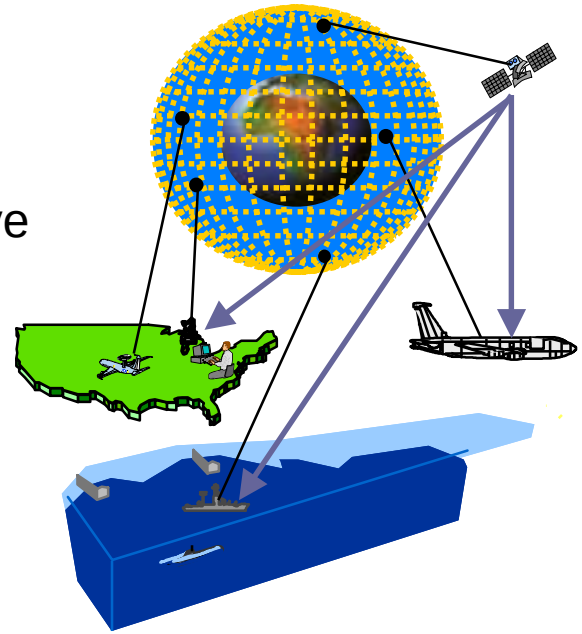
Model-Driven QoS Provisioning Techniques for CCM DRE Systems

Stoyan Paunov, Gan Deng, Douglas C. Schmidt, and Anirudha Gokhale

ISIS, Vanderbilt University

Motivation for QoS-enabled Middleware



Trends

- Software keeps getting larger, slower, & more expensive

Historical Challenges

- Building distributed systems is hard
- Building them on-time & under budget is even harder

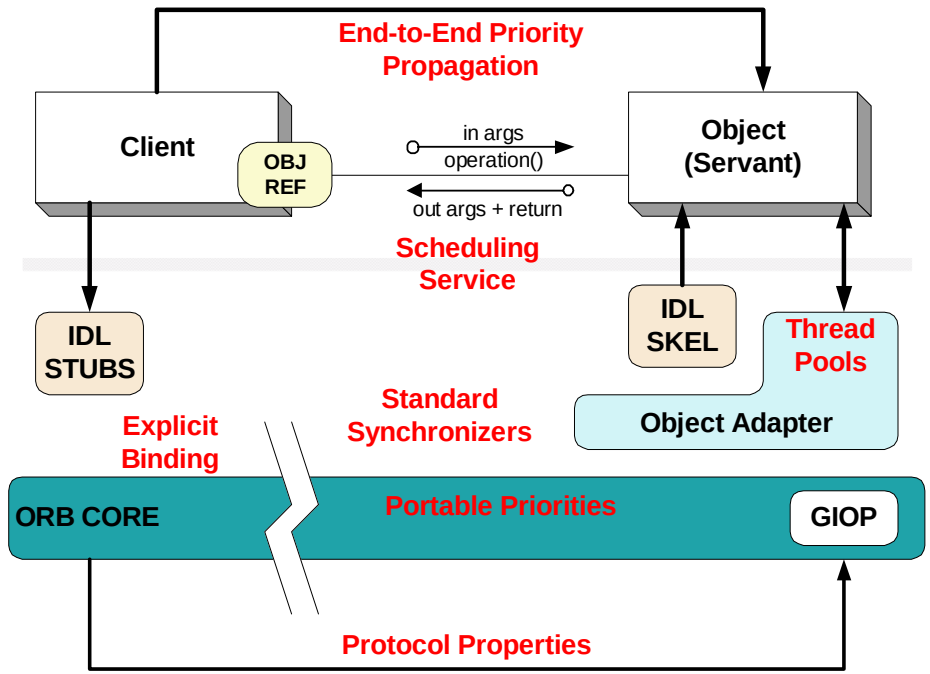
New Challenges

- Many mission-critical distributed applications require real-time QoS support - e.g., combat systems, online trading, telecom
- Building QoS-enabled applications manually is tedious, error prone, & expensive
- Conventional middleware does not support real-time QoS requirements effectively

Real-Time CORBA Policies and Mechanisms

Policies & mechanisms for resource configuration/control in RT-CORBA include:

1. **Processor Resources**
   - Thread pools
   - Priority models
   - Portable priorities
2. **Communication Resources**
   - Protocol policies
   - Explicit binding
3. **Memory Resources**
   - Request buffering



Some QoS-related Design Challenges

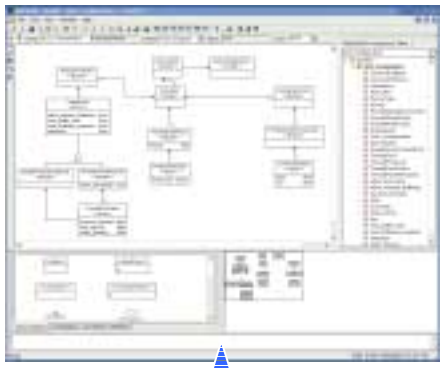
1. *Preserving priorities end-to-end*
2. *Enforcing certain priorities at the server*
3. *Supporting thread pools effectively*
4. *Buffering client requests*
5. *Synchronizing objects correctly*
6. *Controlling network & end-system resources to minimize priority inversion*

