Comparing graphical DSL editors

AToM^3 vs GMF & MetaEdit+

Nick Baetens

Universiteit Antwerpen

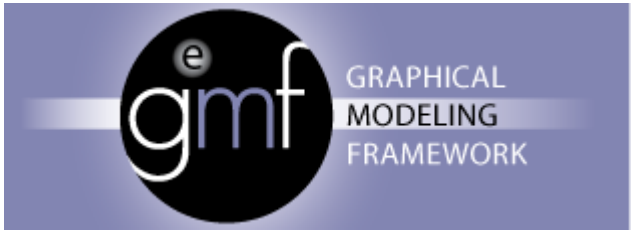
Outline

- Introduction
- MetaEdit+
  - Specifications
  - Workflow
- GMF
  - Specifications
  - Workflow
- Comparison

Introduction

MetaEdit+

- Commercial
- Written in Smalltalk
- Standalone



- Eclipse plug-in
- Depends on & combines other plug-ins

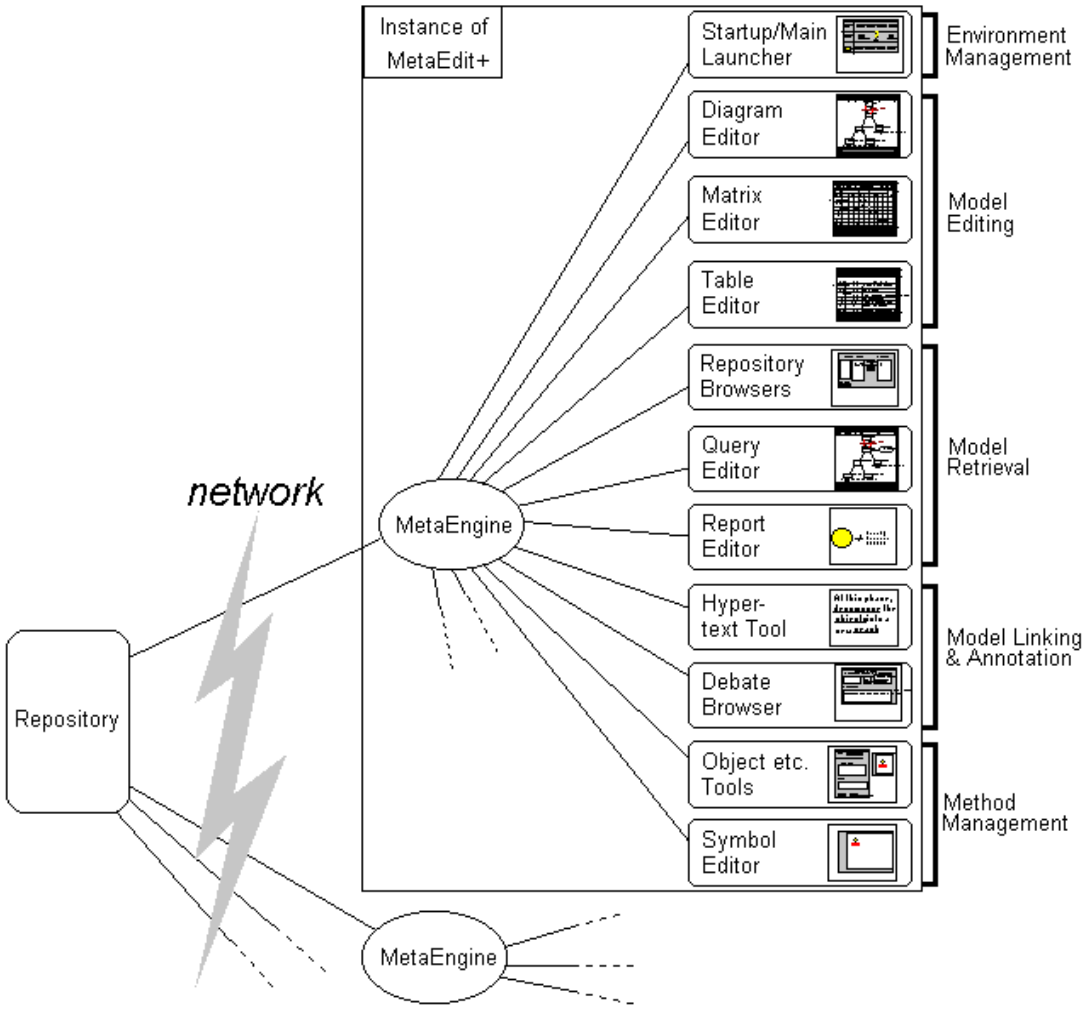
Outline

- Introduction
- MetaEdit+
  - Specifications <=
  - Workflow
- GMF
  - Specifications
  - Workflow
- Comparison

MetaEdit+

- Graph, Object, Port, Property, Relationship and Role
- Graph: Top-level structure of meta-model
- Binding of objects, relationships, roles and ports within graph = actual semantics
- Tools for each base type

MetaEdit+


Instance of MetaEdit+
Startup/Main Launcher
Diagram Editor
Matrix Editor
Table Editor
Repository Browsers
Query Editor
Report Editor
Hyper-text Tool
Debate Browser
Object etc. Tools
Symbol Editor
Environment Management
Model Editing
Model Retrieval
Model Linking & Annotation
Method Management
network
MetaEngine
MetaEngine
Repository

MetaEdit+

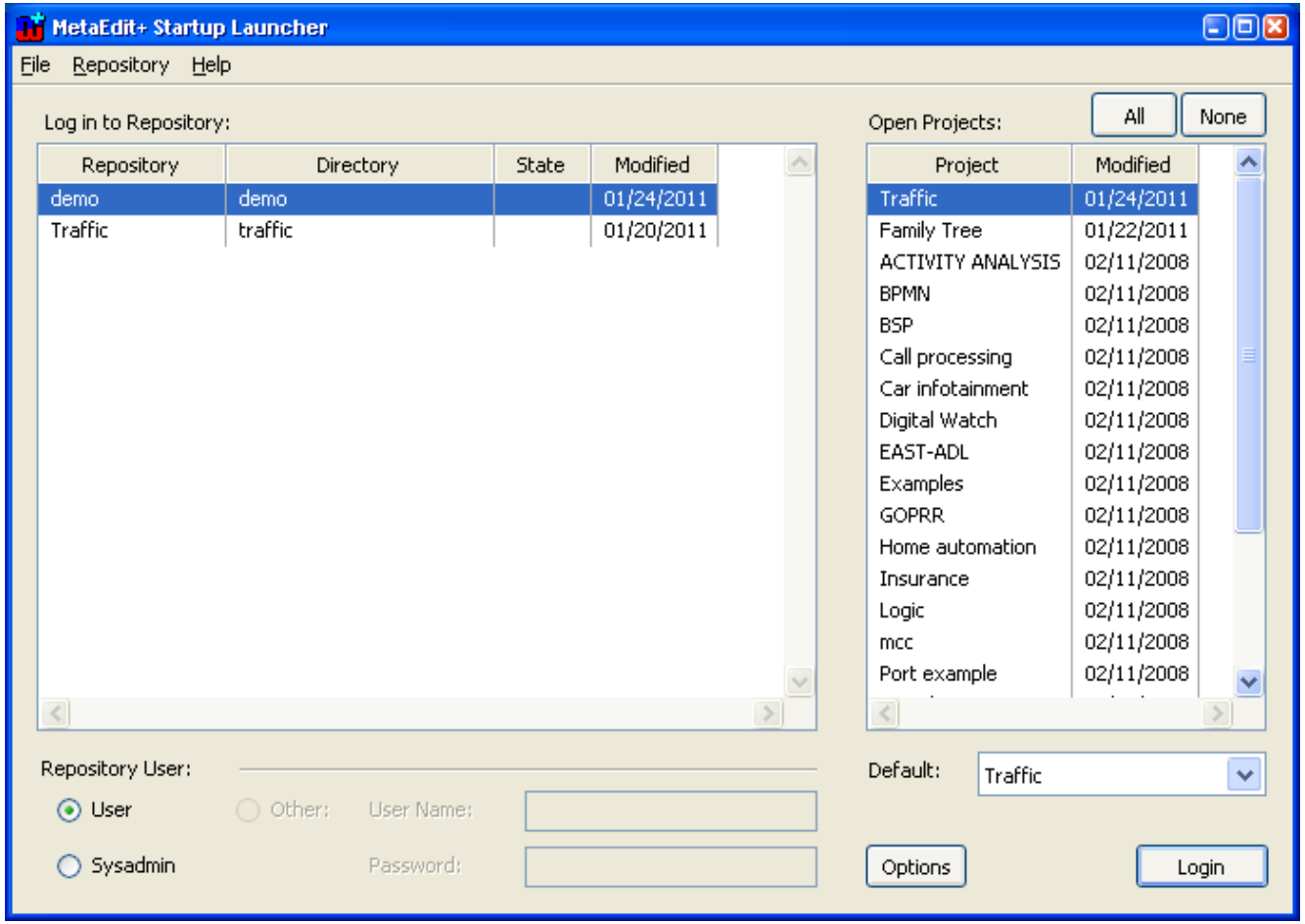
- Information in instance models created with the older version of the meta-model is not lost when the new version is deployed

- Conservative approach
  - If concept is removed
  - Creation of new instances impossible
  - Existing instances are not removed from models
  - Generators will still produce working code from old instances.

