UNIVERSITY OF ANTWERP

MODEL DRIVEN ENGINEERING

THE GENERIC MODELING ENVIRONMENT

STUDENT:

DANIEL DRAGOJEVIC

Index

1. Generic Modeling Environment (GME)
2. Meta Graphical Model Editor (MetaGME)
3. GME 13 with MetaGME paradigm
4. Tool Development in GME
5. Pros & Cons

Generic Modeling Environment (GME)

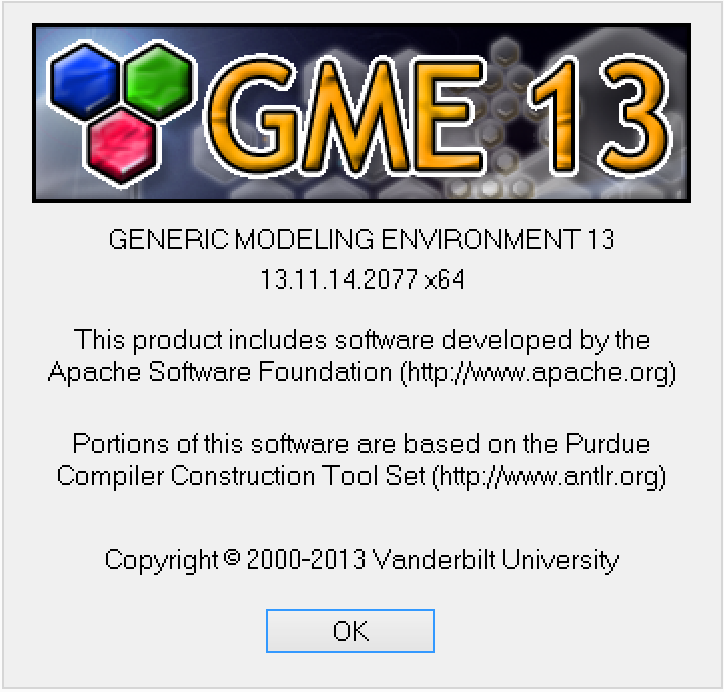
- Configurable UML toolkit that supports the creation of :
  - **domain-specific modeling**
  - **program synthesis environments**

- Institute for Software Integrated Systems, Vanderbilt University

- Free and Open-Source

- Defines modeling paradigms as metamodels

- Describes concepts and relations in UML-like class diagrams



GENERIC MODELING ENVIRONMENT 13
13.11.14.2077 x64

This product includes software developed by the Apache Software Foundation (http://www.apache.org)

Portions of this software are based on the Purdue Compiler Construction Tool Set (http://www.antlr.org)

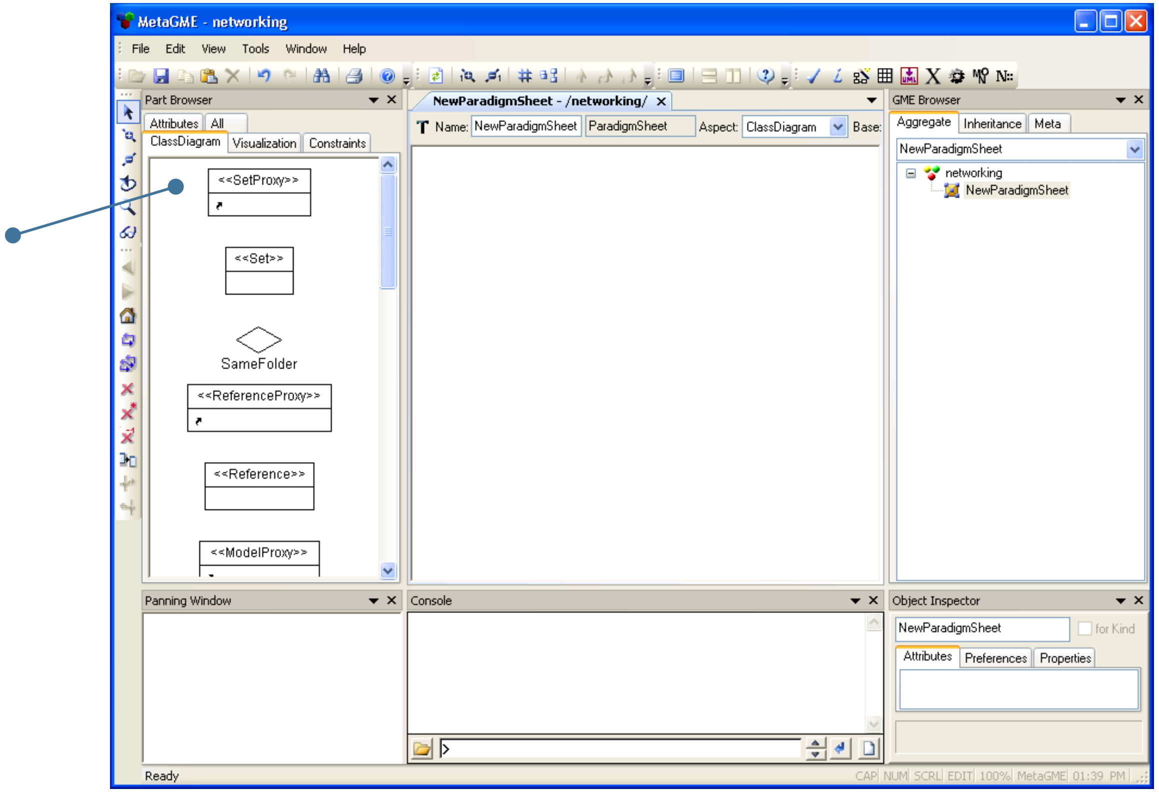
Copyright © 2000-2013 Vanderbilt University

