



Comparing graphical DSL editors

AToM³ vs GMF & MetaEdit+

Nick Baetens



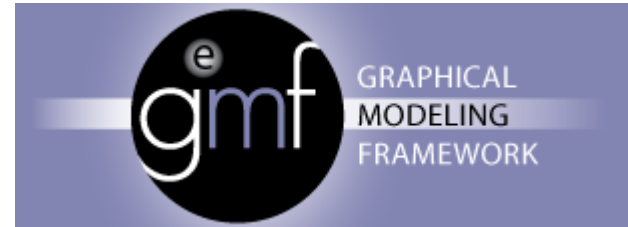
Outline

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



- Commercial
- Written in Smalltalk
- Standalone

Introduction



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



Outline

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

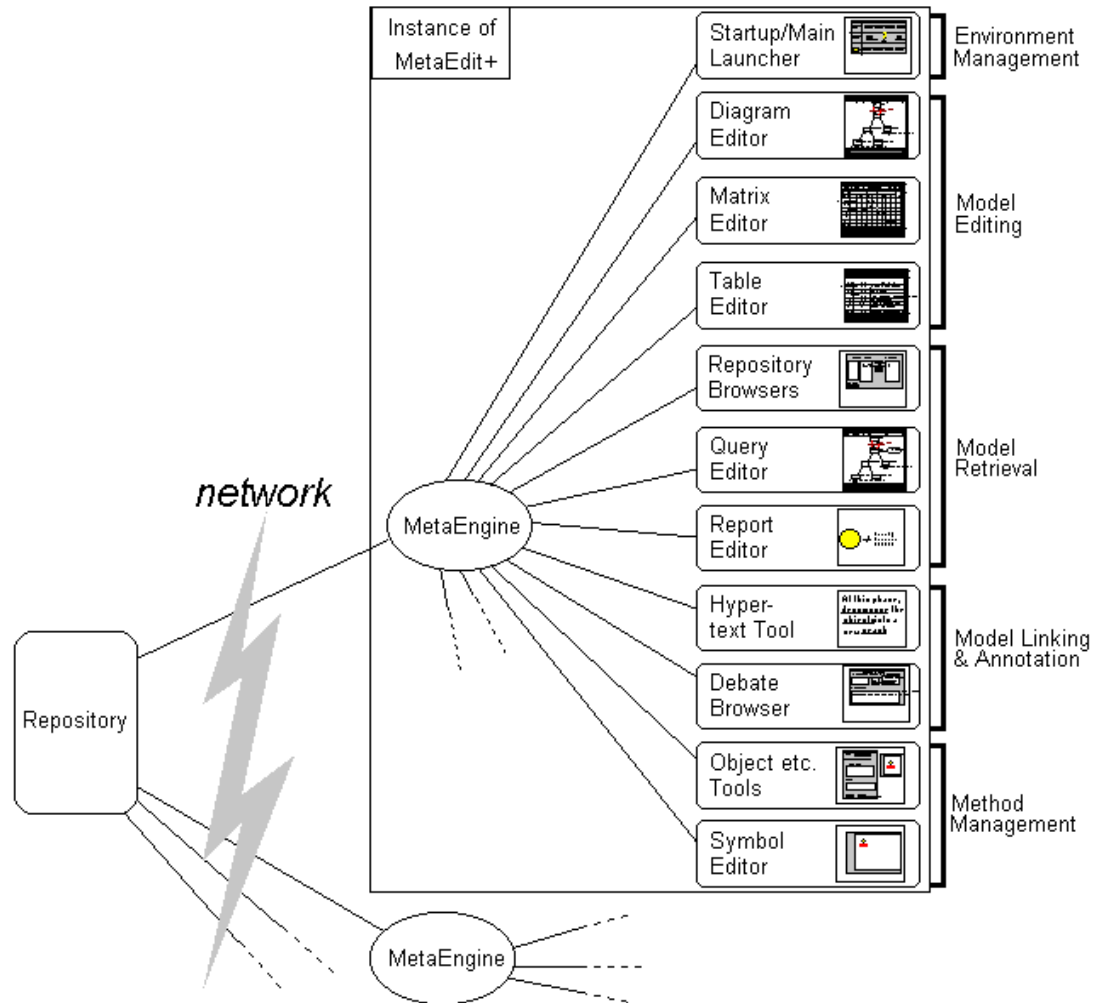


MetaEdit+

- 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+





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.