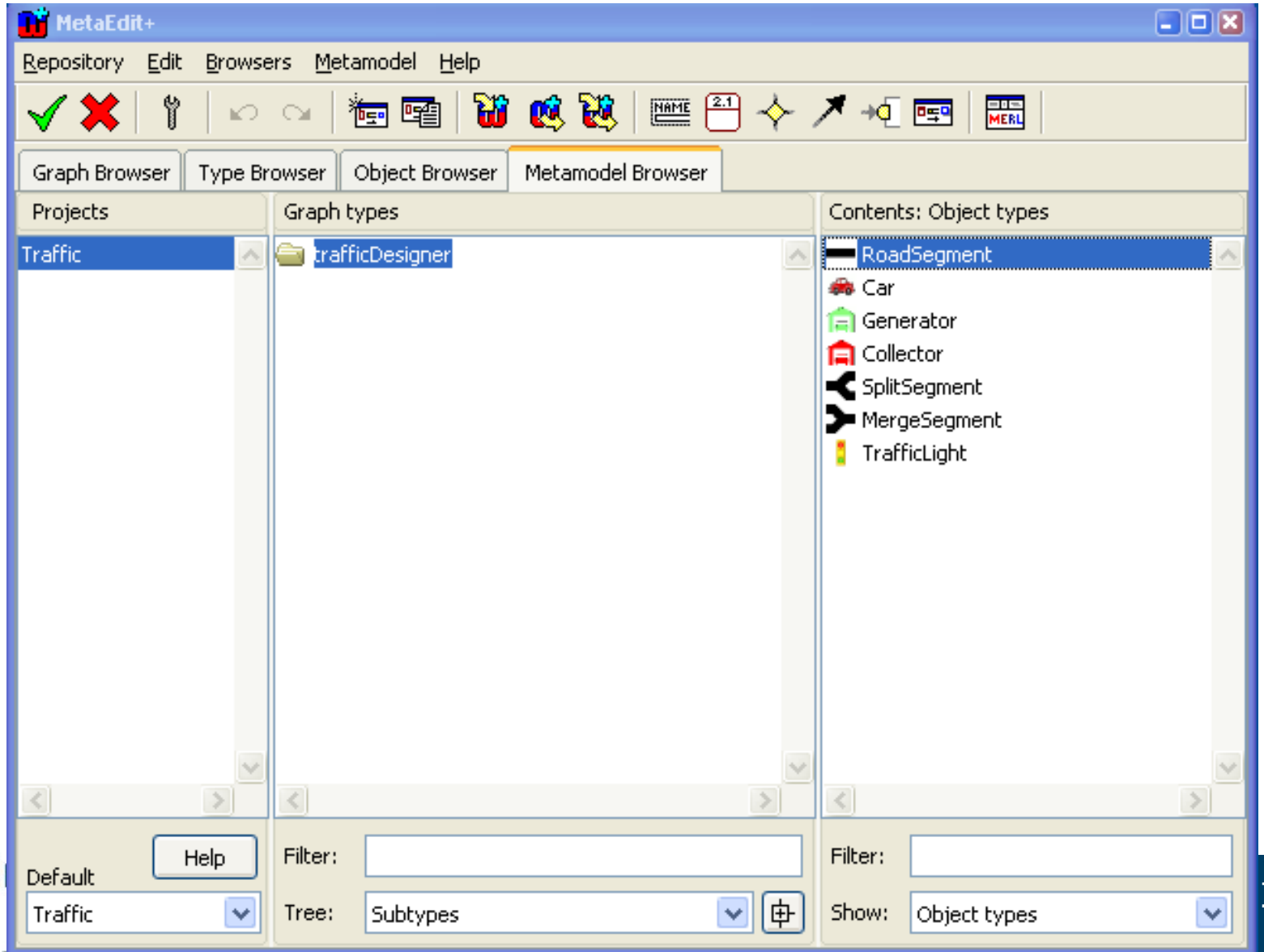
Outline

- Introduction
- MetaEdit+
  - Specifications
  - Workflow <=
- GMF
  - Specifications
  - Workflow
- Comparison