Meta Graphical Model Editor (MetaGME)

- **Modeling paradigm for specifying metamodels**:
  - Predefined meta-metamodel and data types
  - Aspects used to control the visibility of elements
  - OCL constraints used to increase precision and validate use models in the editor

GME 13 with MetaGME paradigm

- Concepts:
  - ***Atom*** – used to represent an atomic element
  - ***Model*** – used to represent a container element
  - ***Reference*** – used as a pointer to other elements
  - ***Set*** – used to group elements
  - ***Connection*** – used to associate elements
- These elements are called **first class objects** (FCOs).
- They can contain: **Attributes** and **Constraints**



Name: NewParadigmSheet ParadigmSheet Aspect: ClassDiagram Base:

NewParadigmSheet

Ready

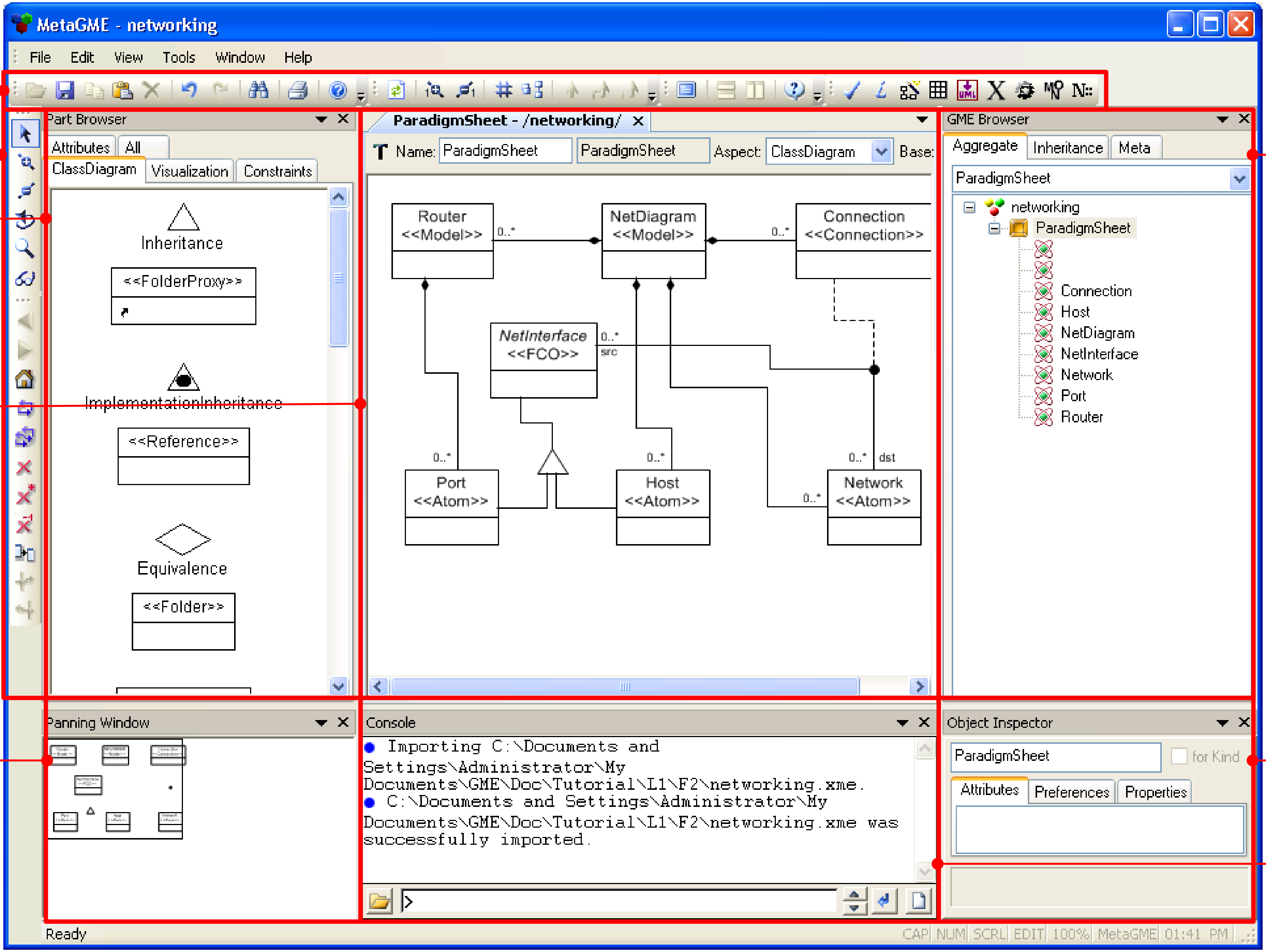
Toolbar

Modebar

Part Browser

Model Editing Window

Panning Window



Name: ParadigmSheet

Aspect: ClassDiagram

```
● Importing C:\Documents and
Settings\Administrator\My
Documents\GME\Doc\Tutorial\L1\F2\networking.xme.
● C:\Documents and Settings\Administrator\My
Documents\GME\Doc\Tutorial\L1\F2\networking.xme was
successfully imported.
```