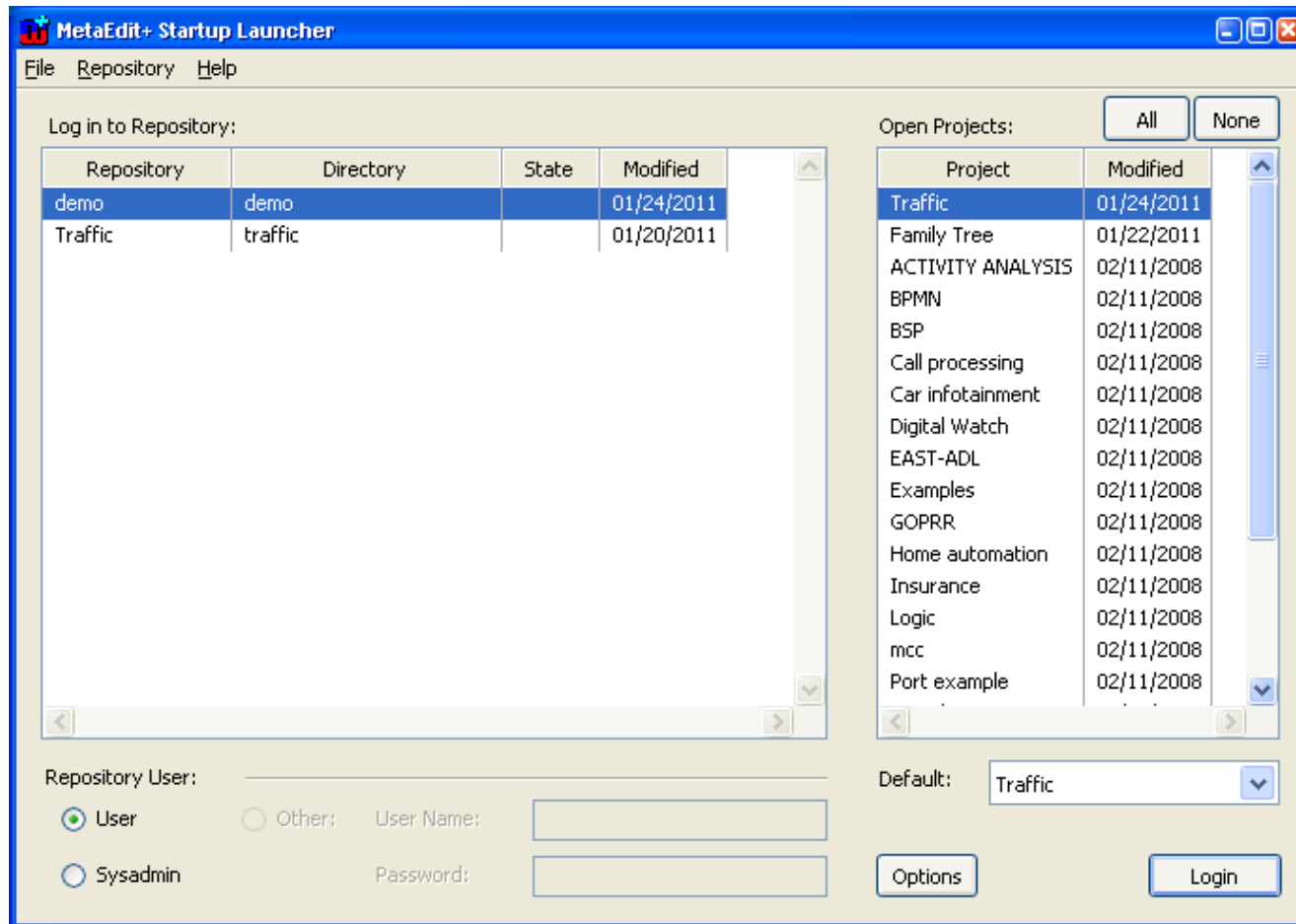


Outline

- Introduction
- MetaEdit+
 - Specifications
 - **Workflow** \leq
- GMF
 - Specifications
 - Workflow
- Comparison



MetaEdit+ Workbench





MetaEdit+ Workbench

MetaEdit+

Repository Edit Browsers Metamodel Help

Graph Browser Type Browser Object Browser Metamodel Browser

Projects

Traffic

Graph types

trafficDesigner

Contents: Object types

RoadSegment

- Car
- Generator
- Collector
- SplitSegment
- MergeSegment
- TrafficLight

Default Traffic

Help

Filter:

Tree: Subtypes

Filter:

Show: Object types



MetaEdit+ Workbench

Object Tool: RoadSegment

Object Tools Help

Save and Close

Name: RoadSegment

Ancestor: Object

Project: Traffic

Properties

Local name	Property name	Data type	Unique?
Capacity	Capacity	Number	F
Load	Load	Number	F

Description

Editor - RoadSegment

View Align Help

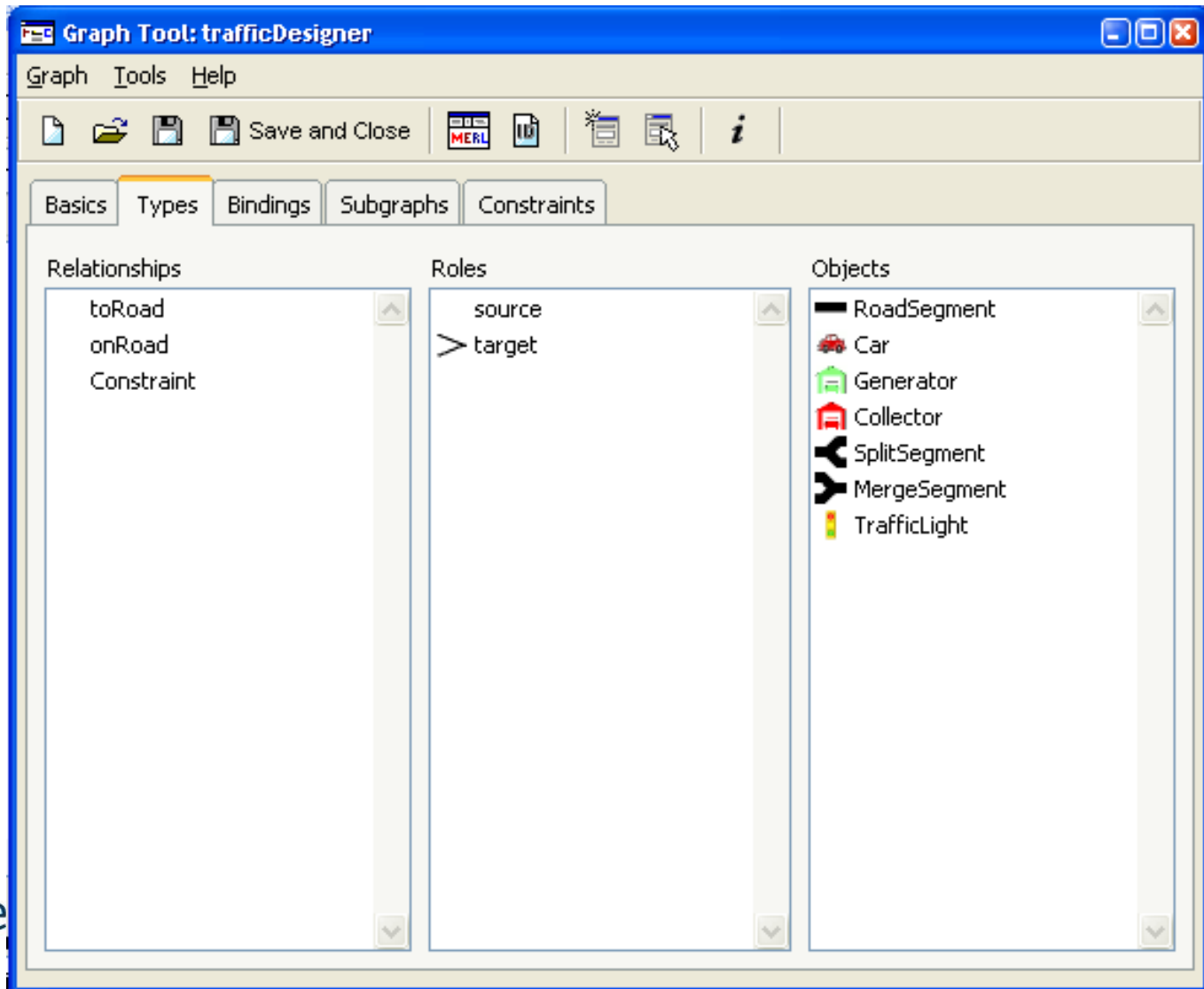
Load Cap

Color: Fill: Style: Weight:

Active: None Grid: 10@10 Snap Show 200%

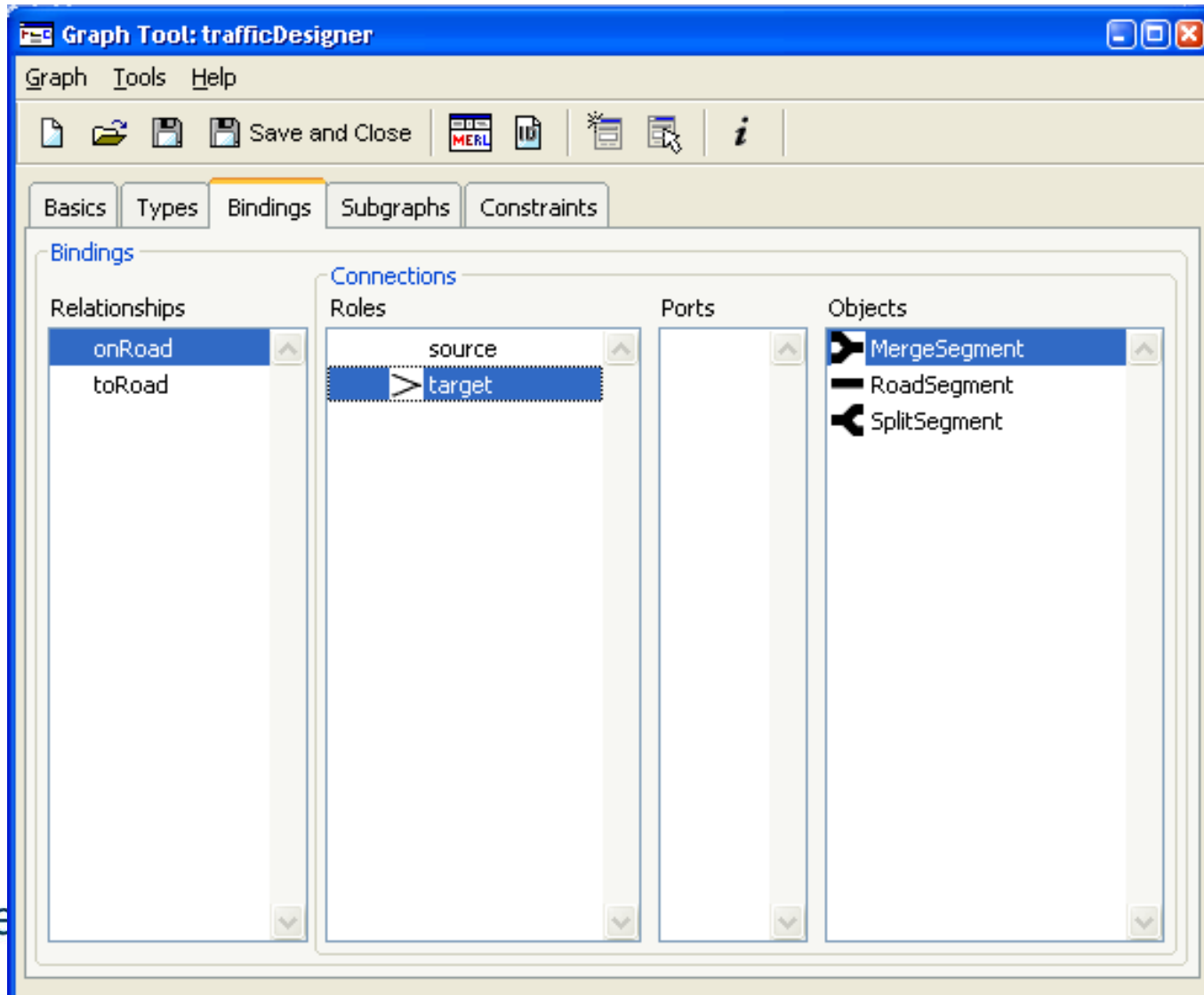


MetaEdit+ Workbench





MetaEdit+ Workbench





MetaEdit+ Workbench

The screenshot shows the MetaEdit+ Workbench interface for a graph tool named "trafficDesigner". The main window has a menu bar with "Graph", "Tools", and "Help". Below the menu bar is a toolbar with icons for file operations and a "Save and Close" button. The main area has several tabs: "Basics", "Types", "Bindings", "Subgraphs", and "Constraints". The "Constraints" tab is active, displaying a list of constraints for the graph type:

- Car may be in at most 1 onRoad relationship
- Collector may be in at most 1 toRoad relationship
- Generator may be in at most 1 toRoad relationship
- RoadSegment may be in at most 1 source role
- SplitSegment may be in at most 1 target role
- SplitSegment may be in at most 2 source roles
- TrafficLight may be in at most 1 toRoad relationship

At the bottom of the main window, there is a section labeled "Add Constraint For:" with a dropdown menu set to "Connectivity" and an "Add" button.