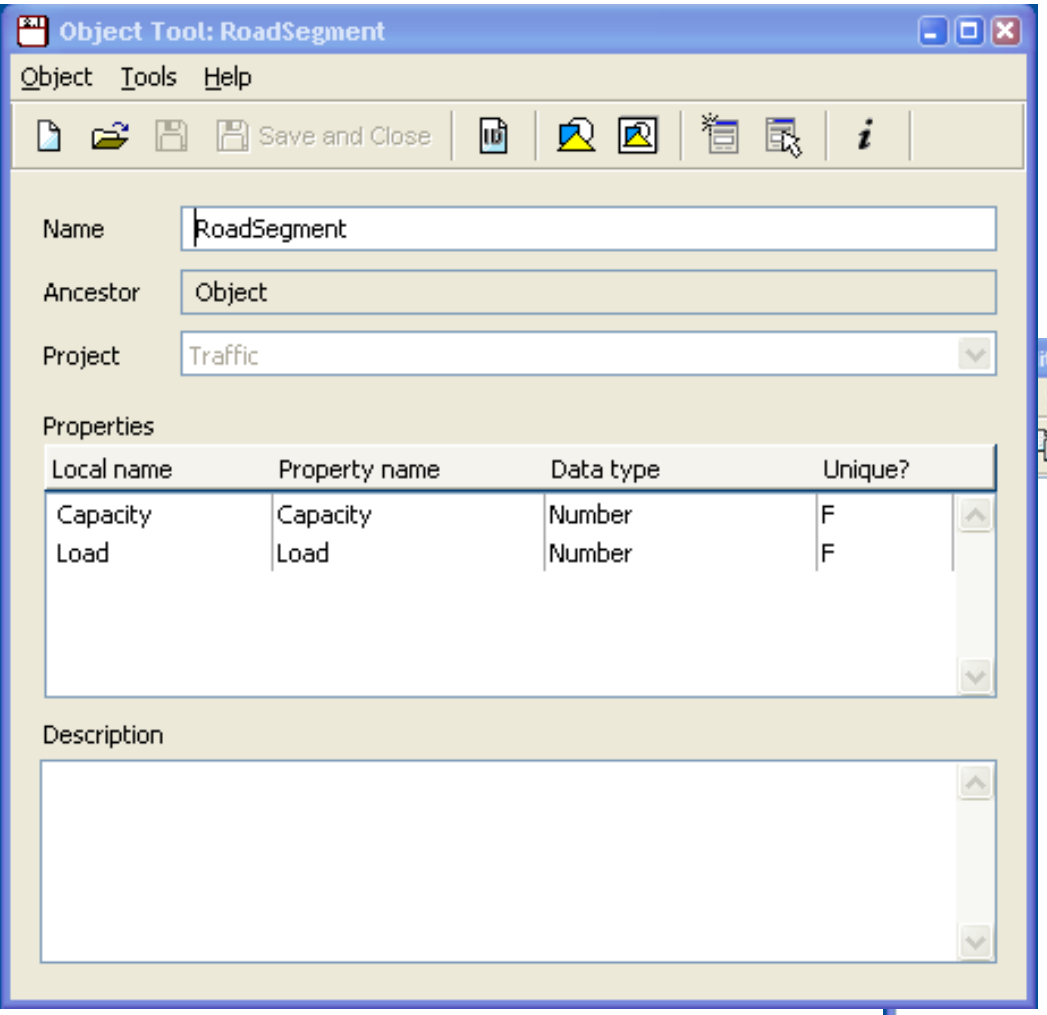
MetaEdit+ Workbench

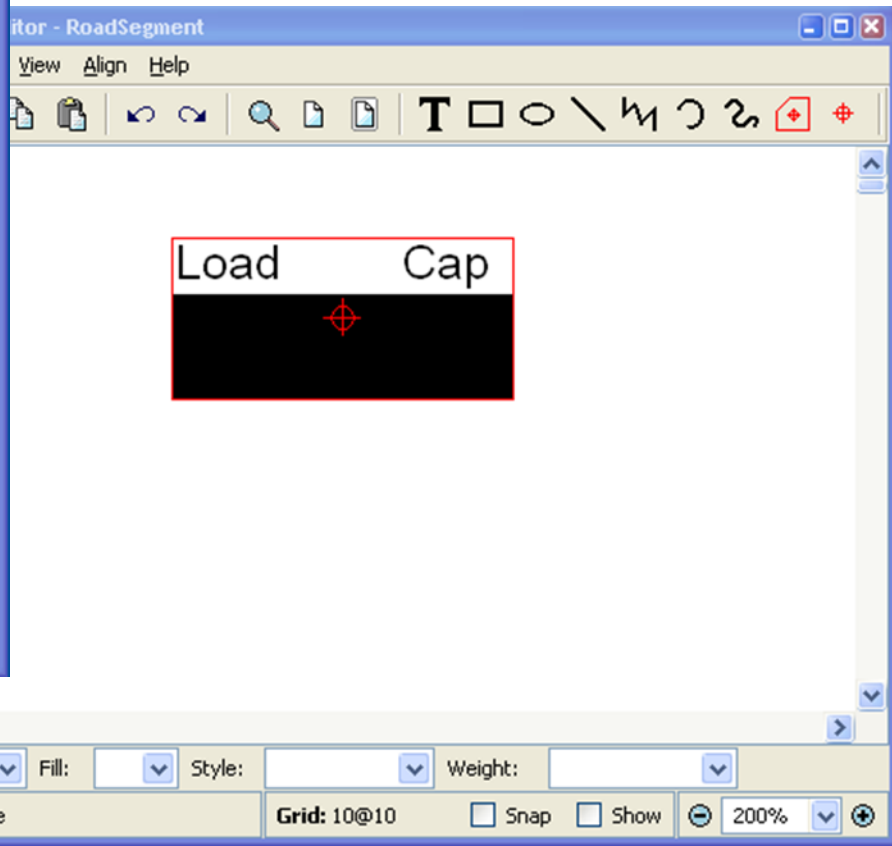


MetaEdit+ Workbench



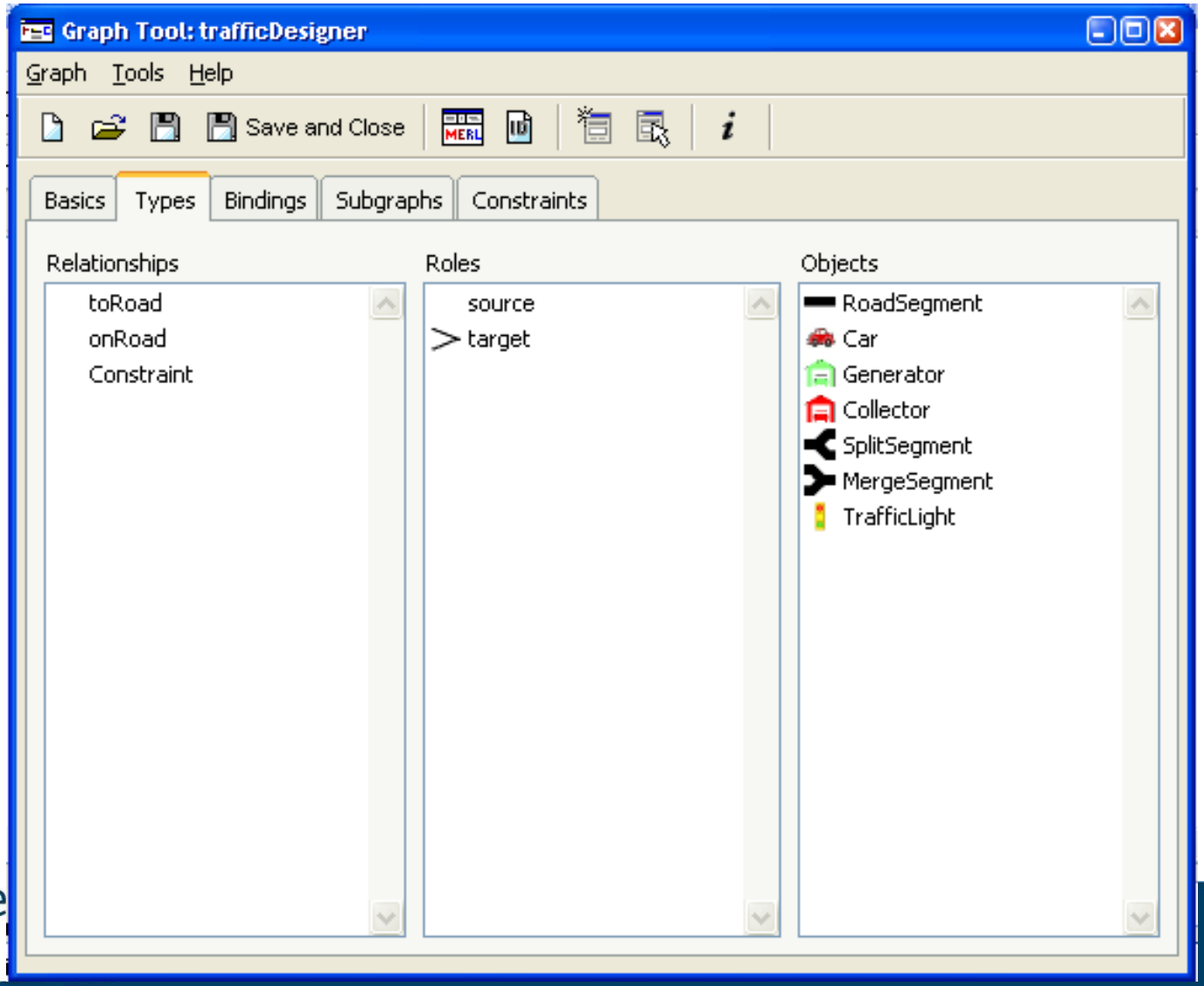
MetaEdit+ Workbench



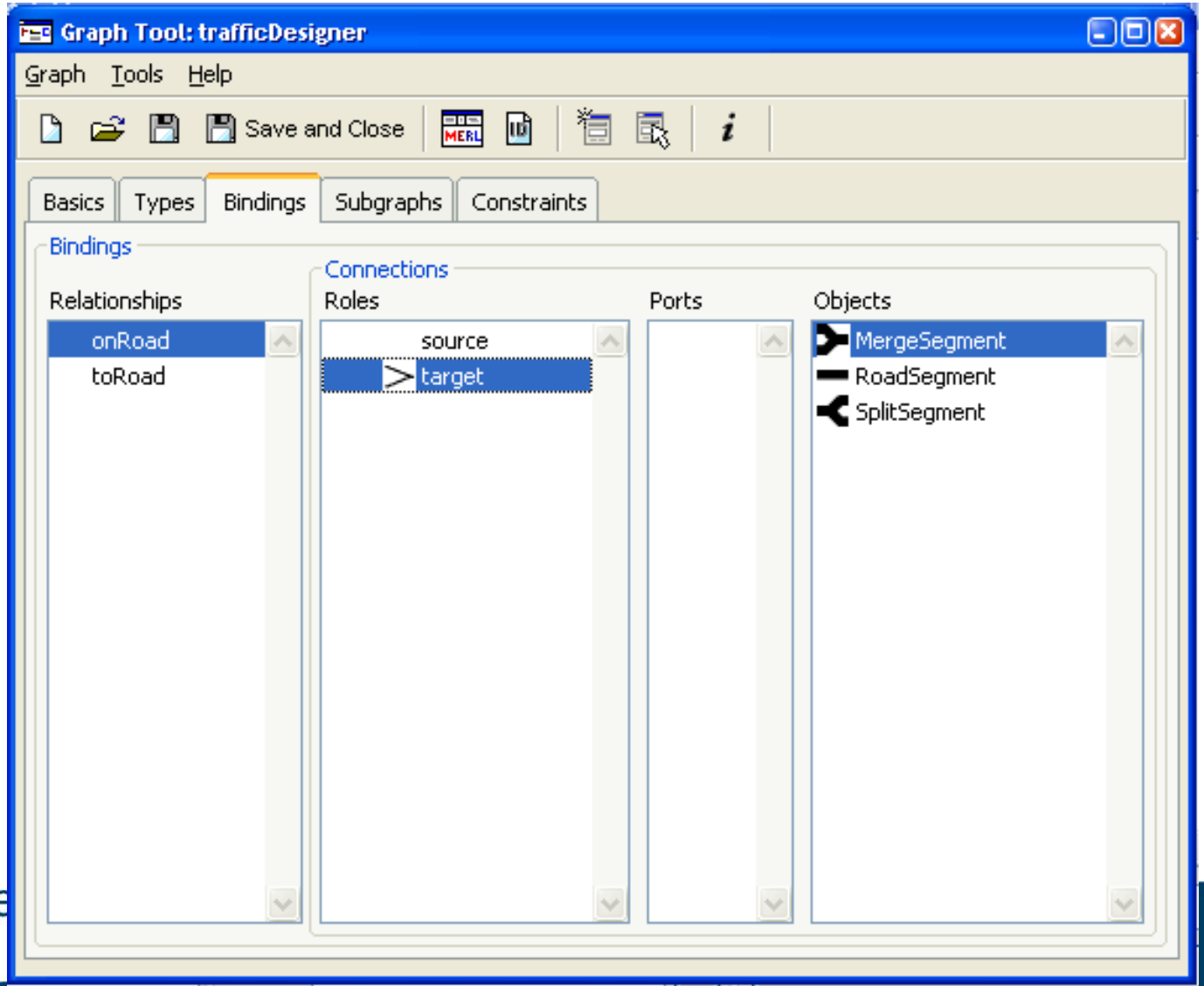


Universiteit Antwerpen