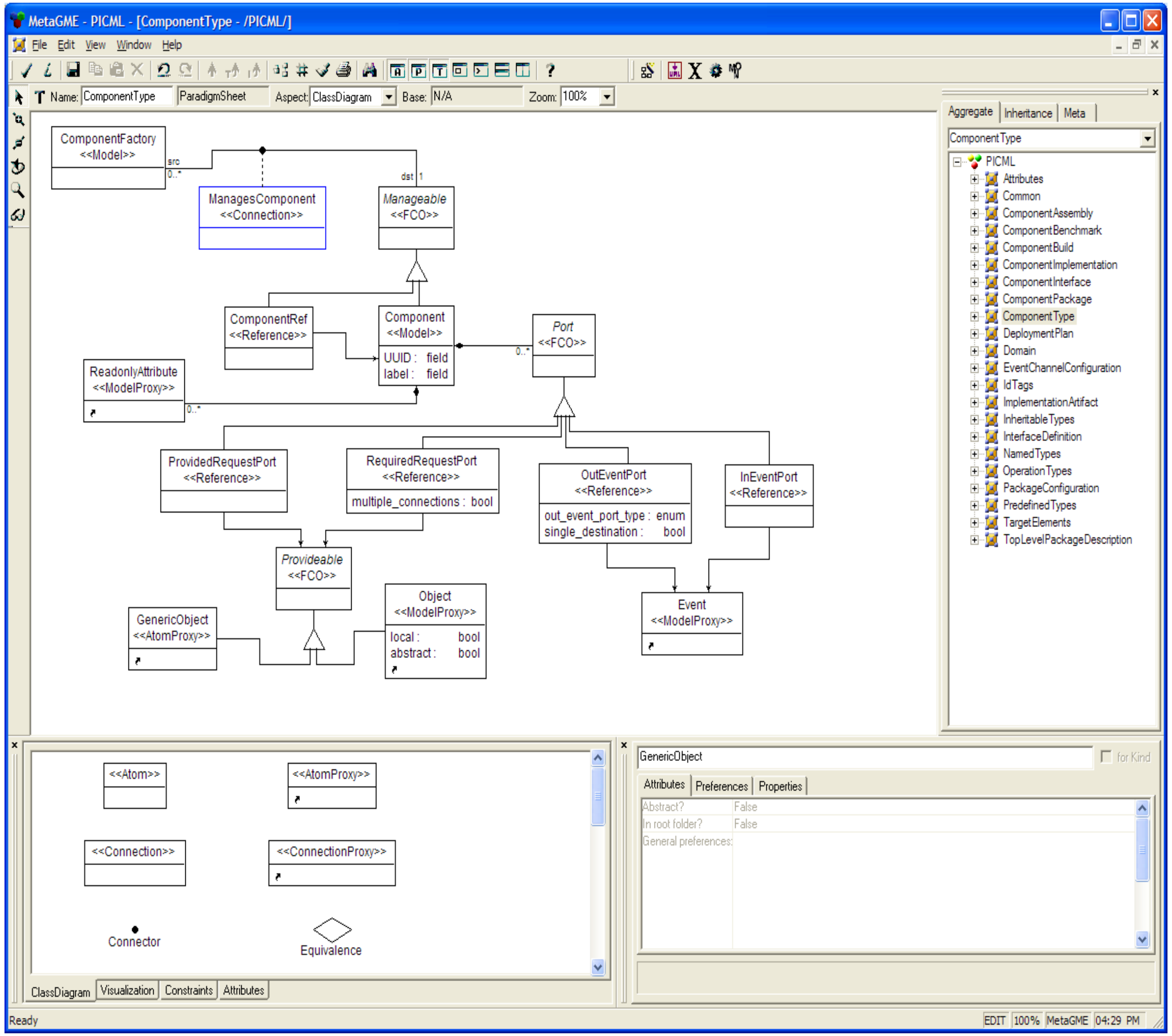
Model Browser

Object Inspector

Console

Tool Development in GME

- **Tool developers** use MetaGME to develop a *domain-specific graphical modeling environment:*
  - **Define syntax & visualization of the environment via *metamodeling***