The following RT CORBA features provide the tools to overcome challenges above:

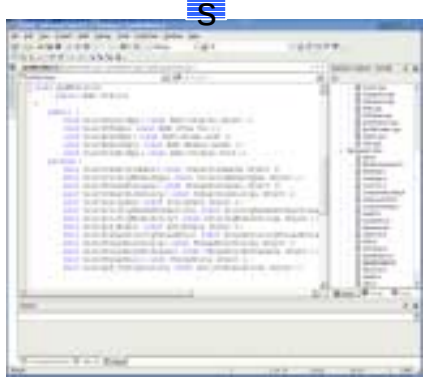
- *Thread pool policy model: thread pools with lanes vs. thread pools running at a single priority*
- *Priority propagation model: server declared vs. client propagated*
- *Connection model: a connection with logical bands for different priorities vs. the lack of such*

What is Model-Driven QoS Provisioning

META-MODEL



implements



INTERPRETER

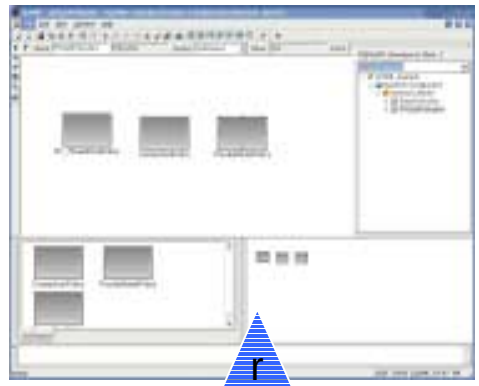
- RT CORBA compliant middleware provide implementations of the discussed features

Specifies

- However, programmatic use of these features could be tedious and error-prone
- It can however be automated via Model-Integrated Computing (MIC)
- By applying MIC we designed and implemented a graphic modeling language which can be used to synthesize XML configuration data

Provides Interpretation

- The middleware layer can then be configured automatically



reflects

MODEL INTERPRETATION

Generic Modeling Environment (GME)

"Write Code That Writes Code That Writes Code!"

GME Architecture

Application Developers (Modelers)

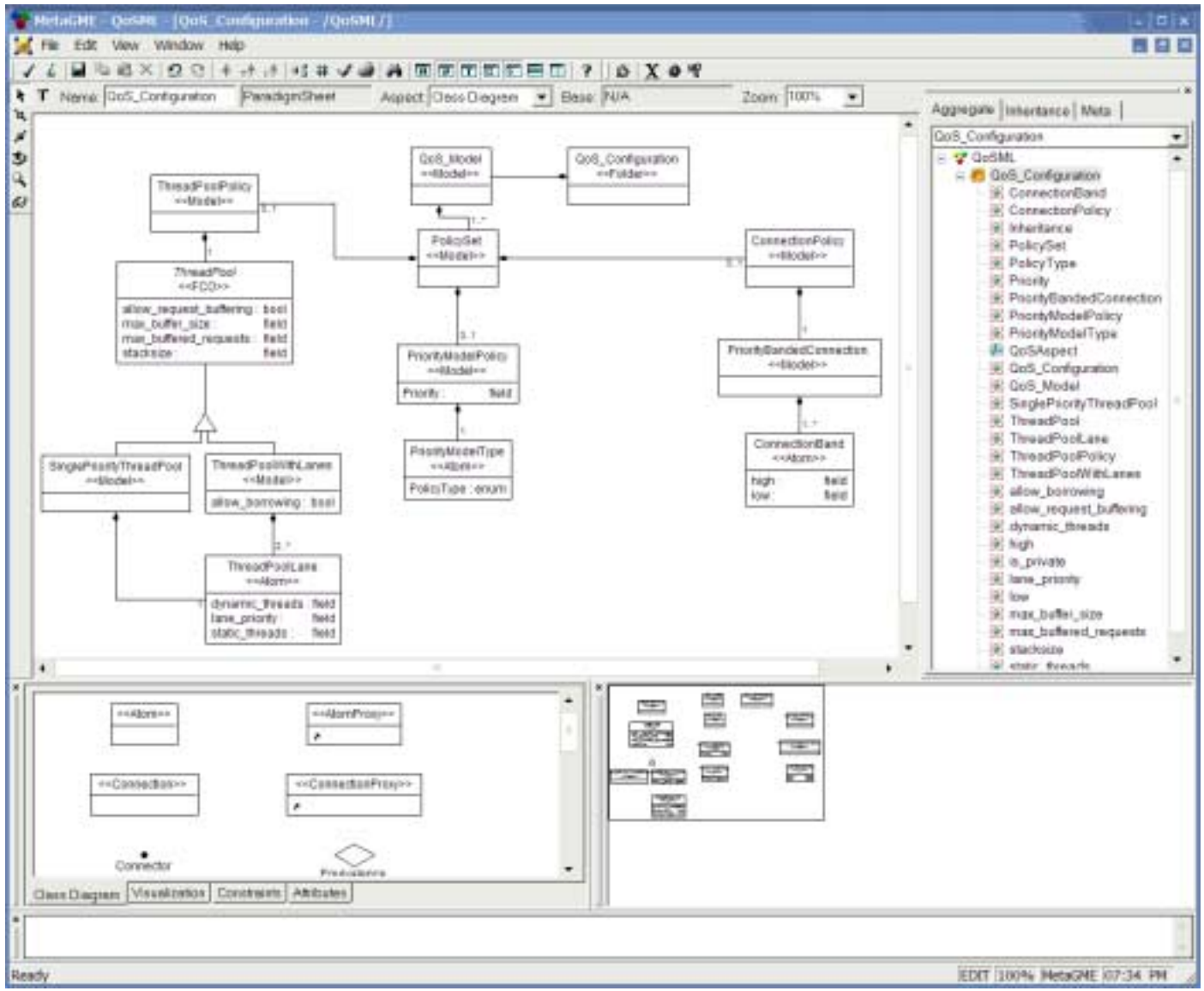
MDD Tool Developers (Metamodelers)

