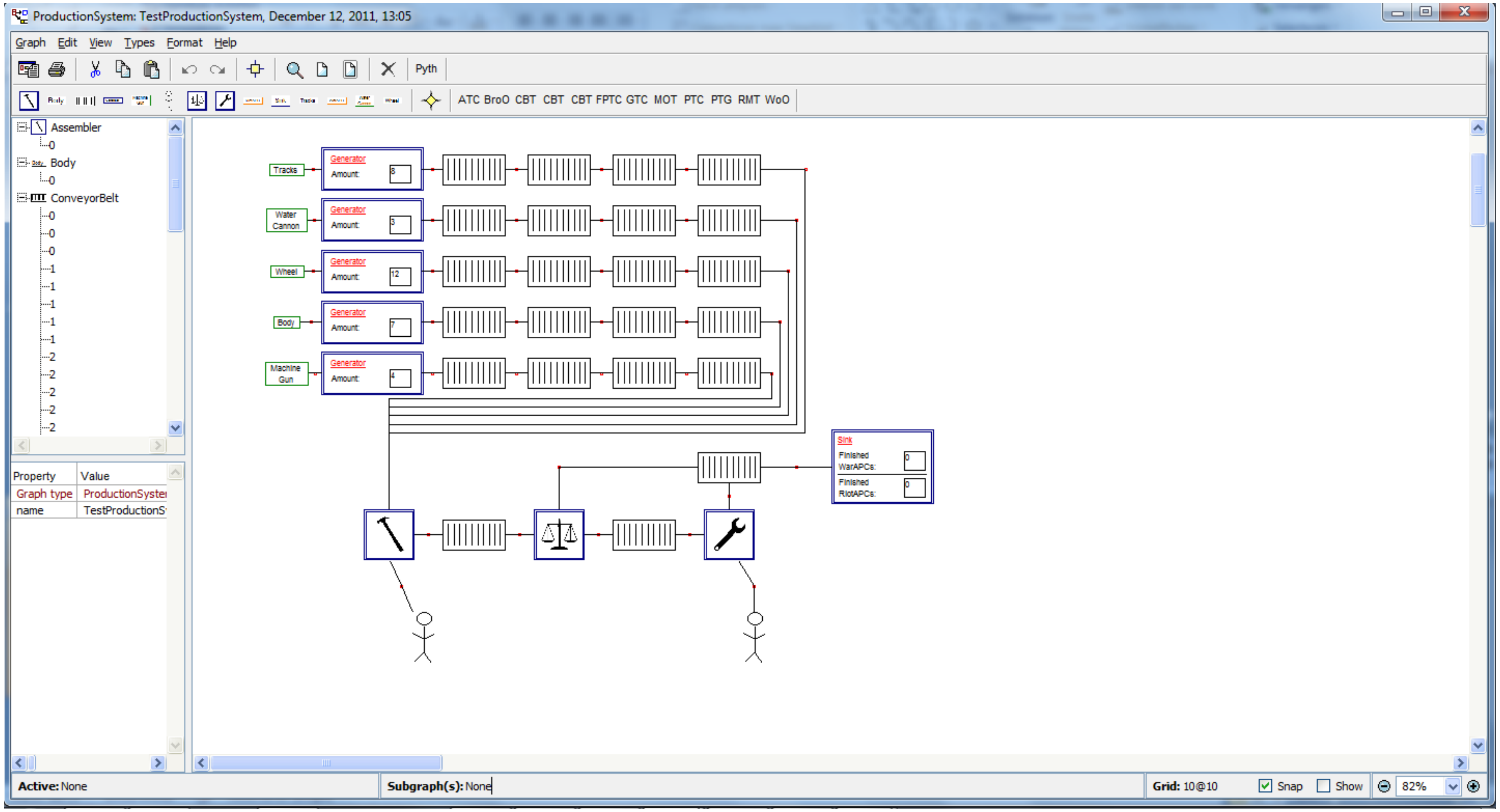
Graphs

- ProductionSystem: Metamodel [GOPRR]
- ProductionSystem_GG: Metamodel [GOPRR]
- ProductionSystem_LHS: Metamodel [GOPRR]
- ProductionSystem_RHS: Metamodel [GOPRR]
- ProductionSystem_Rule: Metamodel [GOPRR]
- ProductionSystem_Semantics: ProductionSystem_GG
  - AssembleRiot: ProductionSystem_Rule
  - AssembleWar: ProductionSystem_Rule
  - ControlQuality_NOK: ProductionSystem_Rule
  - ControlQuality_OK: ProductionSystem_Rule
  - GenerateBody: ProductionSystem_Rule
  - GenerateMachineGun: ProductionSystem_Rule
  - GenerateTracks: ProductionSystem_Rule
  - GenerateWaterCannon: ProductionSystem_Rule
  - GenerateWheel: ProductionSystem_Rule
  - MoveFinishedProduct: ProductionSystem_Rule
  - MoveOperator: ProductionSystem_Rule
  - MovePart: ProductionSystem_Rule
  - Repair: ProductionSystem_Rule
  - ResetFinishedProduct: ProductionSystem_Rule
  - ResetGenerator: ProductionSystem_Rule
  - ResetOperator: ProductionSystem_Rule
  - ResetPart: ProductionSystem_Rule
  - ResetProcessor: ProductionSystem_Rule
  - ResetSink: ProductionSystem_Rule
  - SinkRiot: ProductionSystem_Rule
  - SinkWar: ProductionSystem_Rule
- TestProductionSystem: ProductionSystem