MetaGME - PICML - [ComponentType - /PICML/]

Name: ComponentType ParadigmSheet Aspect: ClassDiagram Base: N/A Zoom: 100%

GenericObject

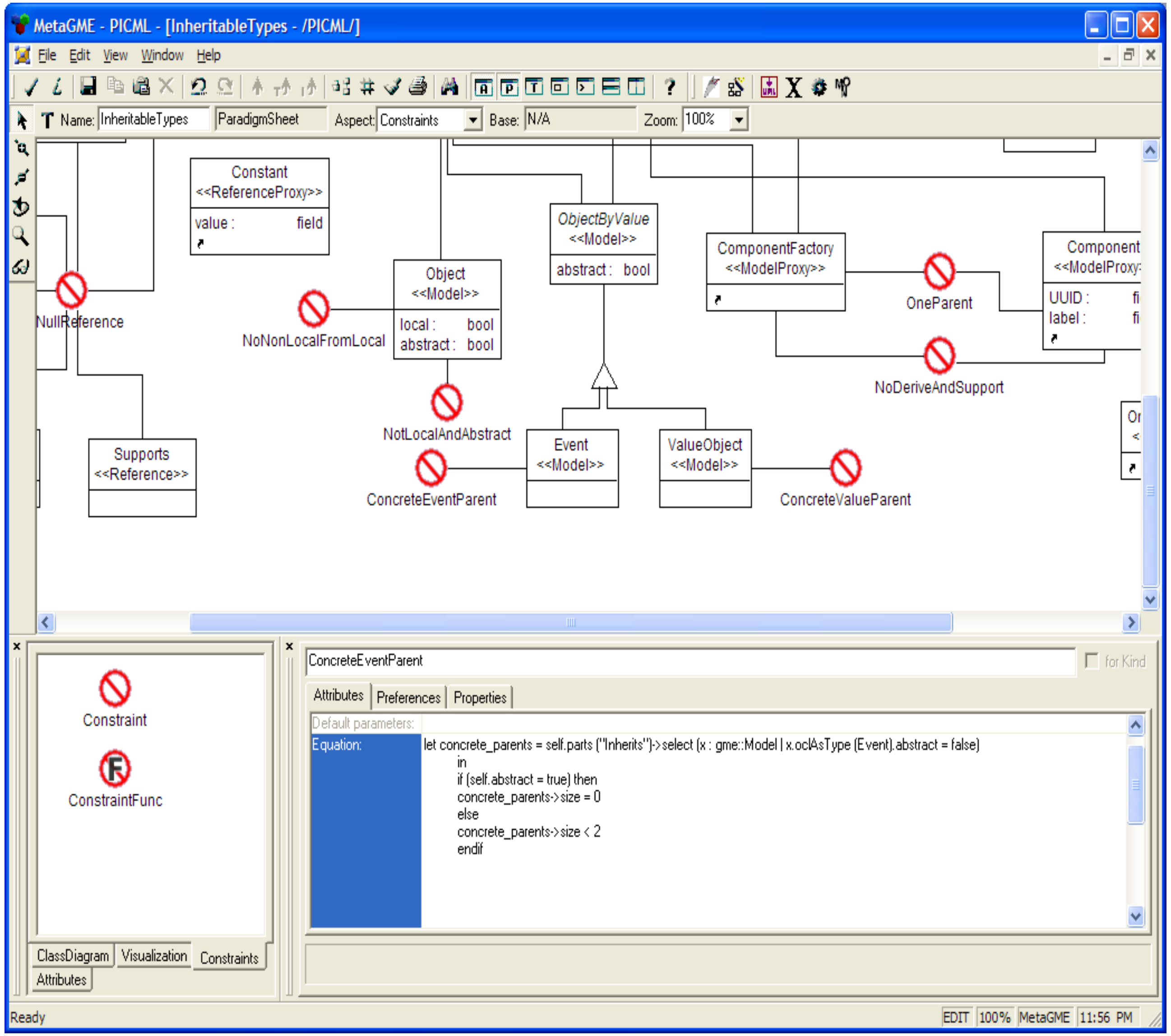
Abstract?	False
In root folder?	False
General preferences:	