Supports "correct-by-construction" of DRE systems

GME is open-source: www.isis.vanderbilt.edu/Projects/gme/default.htm

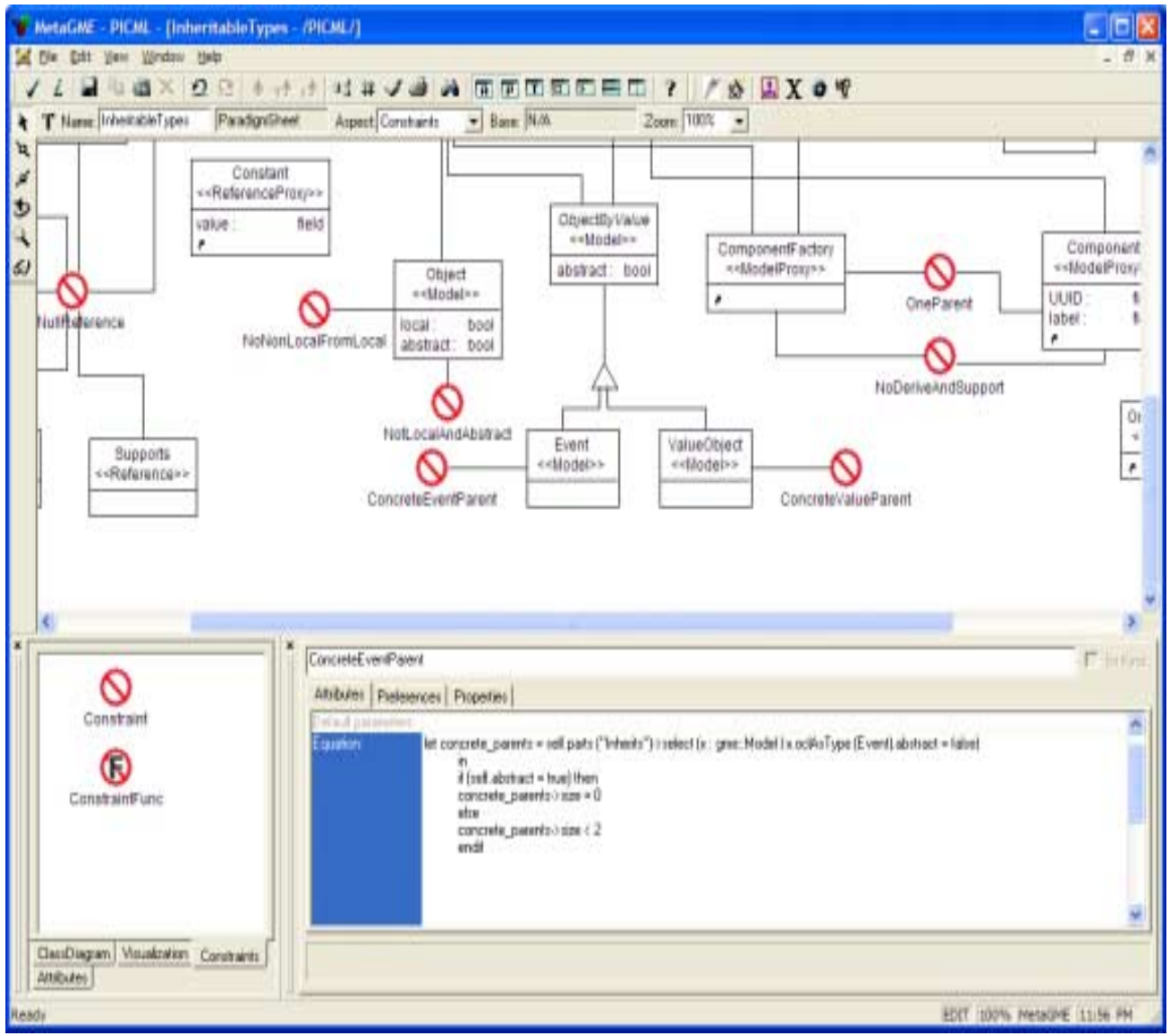
MDD Tool Development in GME

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



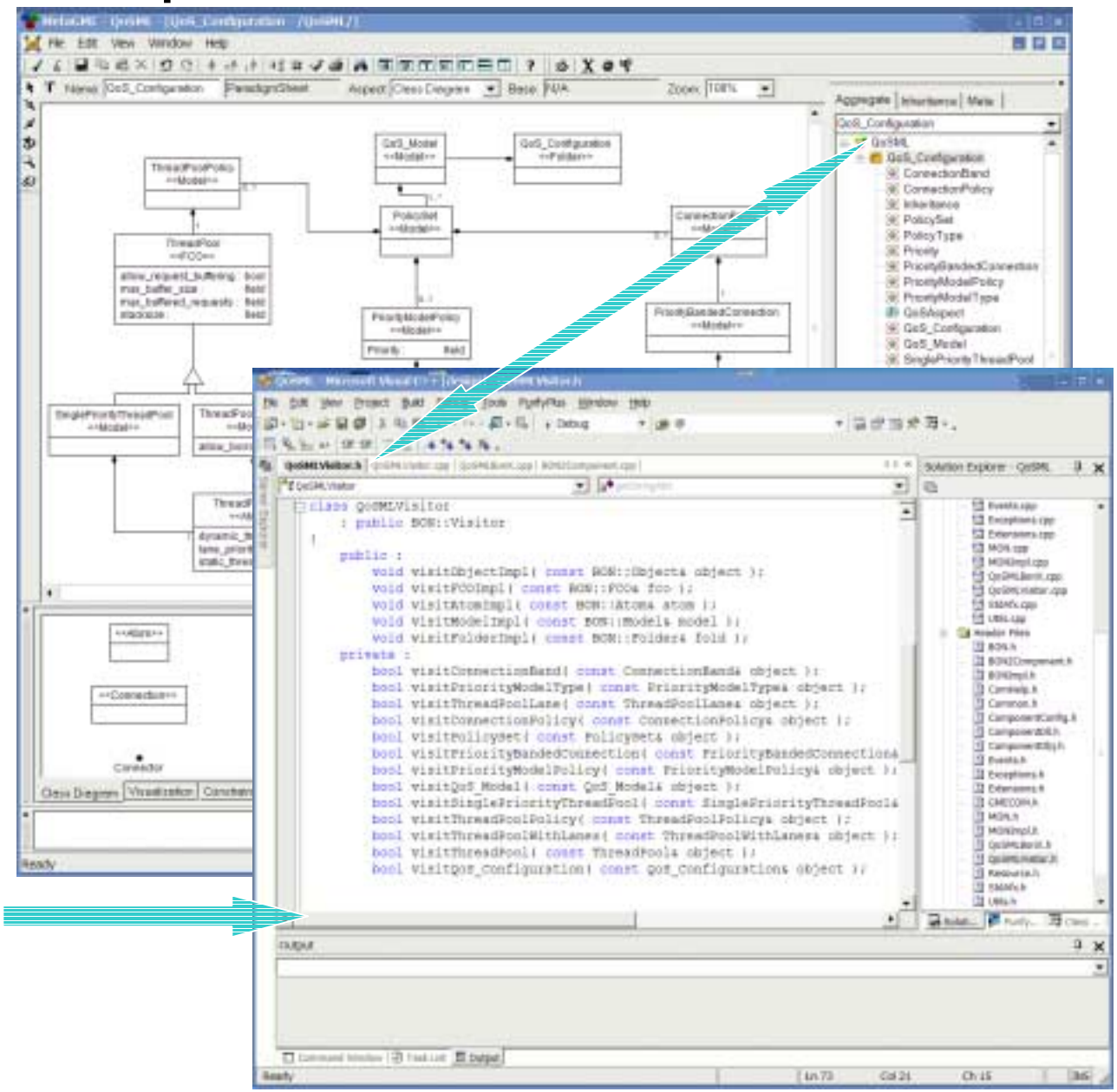
MDD Tool Development in GME