The "Connectivity Constraint Definer" dialog box is open, showing the configuration for a constraint. It has a title bar with a red "X" icon. The dialog contains the following fields and options:

- "Objects of type" dropdown menu: "Car" (with a car icon)
- Text: "may be in at most"
- Text input field: "1"
- Radio buttons: "Roles" (unselected) and "Relationships" (selected)
- Text: "of type"
- "onRoad" dropdown menu
- "OK" and "Cancel" buttons



MetaEdit+ Modeler

trafficDesigner: 1, January 22, 2011, 16:20

Graph Edit View Types Format Help

toRo onRo Cons

Property Value

Graph type	trafficDesigner
Load	0
RoadCapaci	1

Active: None Subgraph(s): None Grid: 10@10 Snap Show 100%



MetaEdit+ Modeler

Application: Restaurant finder, May 20, 2005, 10:50

Graph Edit View Types Format Help

Build Chec Code

Stop
Stop
Form
Text_editor
Listbox
List
Multi_query
Note
SMS message sent
ZIP must have 5 digits
Popup_menu
Restaurant type
Select restaurant
Query
Enter name

Property Value
Object type Condition
Condition variable legal_zip
Condition if len(str(Location))

Active: legal_zip: Condition Subgraph(s): None Grid: 10@10 Snap Show 100%

```
graph TD
    Start(( )) --> SelectRestaurant[Select restaurant  
by name  
by type  
exit]
    SelectRestaurant -- by type --> RestaurantType[Restaurant type  
Chinese  
Greek  
Italian]
    RestaurantType --> ProvideLocation[Provide location ZIP code]
    ProvideLocation --> Decision{legal_zip}
    Decision -- ==True --> RestType[555648606 Resttype  
+Type+, +Location]
    Decision -- ==False --> ZipError[ZIP must have 5 digits]
    RestType --> SMSSent[6 SMS message sent]
    SMSSent --> RestName[555648606 Restname  
+Name]
```

Condition: Object

Condition variable: legal_zip

Condition:

```
if len(str(Location)) == 5:
    legal_zip=True
else:
    legal_zip=False
```

OK Cancel Info...

555648606 Restname
+Name

6 SMS message sent

555648606 Resttype
+Type+, +Location

ZIP must have 5 digits



MetaEdit+ Workbench

Generator Editor for trafficDesigner

Generator Edit View Breakpoint Format Help

Hierarchical

Traffic Generator

Graph

Object

Port

Role

Relationship

Templates

General

Control

External I/O

Strings & Numbers

Representations

RoadSegment

Capacity: Number

Load: Number

Car

Generator

Collector

SplitSegment

MergeSegment

TrafficLight

```
foreach .RoadSegment
{
:Capacity
}
```



Outline

- Introduction
- MetaEdit+
 - Specifications
 - Workflow
- GMF
 - Specifications \leq
 - 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