Ready

EDIT 100% MetaGME 04:29 PM

Tool Development in GME

- **Tool developers** use MetaGME to develop a *domain-specific graphical modeling environment:*
  - Define syntax & visualization of the environment via *metamodeling*
  - **Define static semantics via *Object Constraint Language (OCL)***



MetaGME - PICML - [InheritableTypes - /PICML/]

Name: InheritableTypes ParadigmSheet Aspect: Constraints Base: N/A Zoom: 100%

ConcreteEventParent

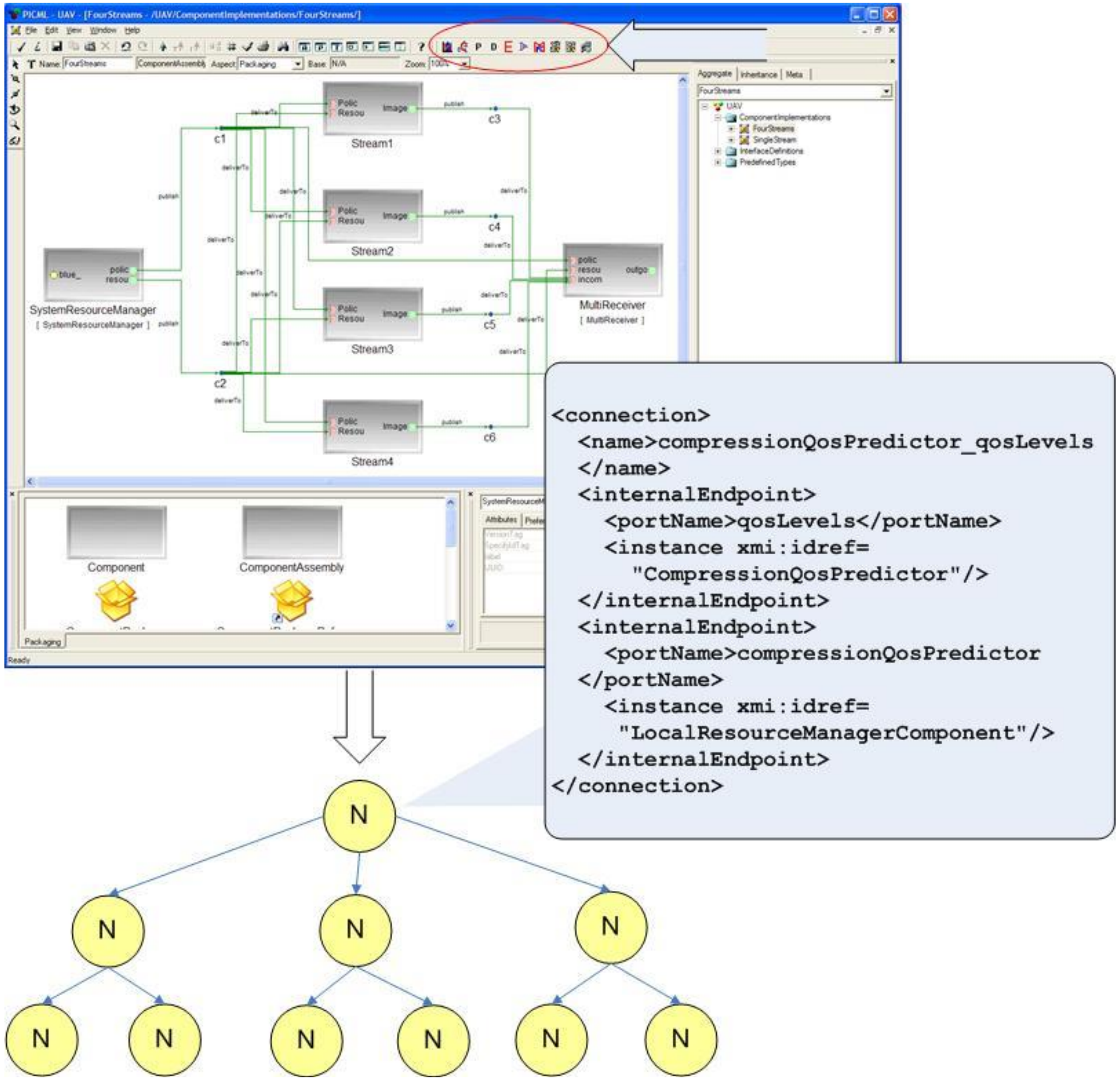
Equation:

```
let concrete_parents = self.parts ("Inherits")->select (x : gme::Model | x.oclAsType (Event).abstract = false)
        in
        if (self.abstract = true) then
        concrete_parents->size = 0
        else
        concrete_parents->size < 2
        endif
```