- **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)*



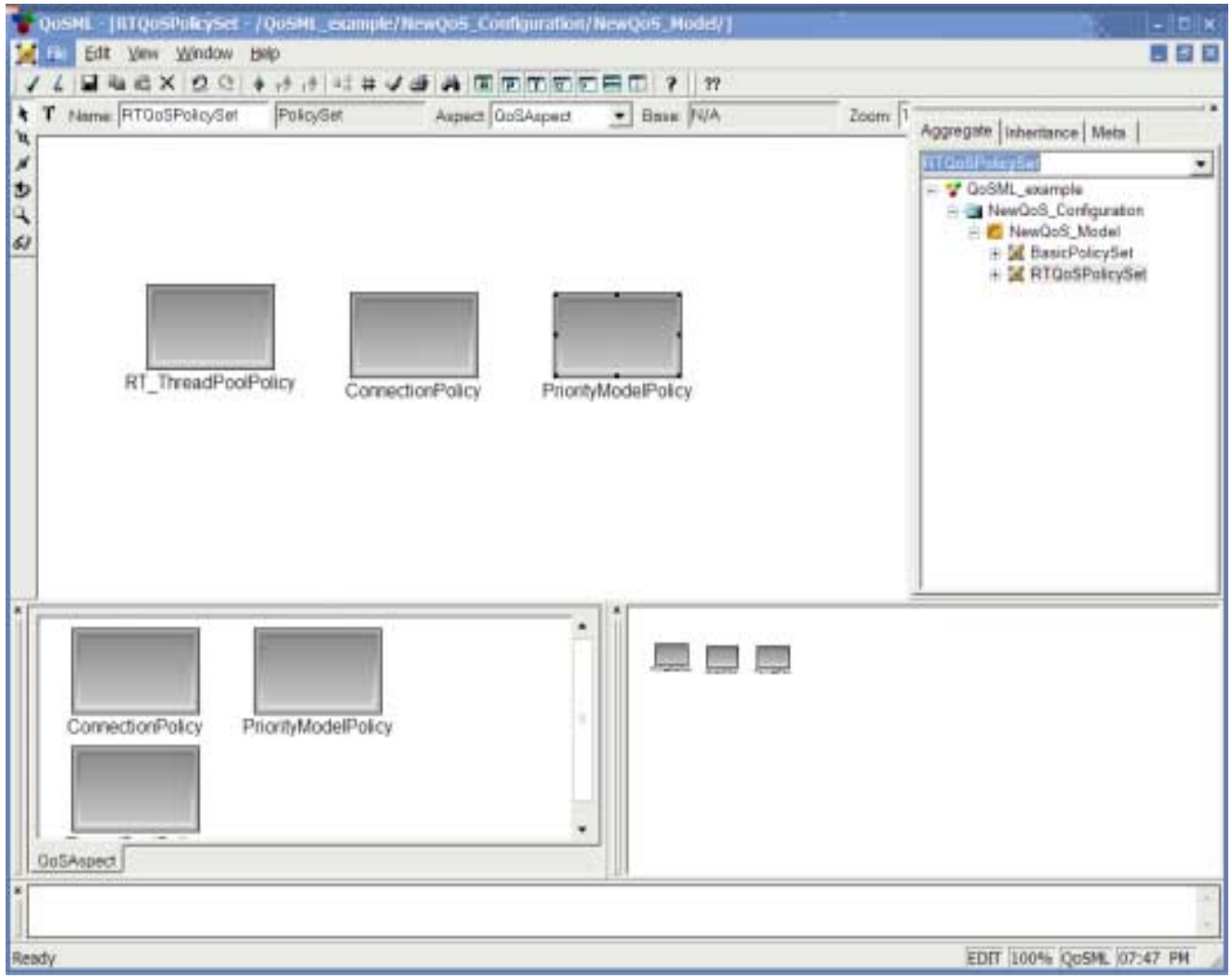
MDD Tool Development in GME

- **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*



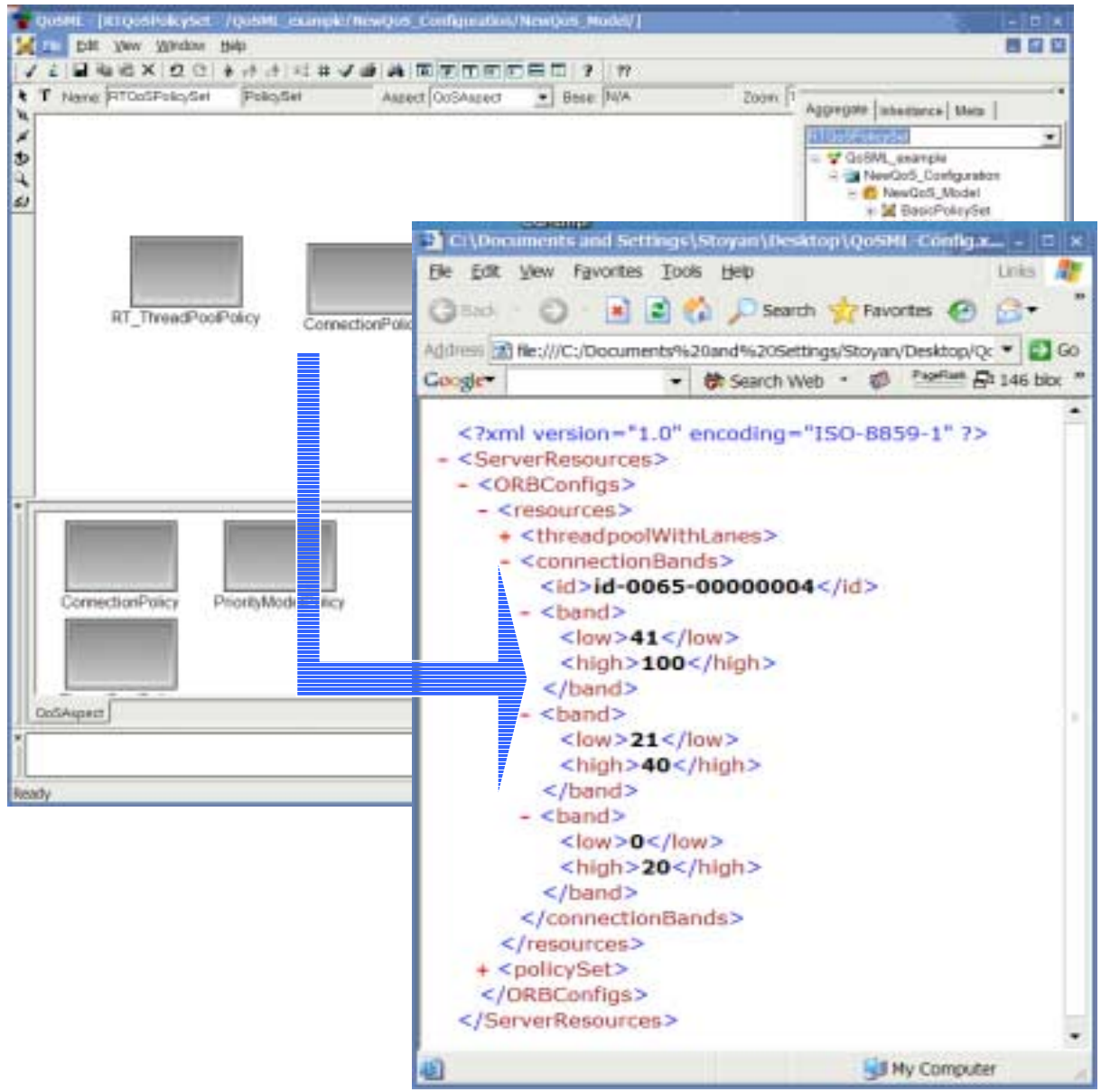
MDD Application Development with GME

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

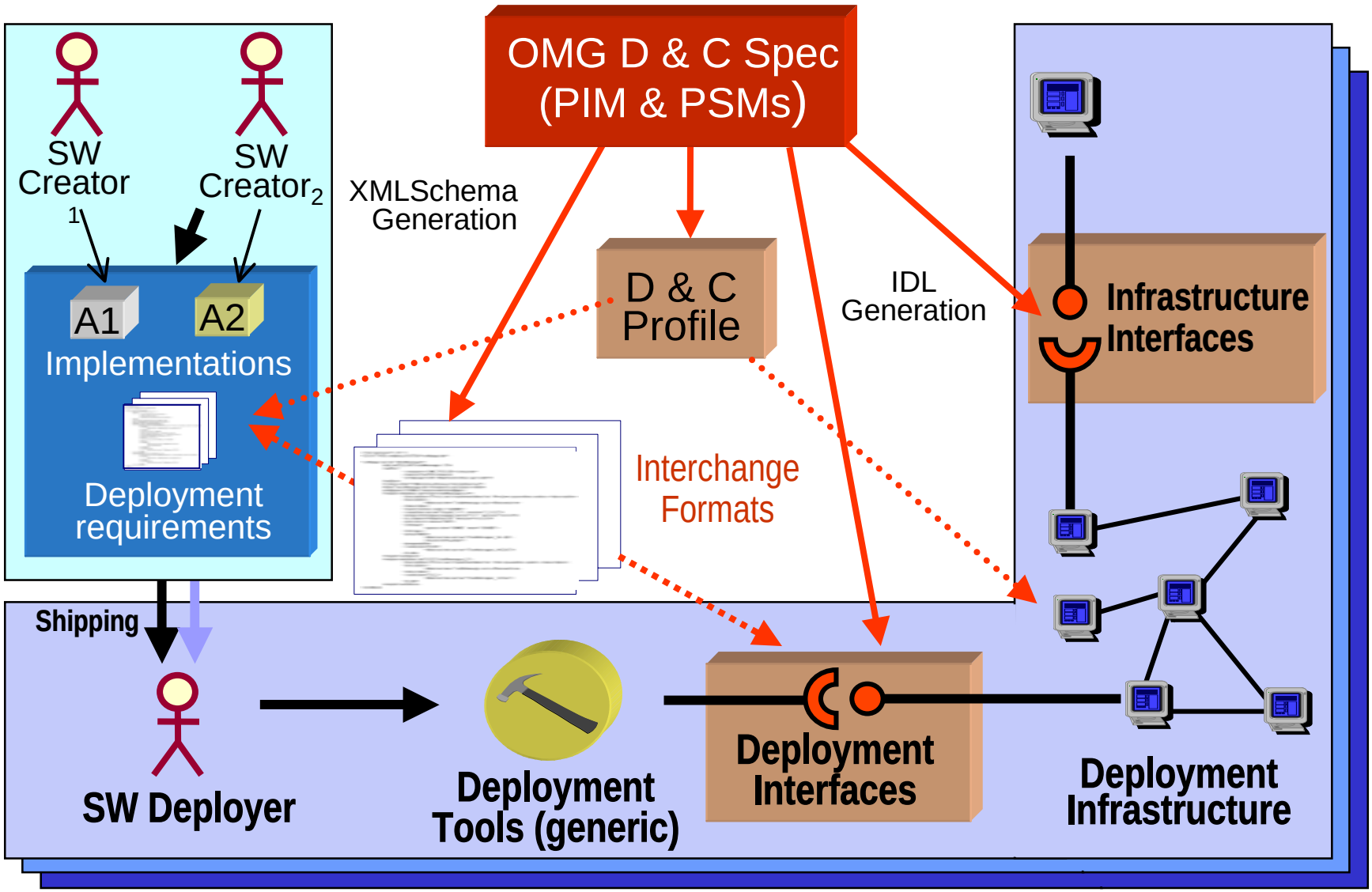