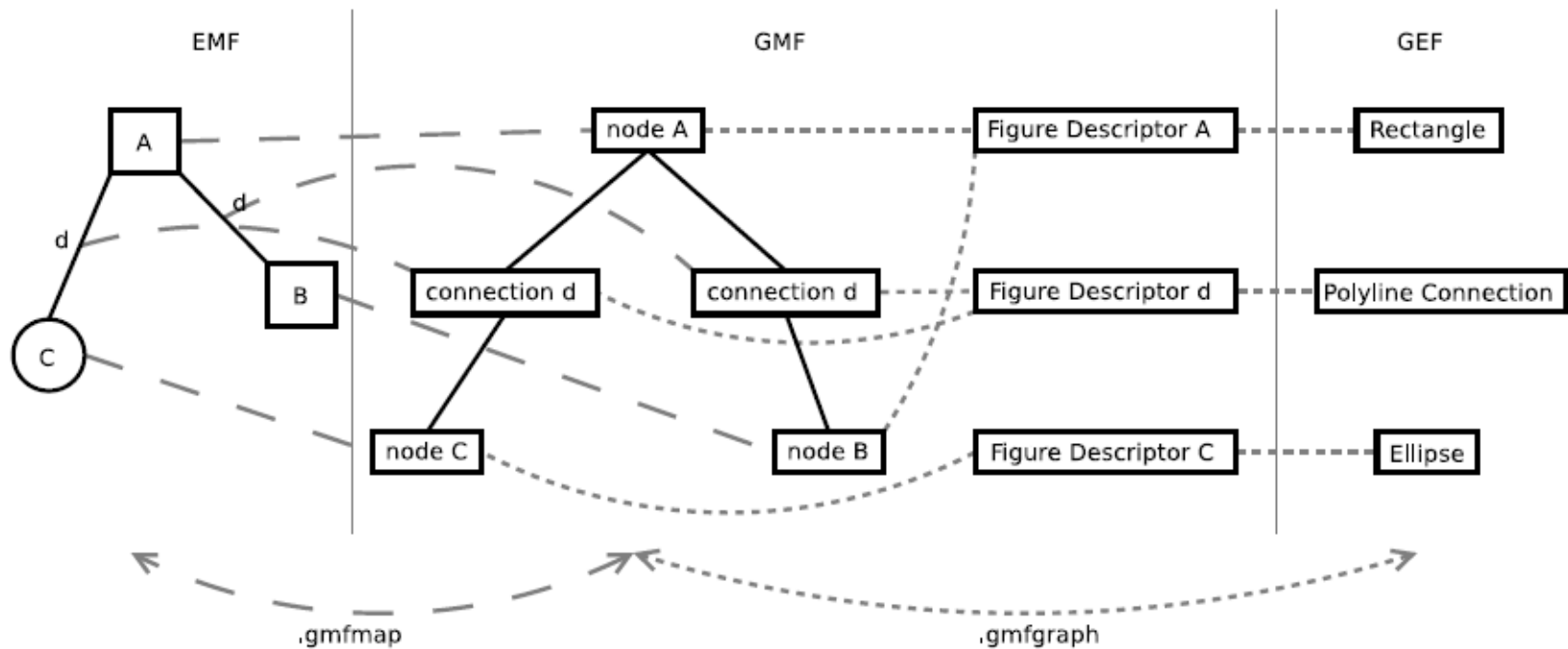


GMF Specifications

- 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



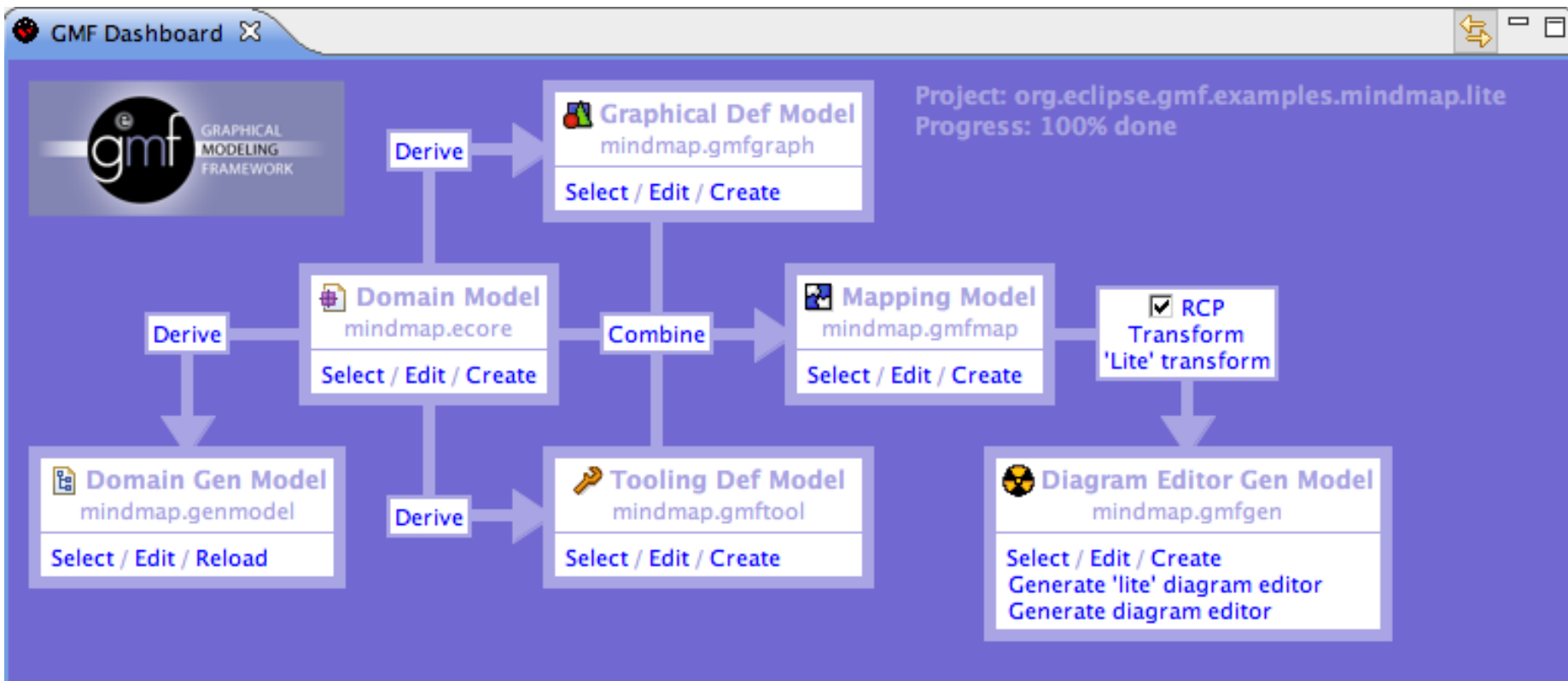


Outline

- Introduction
- MetaEdit+
 - Specifications
 - Workflow
- GMF
 - Specifications
 - **Workflow** \leq
- Comparison



GMF Workflow





GMF Workflow

The screenshot displays the Eclipse IDE interface for the traffic.ecore project. The top toolbar shows the 'Platform' icon and the file name 'traffic.ecore'. The breadcrumb path is 'platform:/resource/be.ac.ua.traffic/model/traffic.ecore'. The main workspace shows a tree view of the model structure:

- traffic
 - Car -> Node
 - Road -> Node
 - Capacity : EBigInteger
 - Load : EBigInteger
 - Split -> Road
 - Merge -> Road
 - Generator -> Node
 - Collector -> Node
 - TrafficLight -> Node
 - State : EChar
 - Node
 - input : Connection
 - output : Connection
 - Connection
 - source : Node
 - target : Node
 - Traffic
 - connections : Connection
 - nodes : Node
 - name : EString
 - Straight -> Road



GMF Workflow

The screenshot shows a GMF editor interface with three tabs: traffic.gmfgraph, traffic.genmodel, and traffic.gmfmap. The active tab is traffic.gmfmap, which displays a tree view of a Resource Set. The tree structure is as follows:

- Resource Set
 - platform:/resource/be.ac.ua.traffic/model/traffic.gmfmap