Ready EDIT 100% MetaGME 11:56 PM

Tool Development in GME

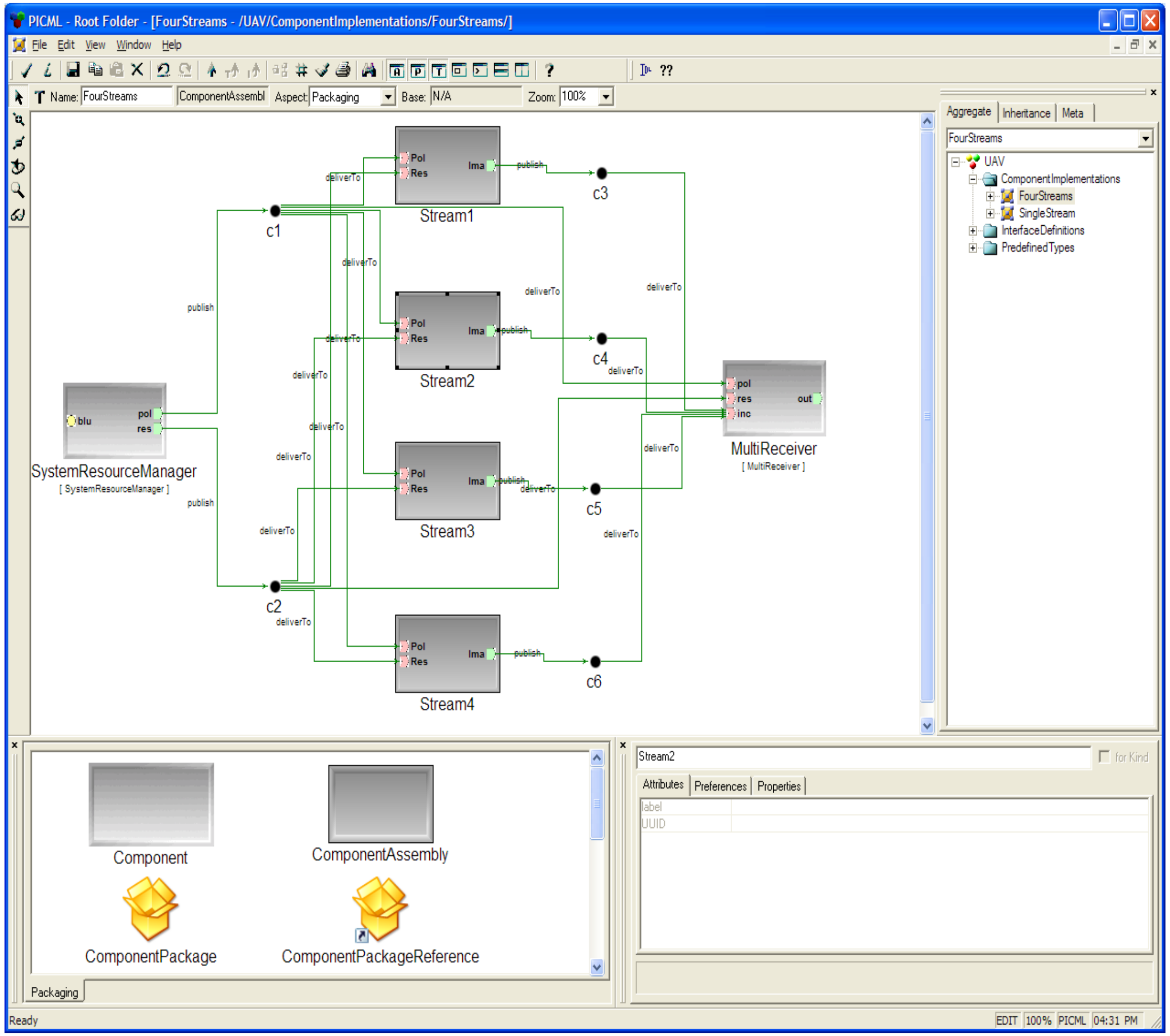
- **Tool developers** use MetaGME to develop a *domain-specific graphical modeling environment:*
  - Define syntax & visualization of the environment via *metamodeling*
  - Define static semantics via *Object Constraint Language (OCL)*
  - **Dynamic semantics implemented via *model interpreters***



```
<connection>
  <name>compressionQosPredictor_qosLevels
  </name>
  <internalEndpoint>
    <portName>qosLevels</portName>
    <instance xmi:idref=
      "CompressionQosPredictor"/>
  </internalEndpoint>
  <internalEndpoint>
    <portName>compressionQosPredictor
  </portName>
    <instance xmi:idref=
     "LocalResourceManagerComponent"/>
  </internalEndpoint>
</connection>
```