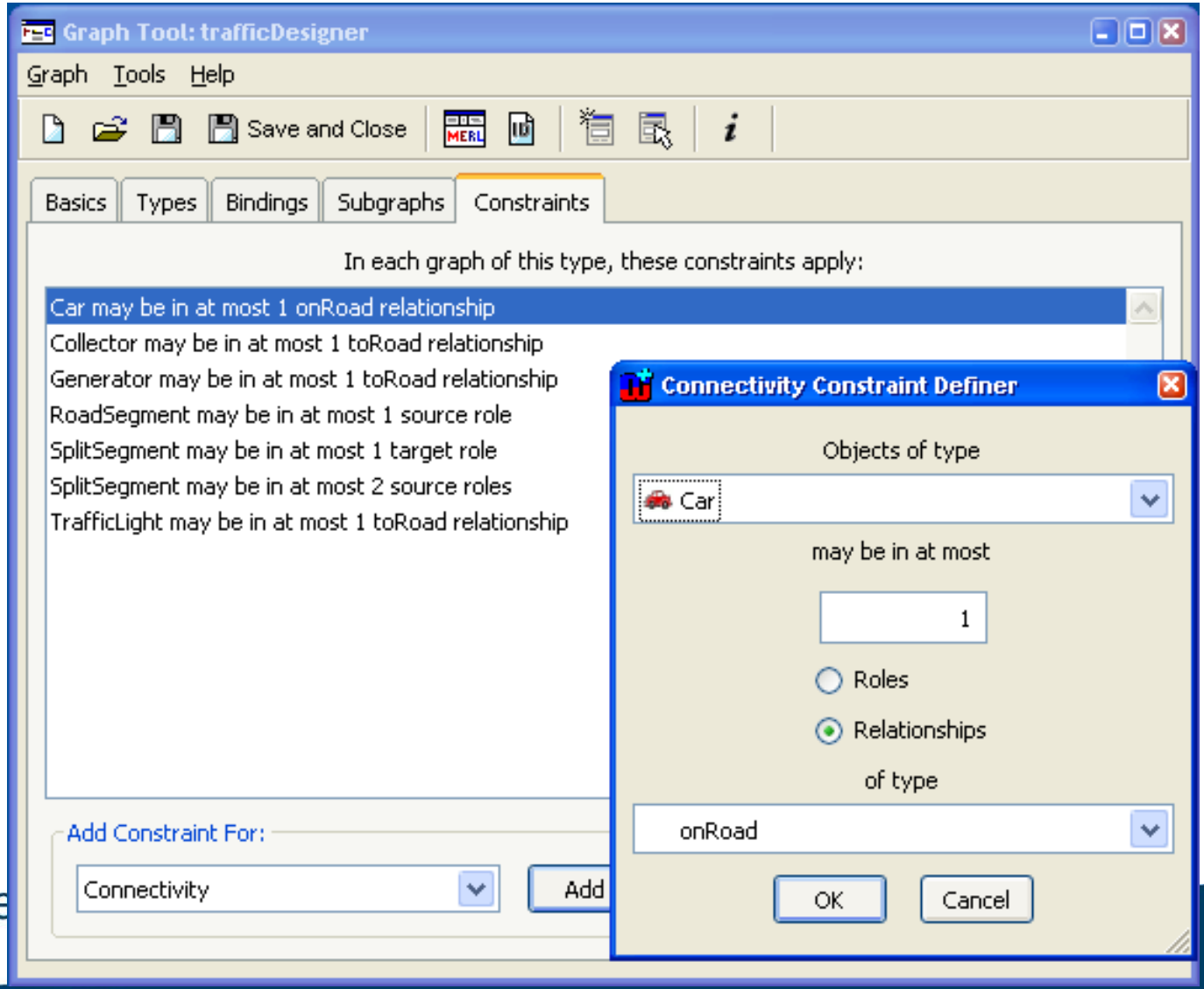
MetaEdit+ Workbench



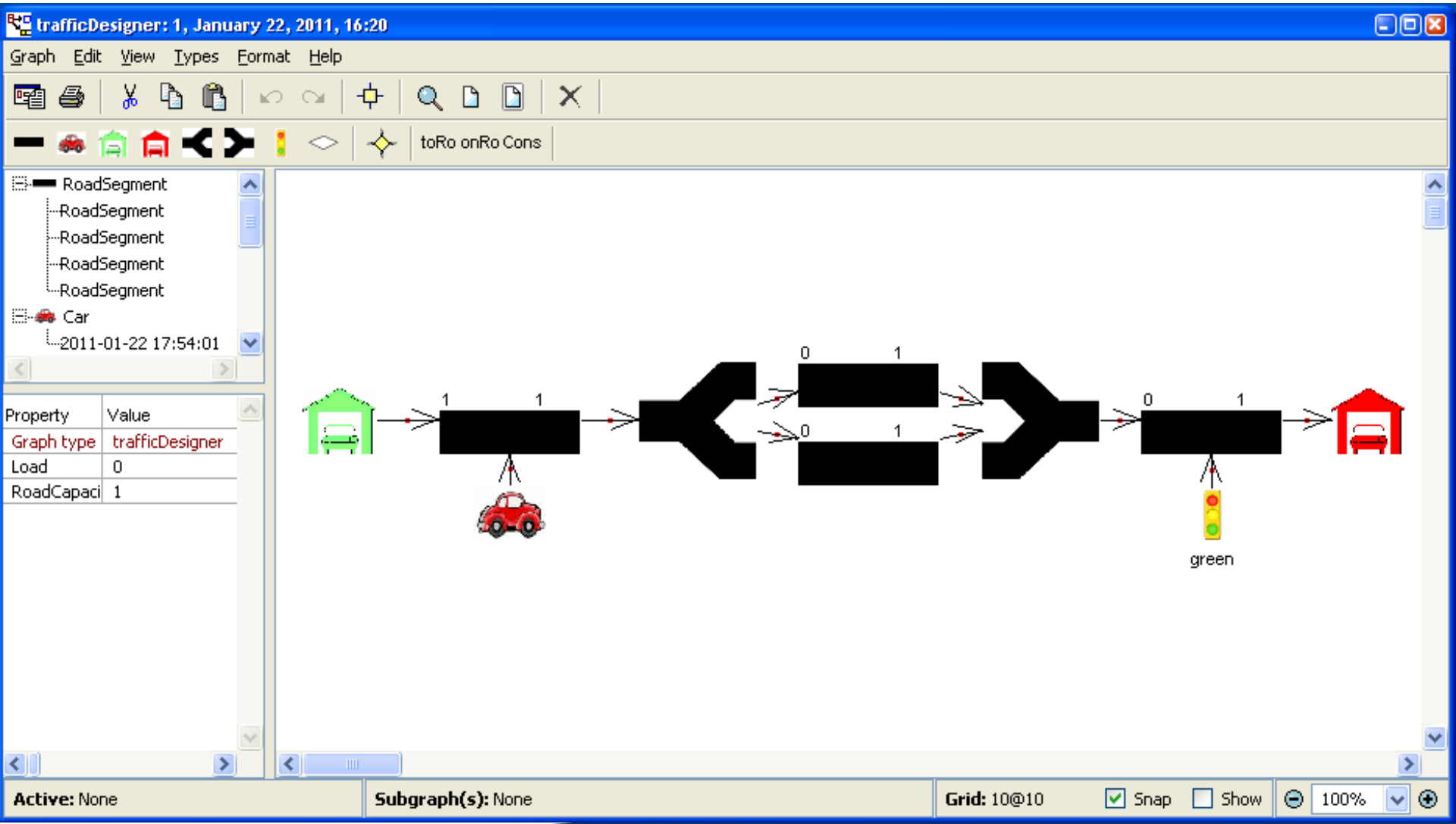
MetaEdit+ Workbench



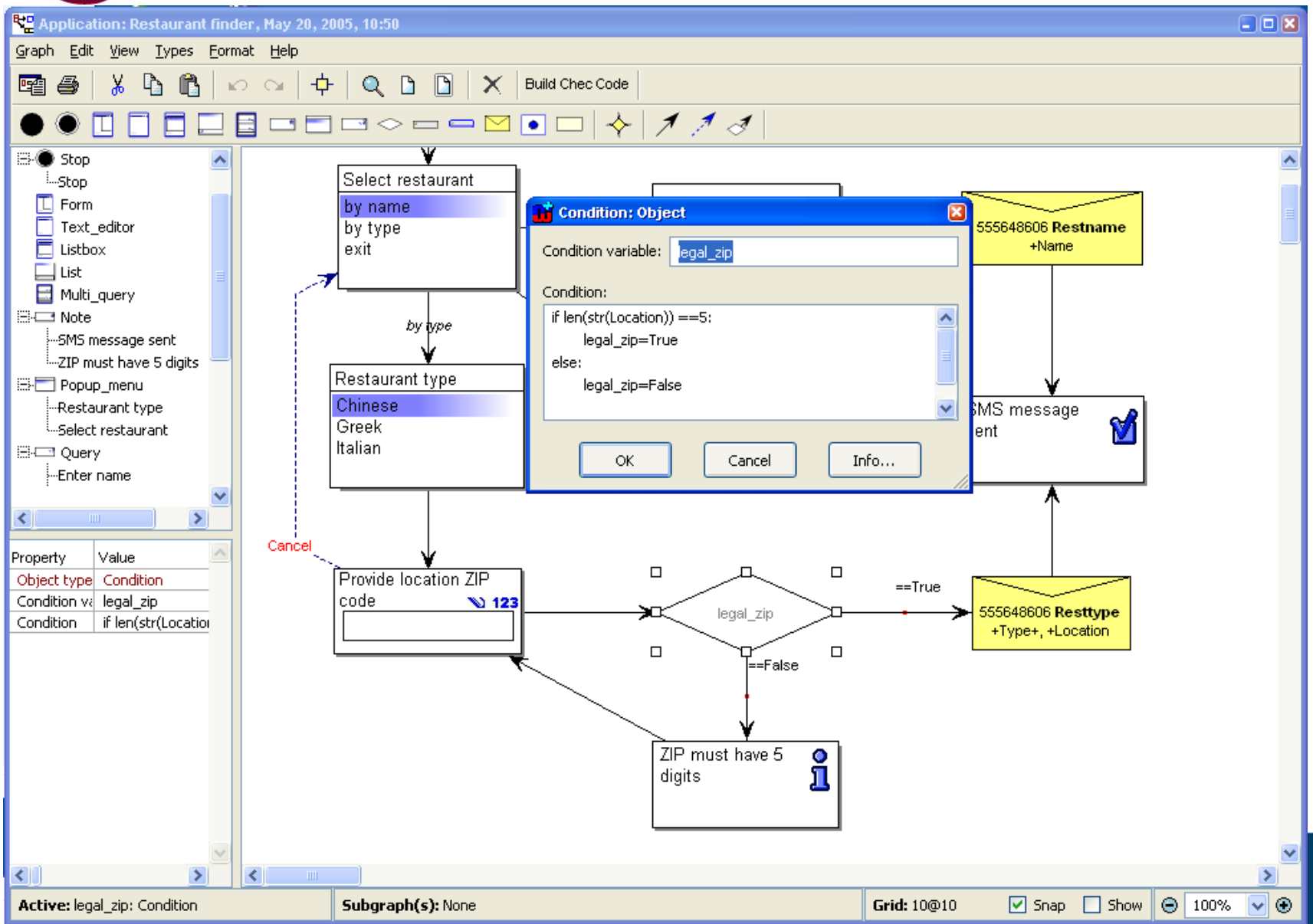
MetaEdit+ Workbench



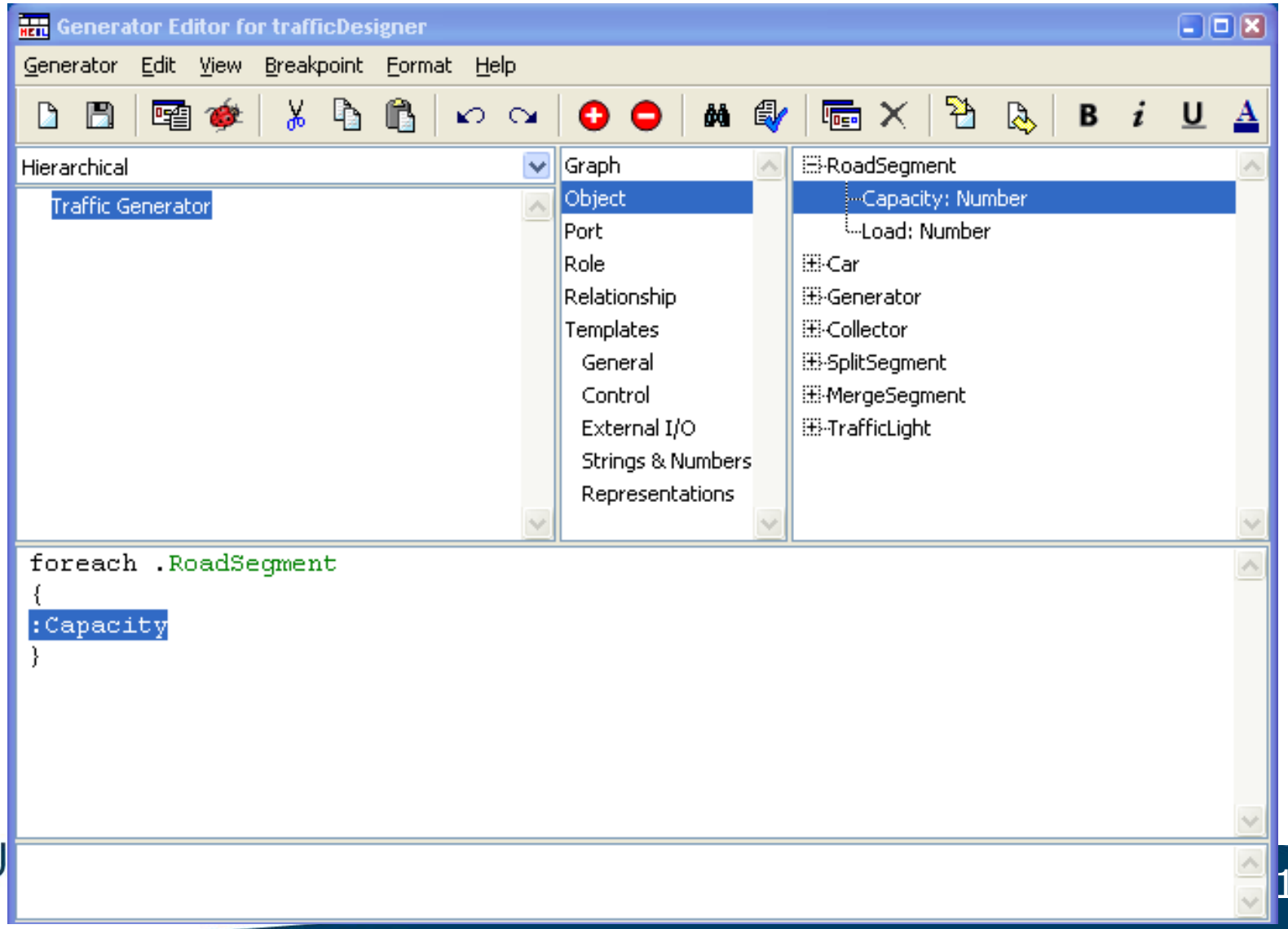
MetaEdit+ Modeler



MetaEdit+ Modeler



MetaEdit+ Workbench



Outline

- Introduction
- MetaEdit+
  - Specifications
  - Workflow
- GMF
  - Specifications <=
  - Workflow
- Comparison

GMF Specifications

- Based on Eclipse Modeling Framework (EMF) & Graphical Editing Framework (GEF)