MDD Application Development with GME

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



MDD Example: Deployment & Configuration Aspects (1/2)



www.cs.wustl.edu/~schmidt/PDF/mde-book.pdf

MDD Example: Deployment & Configuration Aspects (2/2)

Specification & Implementation
- Defining, partitioning, & implementation application functionality as standalone components

Packaging
- Bundling a suite of software binary modules & metadata representing application components

Installation
- Populating a repository with the packages required by the application

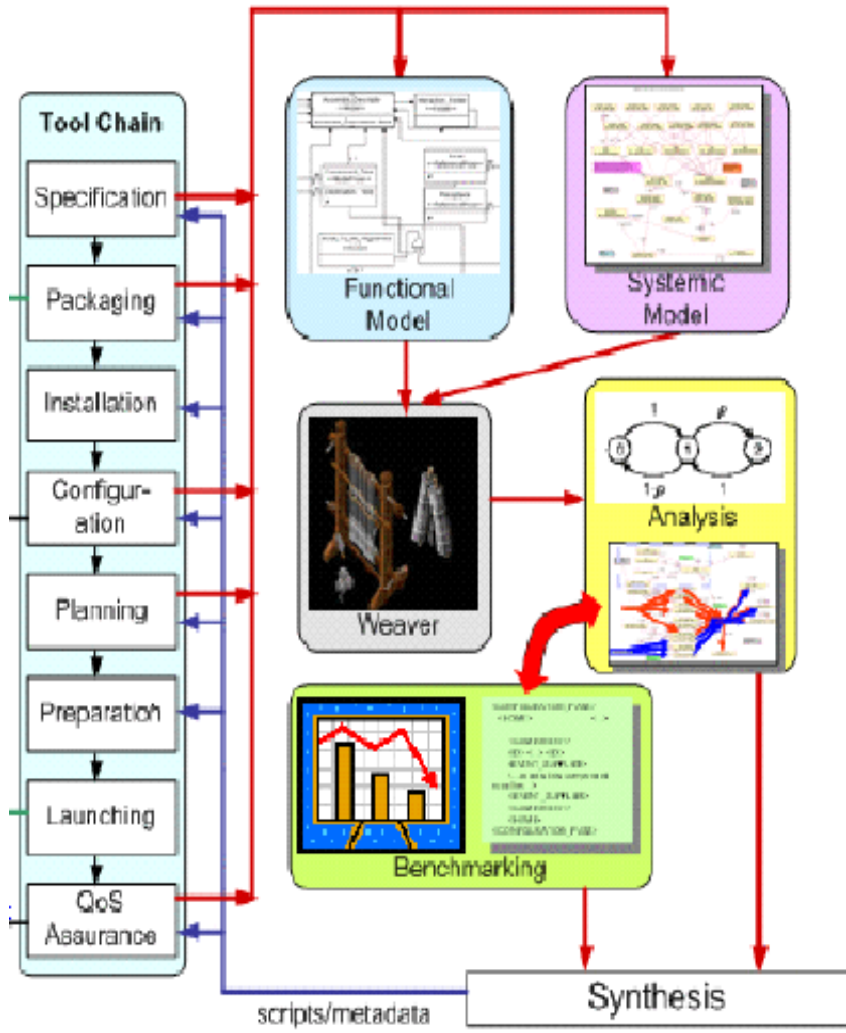
Configuration
- Configuring the packages with the appropriate parameters to satisfy the functional & systemic requirements of an application without constraining to any physical resources

Planning
- Making appropriate deployment decisions including identifying the entities, such as CPUs, of the target environment where the packages will be deployed

Preparation
- Moving the binaries to the identified entities of the target environment