Tool Development in GME

- **Application developers** use modeling environments created with MetaGME to build applications
  - **Capture elements & dependencies visually**



PICML - Root Folder - [FourStreams - /UAV/ComponentImplementations/FourStreams/]

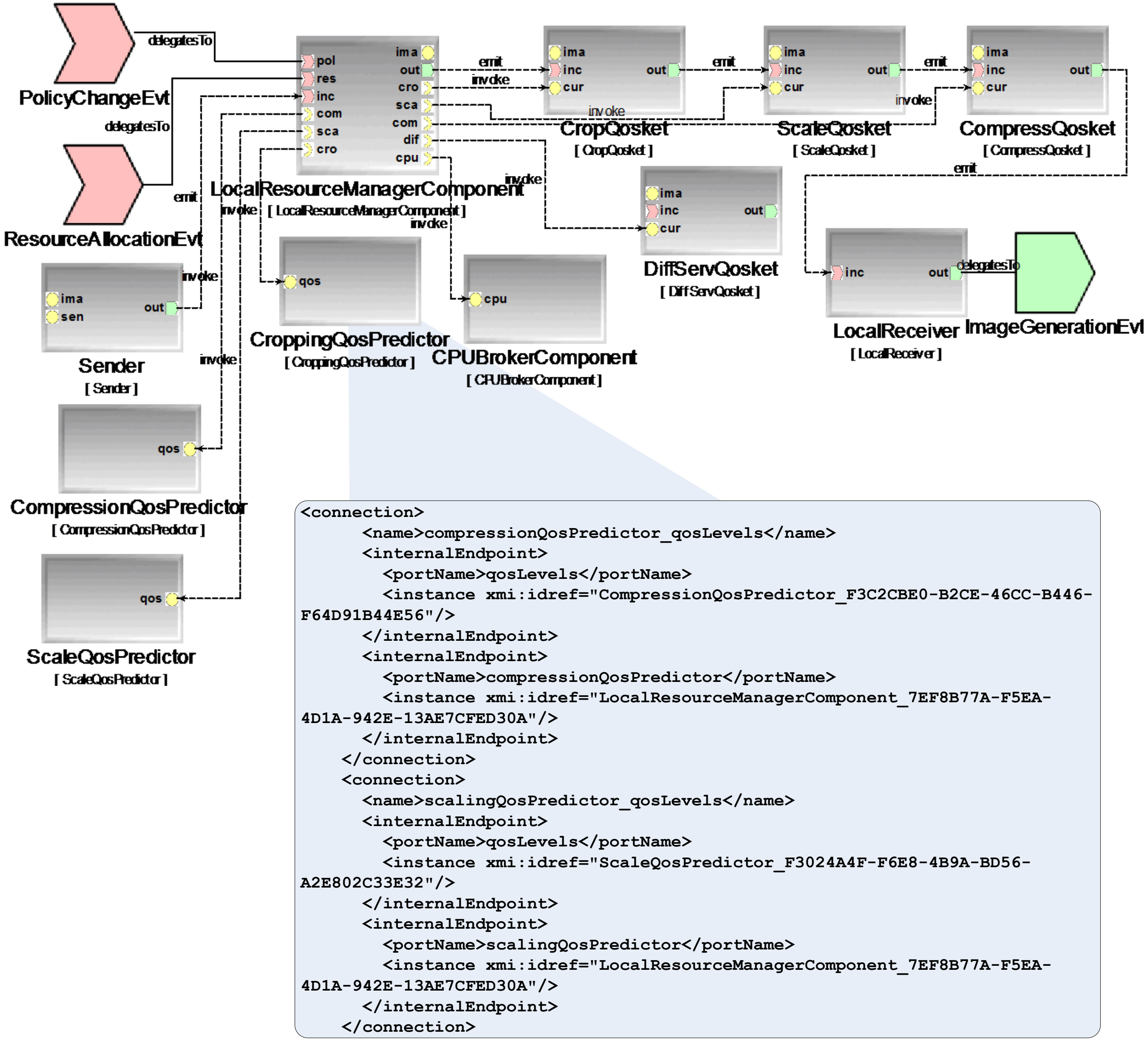
Name: FourStreams ComponentAssembl Aspect: Packaging Base: N/A Zoom: 100%