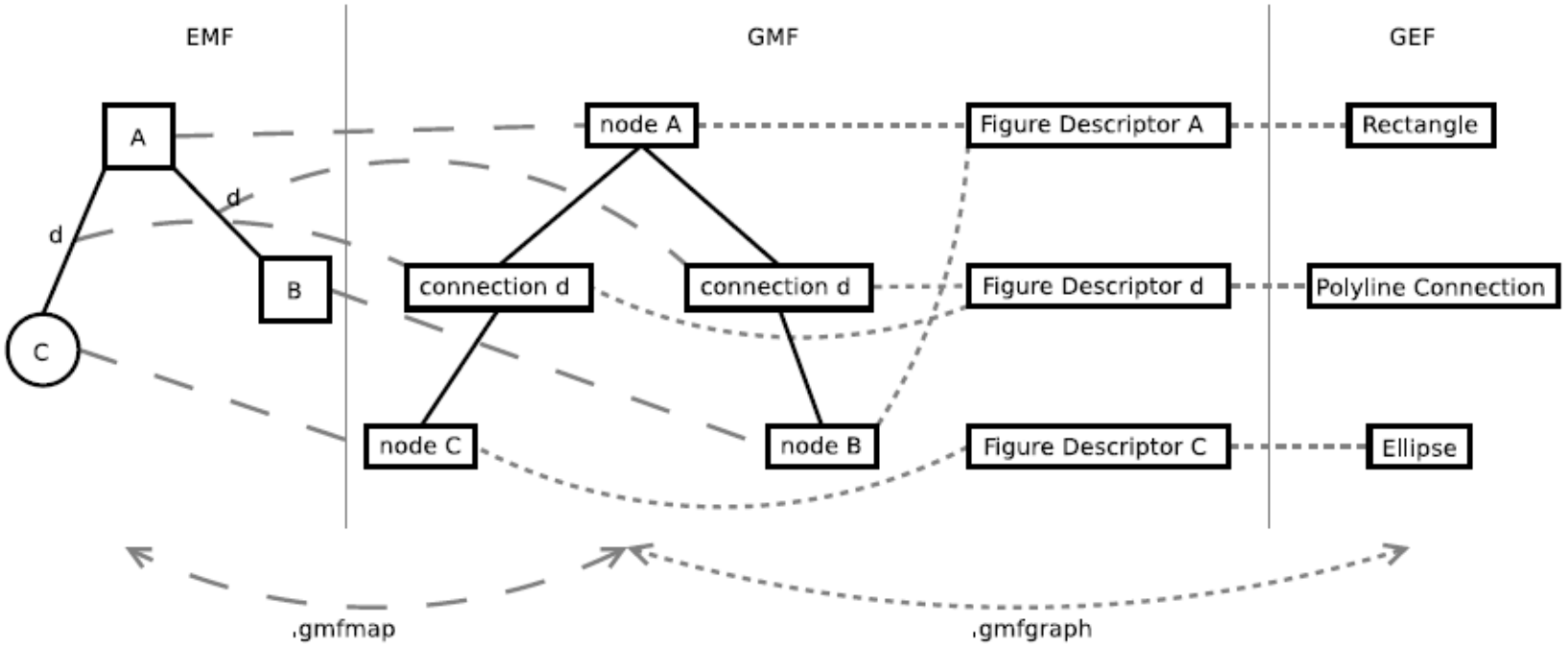
- EMF
  - Core: Ecore => XML Metadata Interchange
  - Edit: Adapter classes to view in JFace viewers
  - Codegen: Ecore to Java

- GEF
  - Rich graphical editors out of domain models
  - No restrictions on underlying model

GMF Specifications

- GMF = bridge between EMF & GEF
- No more model independency of GEF:
  - GMF only accepts EMF models
- 2 parts: extensions of EMF & GEF
  - Runtime environment
  - Generation framework

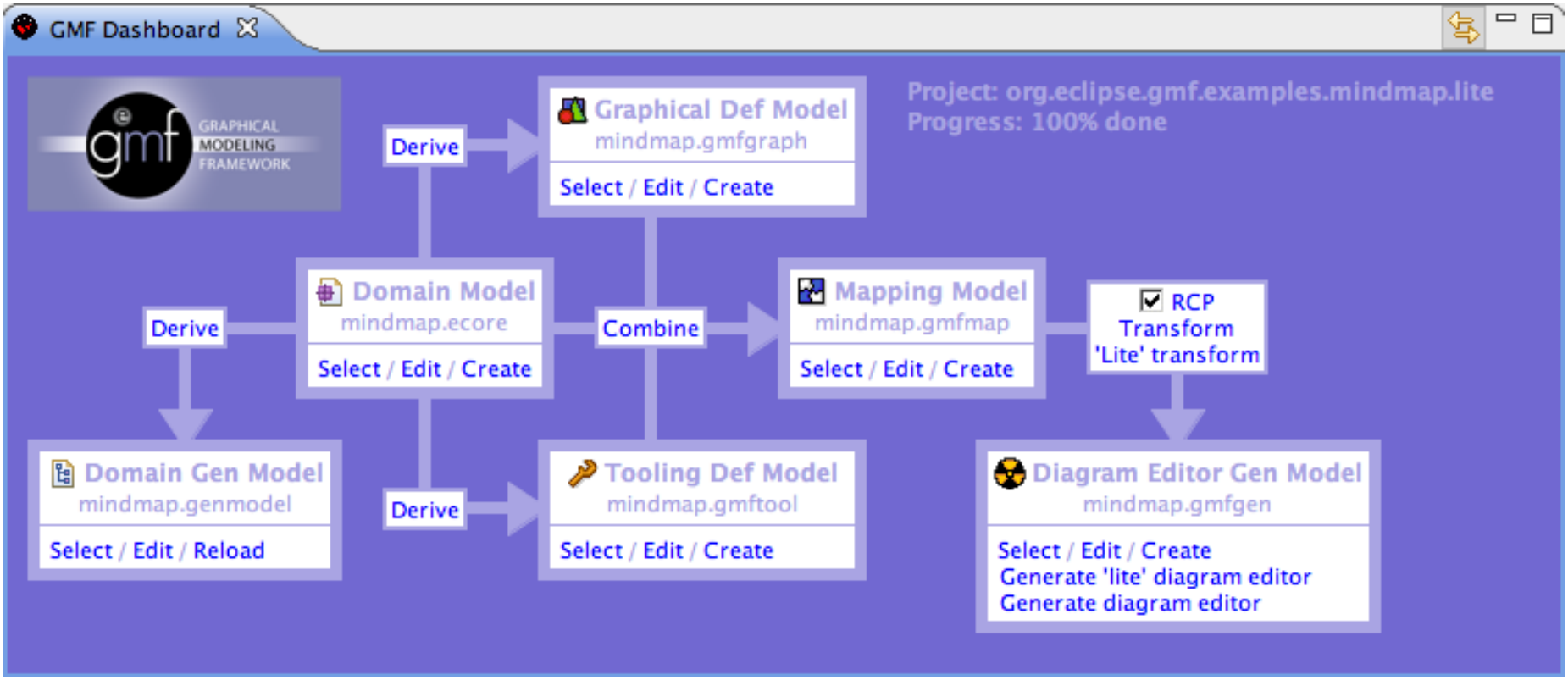
GMF Specifications



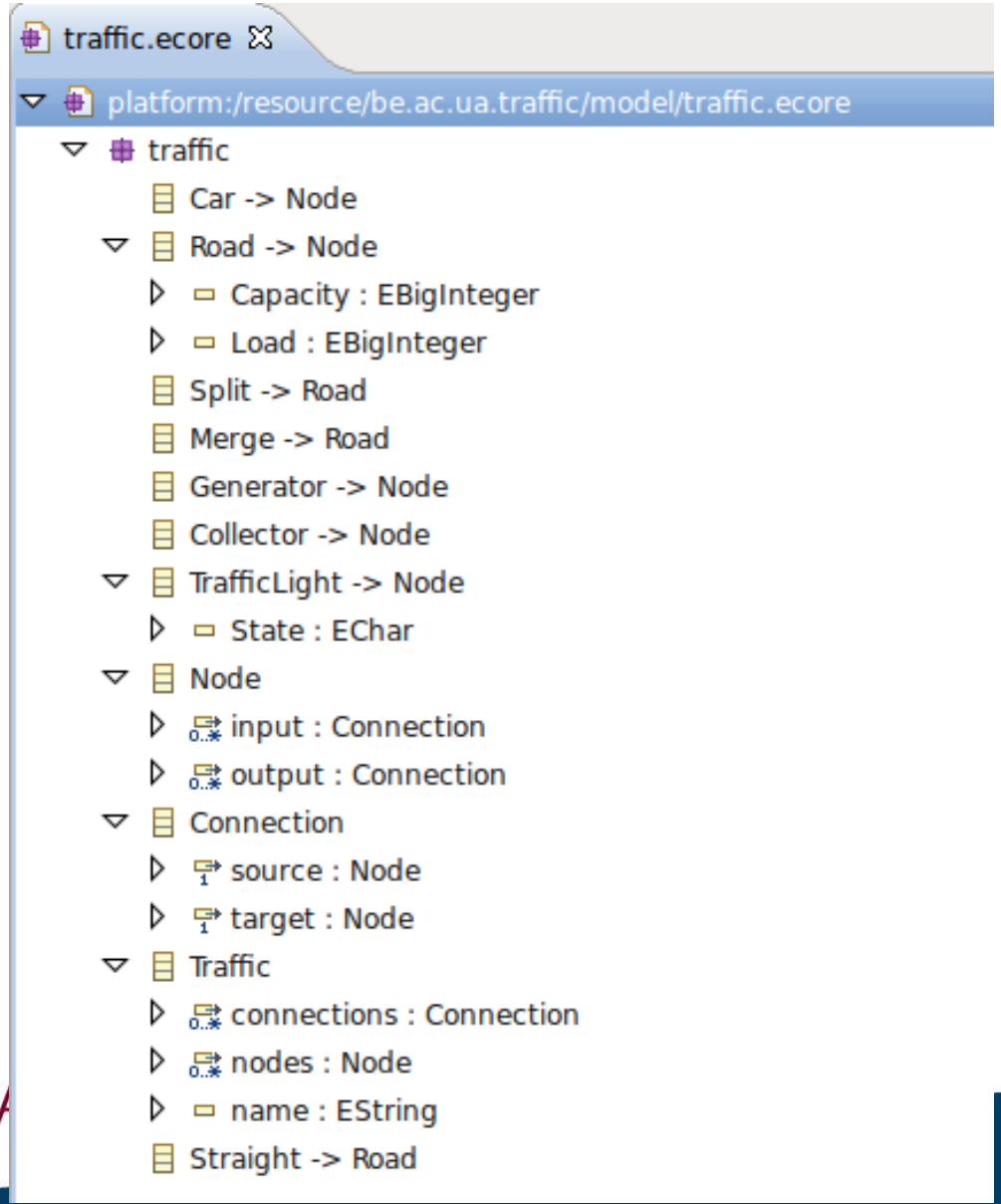
Outline

- Introduction
- MetaEdit+
  - Specifications
  - Workflow
- GMF
  - Specifications
  - Workflow <=
- Comparison