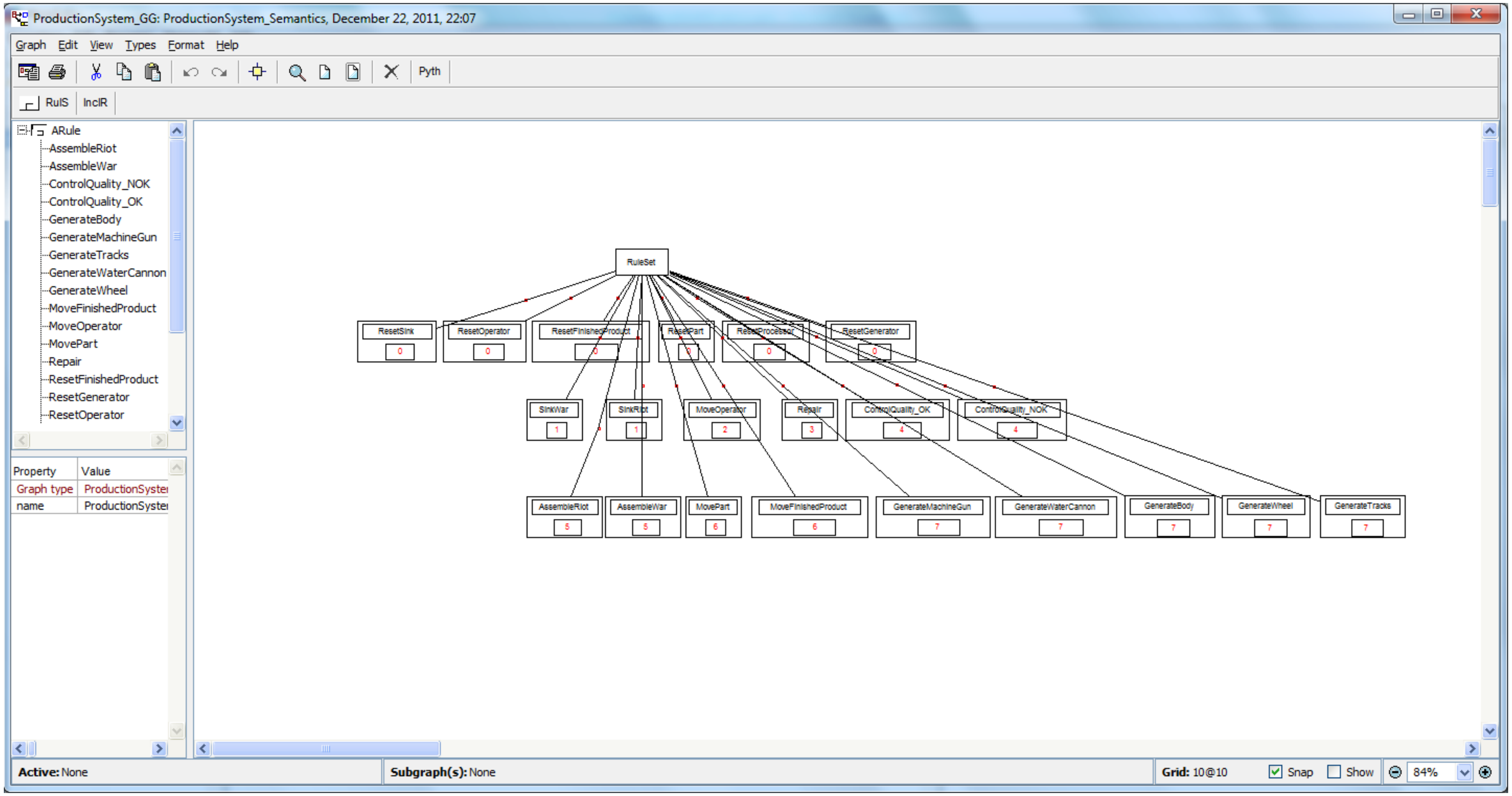
Contents: Objects

- Assembler: Object [GOPRR]
- Body: Object [GOPRR]
- ConveyorBelt: Object [GOPRR]
- FinishedProduct: Object [GOPRR]
- Generator: Object [GOPRR]
- MachineGun: Object [GOPRR]
- MachineOperator: Object [GOPRR]
- Part: Object [GOPRR]
- Processor: Object [GOPRR]
- QualityControl: Object [GOPRR]
- RepairMachine: Object [GOPRR]
- RiotAPC: Object [GOPRR]
- Sink: Object [GOPRR]
- Tracks: Object [GOPRR]
- WarAPC: Object [GOPRR]
- WaterCannon: Object [GOPRR]
- Wheel: Object [GOPRR]

Default: ProductionSystem
Help
Filter: *: *
Tree: All subgraphs
Filter: *: *
Show: Objects