 - Mapping
 - Top Node Reference <nodes:Car/Car>
 - Node Mapping <Car/Car>
 - Top Node Reference <nodes:Generator/Generator>
 - Node Mapping <Generator/Generator>
 - Top Node Reference <nodes:TrafficLight/TrafficLight>
 - Node Mapping <TrafficLight/TrafficLight>
 - Feature Label Mapping false
 - Top Node Reference <nodes:Split/Split>
 - Node Mapping <Split/Split>
 - Feature Label Mapping false
 - Feature Label Mapping false
 - Top Node Reference <nodes:Straight/Straight>
 - Node Mapping <Straight/Straight>
 - Feature Label Mapping false
 - Feature Label Mapping false
 - Top Node Reference <nodes:Collector/Collector>
 - Node Mapping <Collector/Collector>
 - Top Node Reference <nodes:Merge/Merge>
 - Node Mapping <Merge/Merge>
 - Feature Label Mapping false
 - Feature Label Mapping false
 - Link Mapping <Connection{Connection.source:Node->Connection.target:Node}/Connection>
 - Canvas Mapping



GMF Workflow

Resource - Test/default.traffic_diagram - Eclipse Platform

File Edit Diagram Navigate Search Project Run Window Help

Sans 9 B I A ↻ ↵ ↶ ↷ 100% >>

Project Explorer

- Test
 - src
 - JRE System Library [JavaSE-1.6]
 - default.traffic
 - default.traffic_diagram

default.traffic_diagram

Palette

- Car
- Split
- Merge
- Generator
- Collector
- TrafficLight
- Connection
- Straight