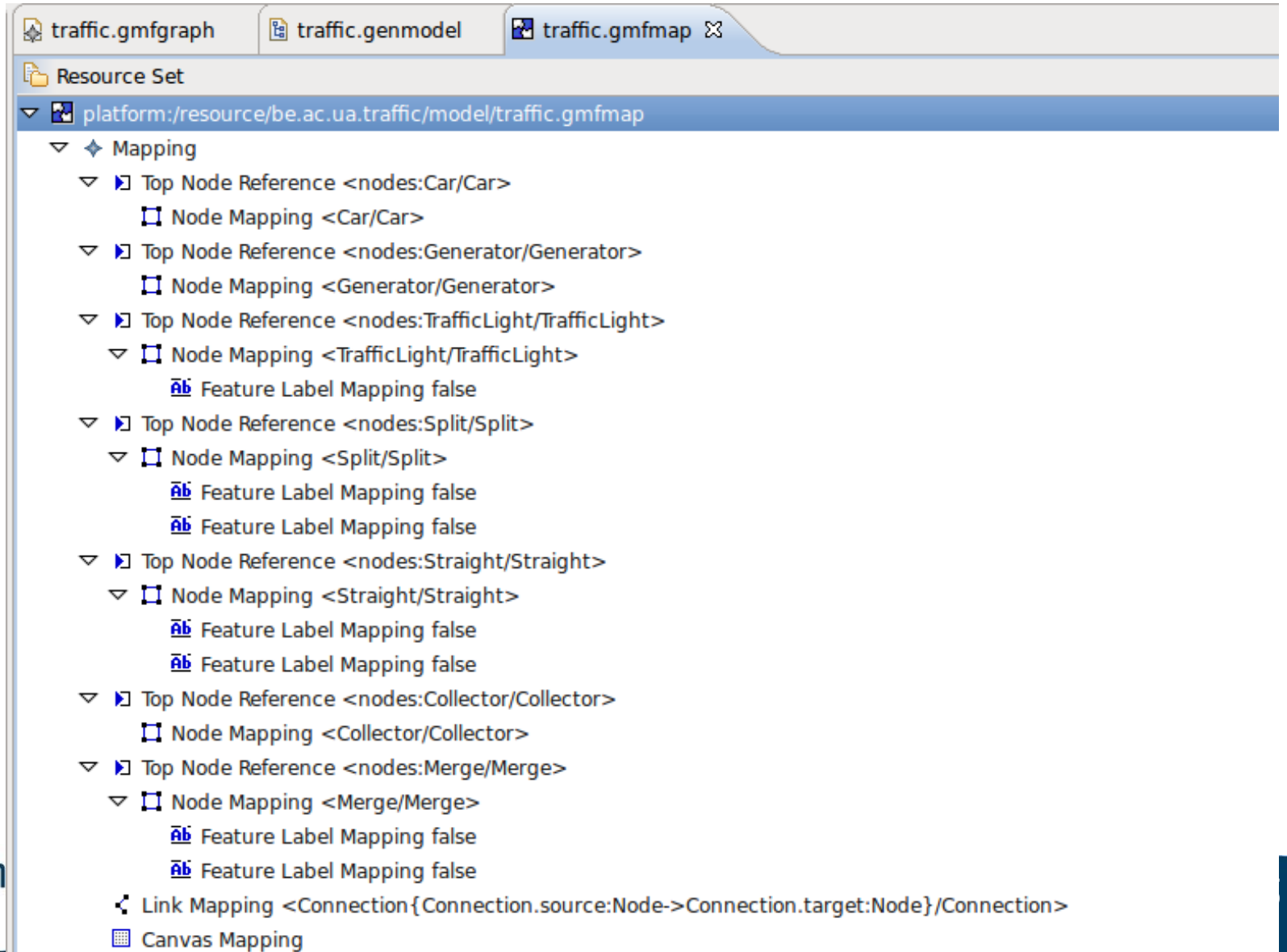
GMF Workflow



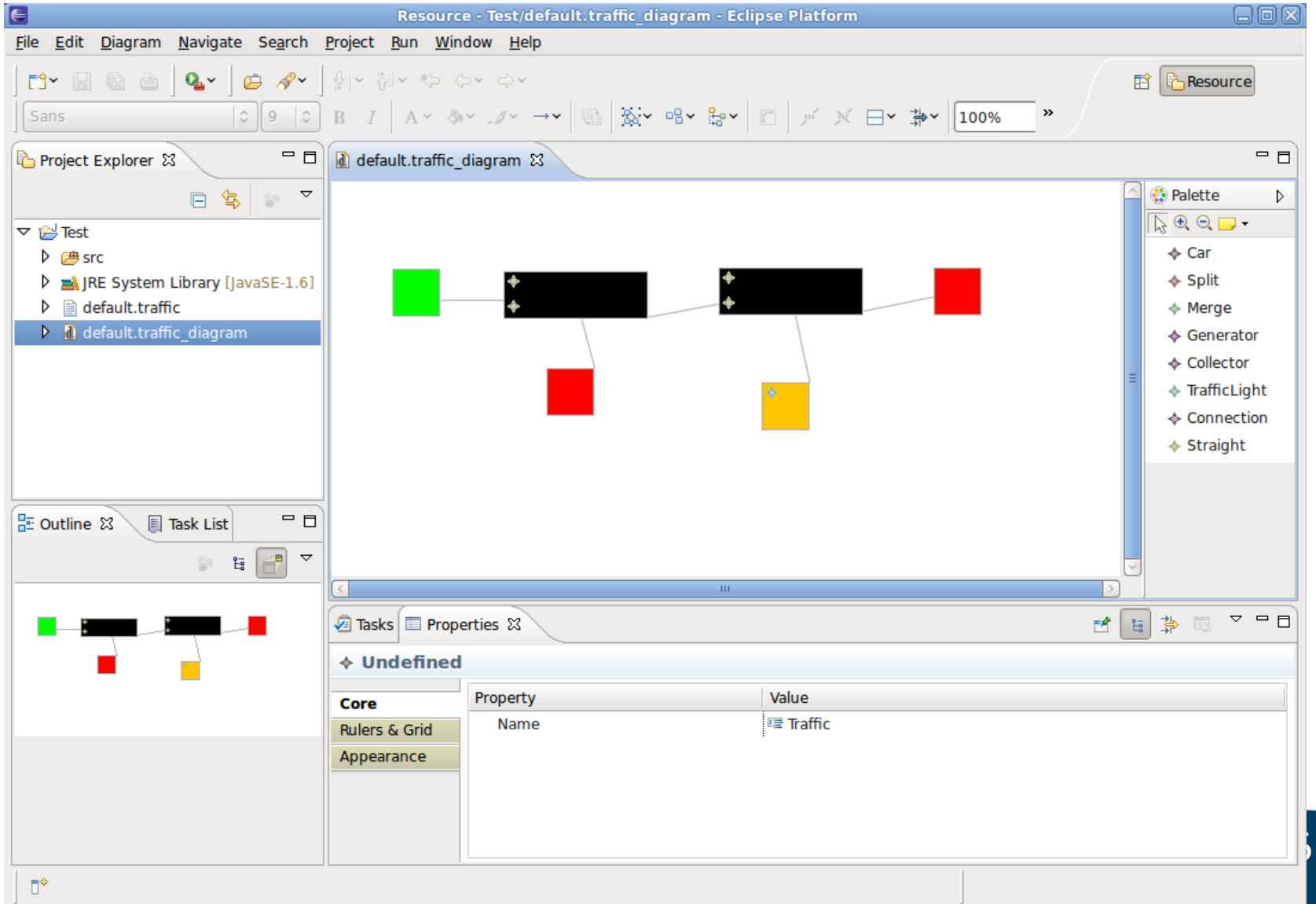
GMF Workflow



GMF Workflow

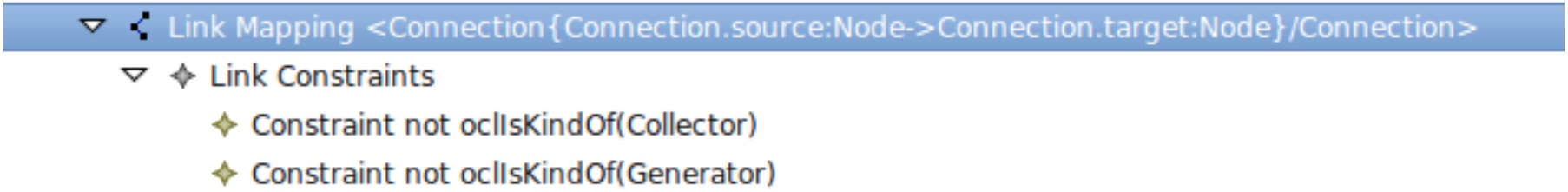


GMF Workflow



GMF Workflow

- Constraints:
  - Object Constraint Language
  - Language to define constraints on meta-models
  - Use in mapping