Property	Value
Graph type	ProductionSyste
name	TestProductionS

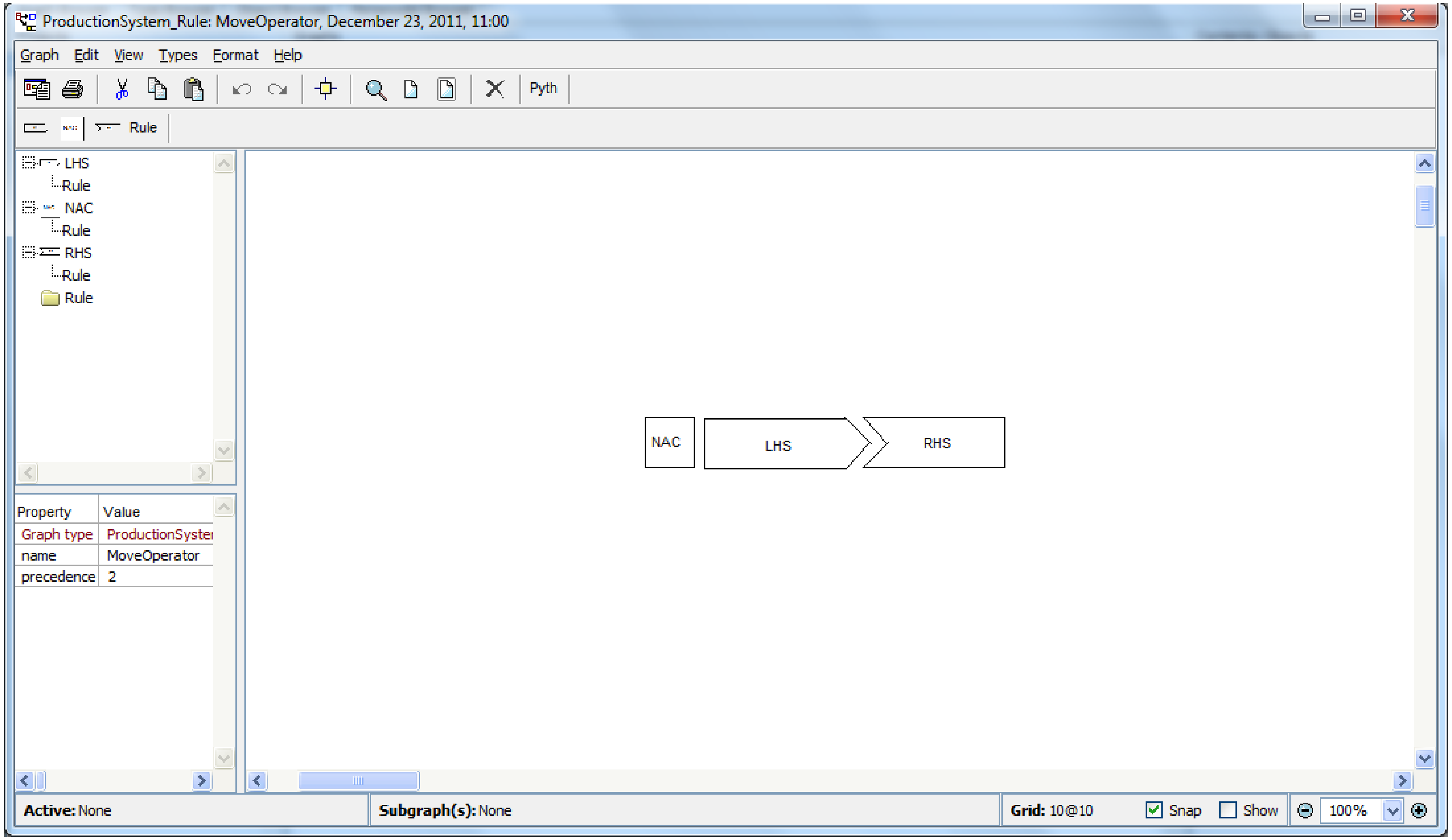


ProductionSystem_GG: ProductionSystem_Semantics, December 22, 2011, 22:07

ARule
- AssembleRiot
- AssembleWar
- ControlQuality_NOK
- ControlQuality_OK
- GenerateBody
- GenerateMachineGun
- GenerateTracks
- GenerateWaterCannon
- GenerateWheel
- MoveFinishedProduct
- MoveOperator
- MovePart
- Repair
- ResetFinishedProduct
- ResetGenerator
- ResetOperator

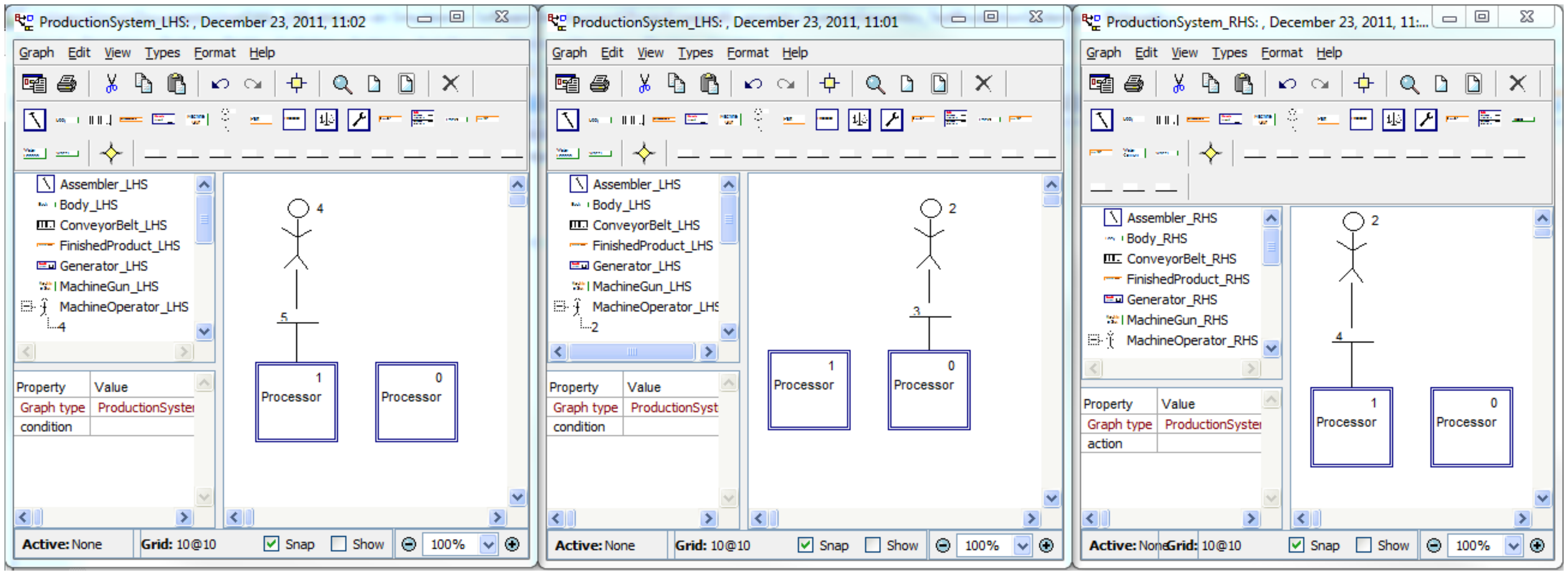
Property	Value
Graph type	ProductionSyste
name	ProductionSyste

Active: None **Subgraph(s):** None **Grid:** 10@10 [✓ Snap] Show 84%


ProductionSystem_Rule: MoveOperator, December 23, 2011, 11:00
Graph Edit View Types Format Help
Pyth
Rule
LHS
Rule
NAC
Rule
RHS
Rule
Rule
NAC
LHS
RHS
Grid: 10@10
Snap
Show
100%

Property	Value
Graph type	ProductionSyster
name	MoveOperator
precedence	2

Active: None **Subgraph(s):** None



Property	Value
Graph type	ProductionSyste
condition	

Property	Value
Graph type	ProductionSyst
condition	

Property	Value
Graph type	ProductionSyste
action	