Outline

Task List

Tasks Properties

◆ Undefined

Core	Property	Value
Rulers & Grid	Name	Traffic
Appearance		



GMF Workflow

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

```
▼ ↗ Link Mapping <Connection{Connection.source:Node->Connection.target:Node}/Connection>
  ▼ ✦ Link Constraints
    ✦ Constraint not oclIsKindOf(Collector)
    ✦ Constraint not oclIsKindOf(Generator)
```



Outline

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



Comparison

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



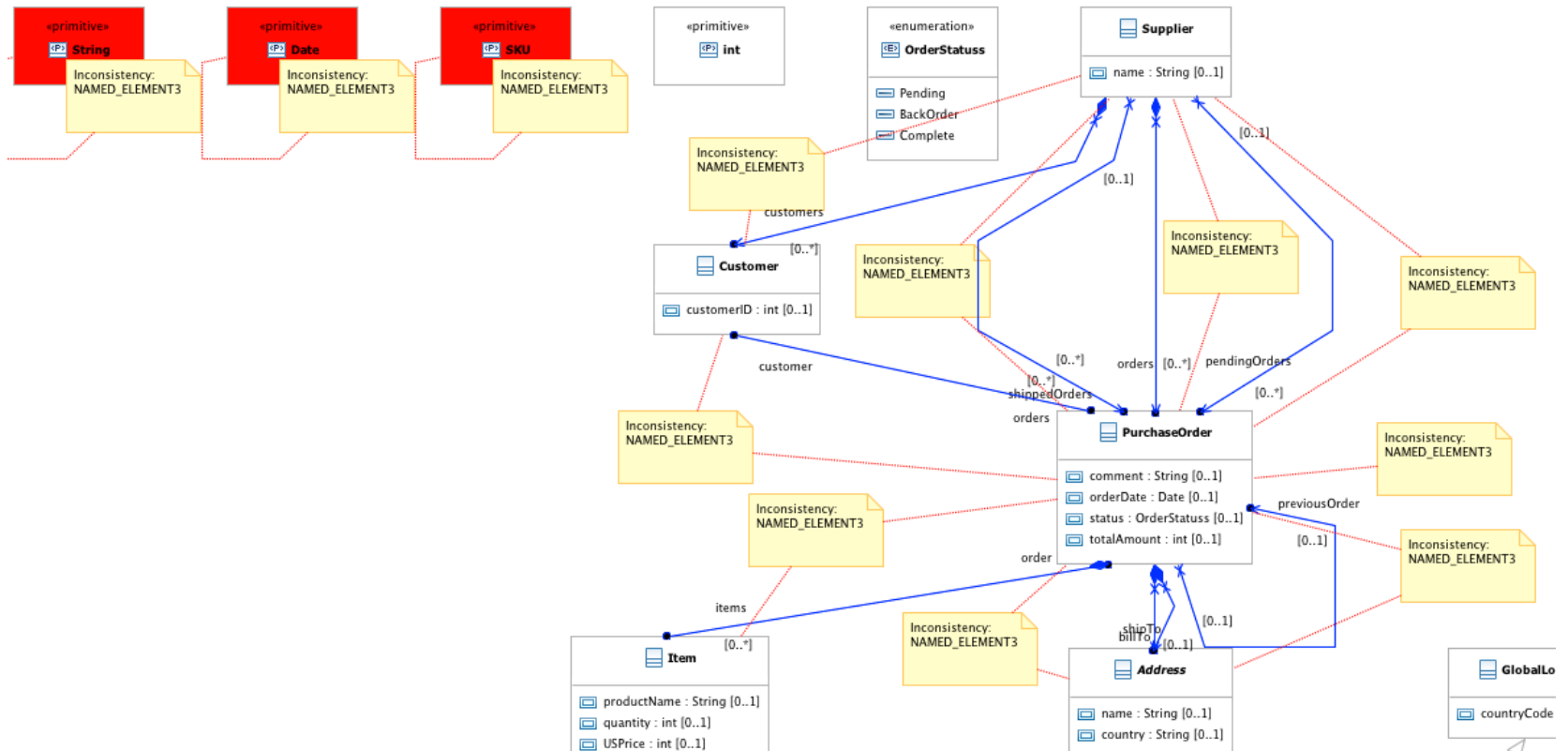
LiveUpdating

- 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





- 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			



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