Outline

- Introduction
- MetaEdit+
  - Specifications
  - Workflow
- GMF
  - Specifications
  - Workflow
- Comparison <=

Comparison

Feature	Atom3	MetaEdit+	GMF
Multi-user			
Multi-view			
Update Cycle			
Live Updating			
GraphGrammar			
Build Models			
Rules			
Simulation			
Code gen			
Symbol Editor			
User-friendly			

Multi-View

- Different way's to look at the same (meta-) model

- MetaEdit+
  - Yes: diagram, matrix and text
- GMF
  - No: only tree representation
- Atom3
  - Possible

UpdateCycle

- Time to update the model when meta-model is changed
- Consistency Model
- MetaEdit+ < Atom3 < GMF

LiveUpdating

- Meta-model changes are propagated to model without restarting the tool / reopening the model

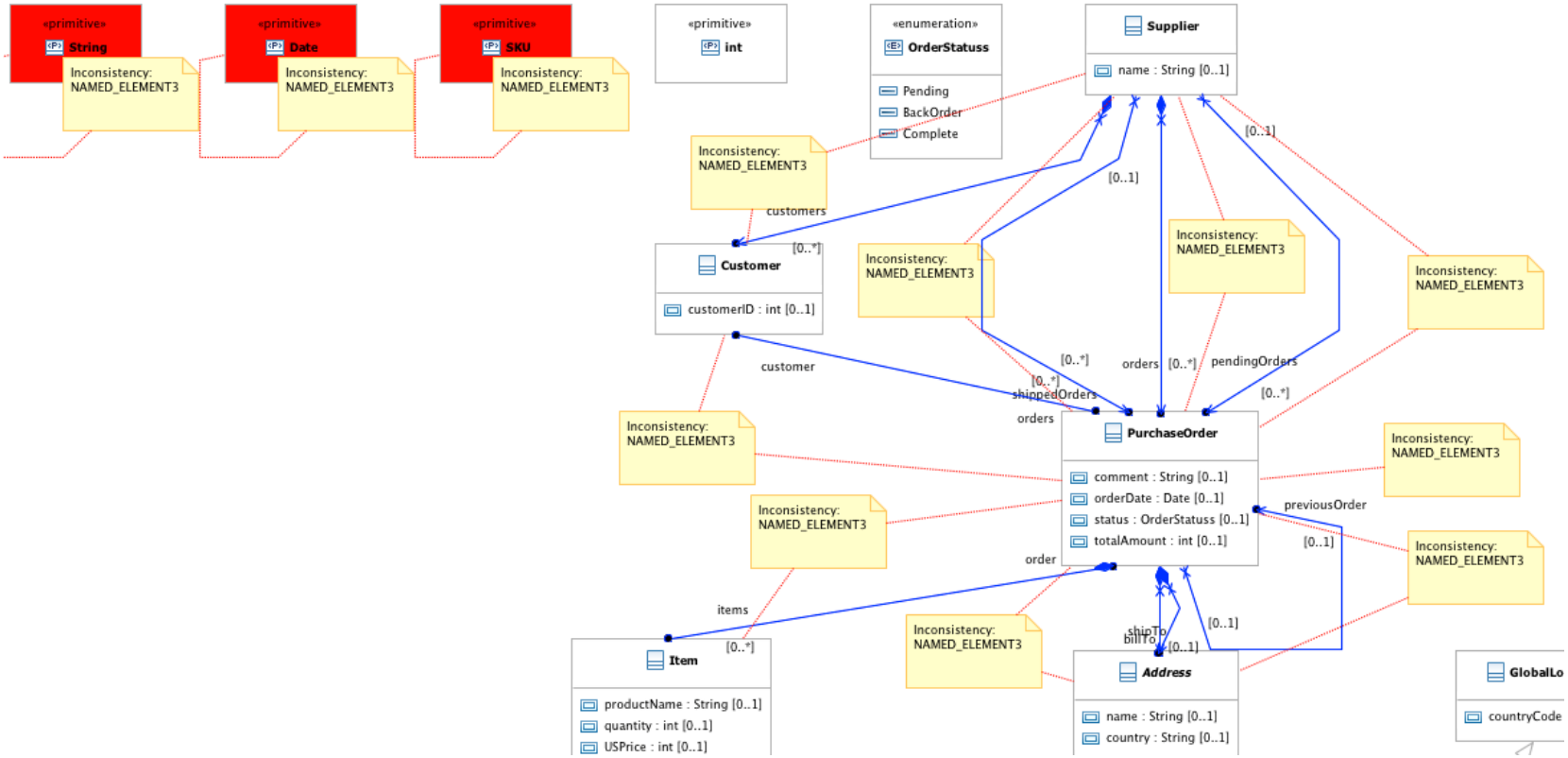
- Atom3
  - Need to reopen the model
- MetaEdit+
  - Yes
- GMF
  - Regenerate entire plug-in
  - Sometimes model is corrupted

GraphGrammar

- Is it possible to define a graph grammar?

- Atom3
  - Yes
- MetaEdit+
  - ??
- GMF
  - Yes, but some development should be done.
  - Associate a builder with the project

GMF


«primitive»
String
Inconsistency:
NAMED_ELEMENT3
«primitive»
Date
Inconsistency:
NAMED_ELEMENT3
«primitive»
SKU
Inconsistency:
NAMED_ELEMENT3
«primitive»
int
«enumeration»
OrderStatuss
Pending
BackOrder
Complete
Supplier
name : String [0..1]
Inconsistency:
NAMED_ELEMENT3
customers
[0..*]
Customer
customerID : int [0..1]
Inconsistency:
NAMED_ELEMENT3
Inconsistency:
NAMED_ELEMENT3
Inconsistency:
NAMED_ELEMENT3
[0..1]
[0..1]
customer
[0..*]
orders
[0..*]
pendingOrders
[0..*]
shippedOrders
[0..*]
orders
Inconsistency:
NAMED_ELEMENT3
PurchaseOrder
comment : String [0..1]
orderDate : Date [0..1]
status : OrderStatuss [0..1]
totalAmount : int [0..1]
Inconsistency:
NAMED_ELEMENT3
Inconsistency:
NAMED_ELEMENT3
previousOrder
[0..1]
Inconsistency:
NAMED_ELEMENT3
order
items
[0..*]
Item
productName : String [0..1]
quantity : int [0..1]
USPrice : int [0..1]
Inconsistency:
NAMED_ELEMENT3
shipTo
billTo
[0..1]
[0..1]
Address
name : String [0..1]
country : String [0..1]
GlobalLo
countryCode

GMF

- You will need:
  - Create new kind of projects: ProjectNature
  - Create a new builder to build the diagram

- Like in Java, the diagram will be updated everytime you save.

Build models

- Can we use the same tool to build models and meta-models?

- Atom3 / MetaEdit+
  - Yes
- GMF
  - Build meta-models in Eclipse + GMF
  - Generate new plug-in
  - Build models in Eclipse + Plug-in

Simulation

- MetaEdit+ and Atom3
  - Yes, program through API
  - Changes are reflected live in the model
- Atom3
  - Offers debug window
- GMF
  - Possible, needs some coding
  - Models can not be accessed directly

Transformation Rules

- Atom3
  - Yes, even visual
- MetaEdit+ & GMF
  - Possible, but needs coding
  - Through API, develop class for each rule
  - Not visual

User Friendly

- Subjective

- MetaEdit+
  - Different tools are sometimes confusing
  - Information is spread
- GMF
  - Many wizards are provided
  - Not well documented
- Atom3
  - Many control combinations

Comparison

Feature	Atom3	MetaEdit+	GMF
Multi-user			
Multi-view			
Update Cycle	2	1	3
Live Updating			
GraphGrammar			
Build Models			
Rules			
Simulation			
Code gen			
Symbol Editor			
User-friendly			
	6,5/11	8,5/11	4/11

Conclusion

- Industrial Environments:
  - Stability
  - Features need to work *out of the box*
  - MetaEdit+
- Research Environments:
  - Preferably no licenses
  - Make choice based on goals and habits