Stream2

Ready

Tool Development in GME

- **Application developers** use modeling environments created with MetaGME to build applications
  - Capture elements & dependencies visually
  - **Model interpreter produces something useful from the models**
    - e.g., 3rd generation code, simulations, deployment descriptions & configurations



```
<connection>
      <name>compressionQosPredictor_qosLevels</name>
      <internalEndpoint>
        <portName>qosLevels</portName>
        <instance xmi:idref="CompressionQosPredictor_F3C2CBE0-B2CE-46CC-B446-
F64D91B44E56"/>
      </internalEndpoint>
      <internalEndpoint>
        <portName>compressionQosPredictor</portName>
        <instance xmi:idref="LocalResourceManagerComponent_7EF8B77A-F5EA-
4D1A-942E-13AE7CFED30A"/>
      </internalEndpoint>
    </connection>
    <connection>
      <name>scalingQosPredictor_qosLevels</name>
      <internalEndpoint>
        <portName>qosLevels</portName>
        <instance xmi:idref="ScaleQosPredictor_F3024A4F-F6E8-4B9A-BD56-
A2E802C33E32"/>
      </internalEndpoint>
      <internalEndpoint>
        <portName>scalingQosPredictor</portName>
        <instance xmi:idref="LocalResourceManagerComponent_7EF8B77A-F5EA-
4D1A-942E-13AE7CFED30A"/>
      </internalEndpoint>
    </connection>
```

Pros & Cons

- Default shapes, connections, and bitmaps support
- Additional decorators (COM objects, in C++) can be programmed for specific shapes and behaviors
- Also available: higher-level C++ API (Builder Object Network)
    - simpler than COM decorators but more limited.
- Free: loading/saving (binary and XML), multiple undo/redo, drag&drop, validation against multiplicities and OCL constraints, printing, zooming, overviews, property views, etc.
- Good documentation and examples

Pros & Cons

- Evolving paradigms preserve backward compatibility
  - If elements, links and references are added but not renamed or deleted
  - More robustness for attribute handling
- Big effort required to program COM decorators
- Hard to associate decorators to links
- Analysis and transformation require programming skills (C++)
- Standalone editor, no integration with other tools

Any questions?

For further information and downloads, visit: http://www.isis.vanderbilt.edu/projects/gme

Reference

[1] An Introduction to the Generic Modeling Environment,
Z Molnar, D Balasubramanian, A Ledeczi, ISIS, Vanderbilt University

[2] The Generic Modeling Environment,
A Lédeczi, M Maróti, P Völgyesi, ISIS, Vanderbilt University

[3] Evaluation of Development Tools for Domain-Specific Modeling Languages,
D Amyot, H Farah, J F Roy, University Ottawa

[4] ISIS WP: http://www.isis.vanderbilt.edu/Projects/gme

[5] GME 13 User's Manual , ISIS, Vanderbilt University