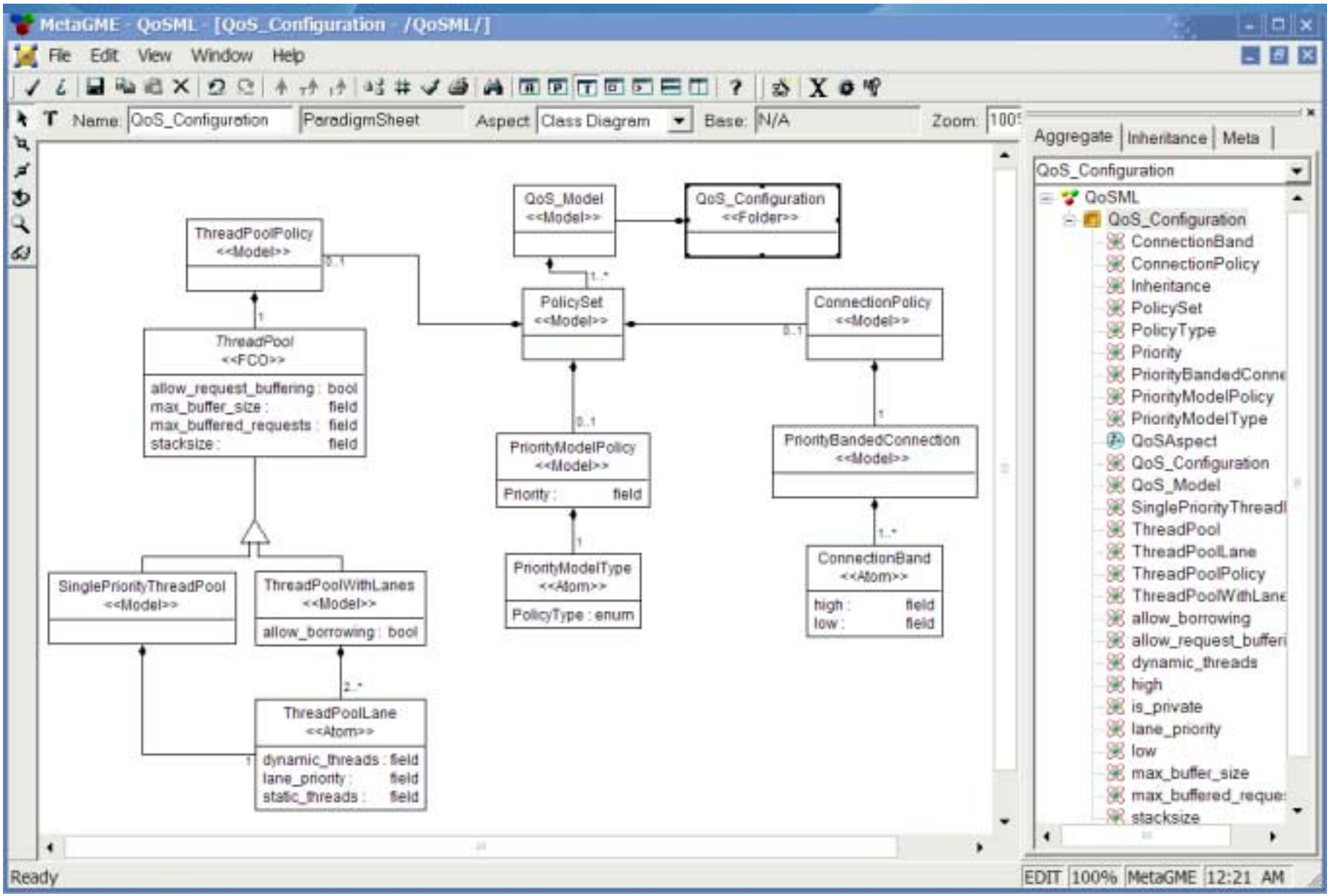
Launching
- Triggering the installed binaries & bringing the application to a ready state

QoS Assurance & Adaptation
- Runtime reconfiguration & resource management to maintain end-to-end QoS



OMG Deployment & Configuration (D&C) specification (ptc/05-01-07)

The QoSML Meta-Model



Features

- Allow you to configure and RT CORBA application for quality of service
- Configurable features:
  - thread pool model
  - connection model
  - priority propagation model
- Create quality of service models
- Synthesize XML configuration data via the model interpreter
- Configuration scripts are then fed into the Deployment and Configuration runtime to automatically configure the desired quality of service properties

Conclusion

- Programmatic use of RT CORBA features is tedious, but it could be avoided by means of MIC
- MIC provides reusability. The interpreter output of a single model can be used to configure multiple applications with similar QoS requirements
- You no longer need to be a RT CORBA domain expert to configure the QoS related features of a CORBA application

Any Questions

References:

- [1]A. Ledeczi, “Metaprogrammable Toolkit for Model-Integrated Computing," Proceedings of the IEEE ECBS’99 Conference, 1999.
- [2] Douglas C. Schmidt and Steve Vinoski, Real-time CORBA, Part 1: Motivation and Overview, C/C++ Users Journal, December, 2001.
- [3] Douglas C. Schmidt and Steve Vinoski, Real-time CORBA, Part 2: Applications and Priorities, C/C++ Users Journal, January, 2002.
- [4] Douglas C. Schmidt and Steve Vinoski, Object Interconnections: Real-time CORBA, Part 3: Thread Pools and Synchronizers C/C++ Users Journal, March, 2002.
- [5] Douglas C. Schmidt and Steve Vinoski, Object Interconnections: Real-time CORBA, Part 4: Protocol Selection and Explicit Binding, C/C++ Users Journal, May, 2002.
- [6] Greg Nordstrom. “Formalizing the Specification of Model Integrated Program Synthesis Environments”.