Fraunhofer FOKUS Institute for Open Communication Systems

IKV++ TECHNOLOGIES AG

NTT DATA

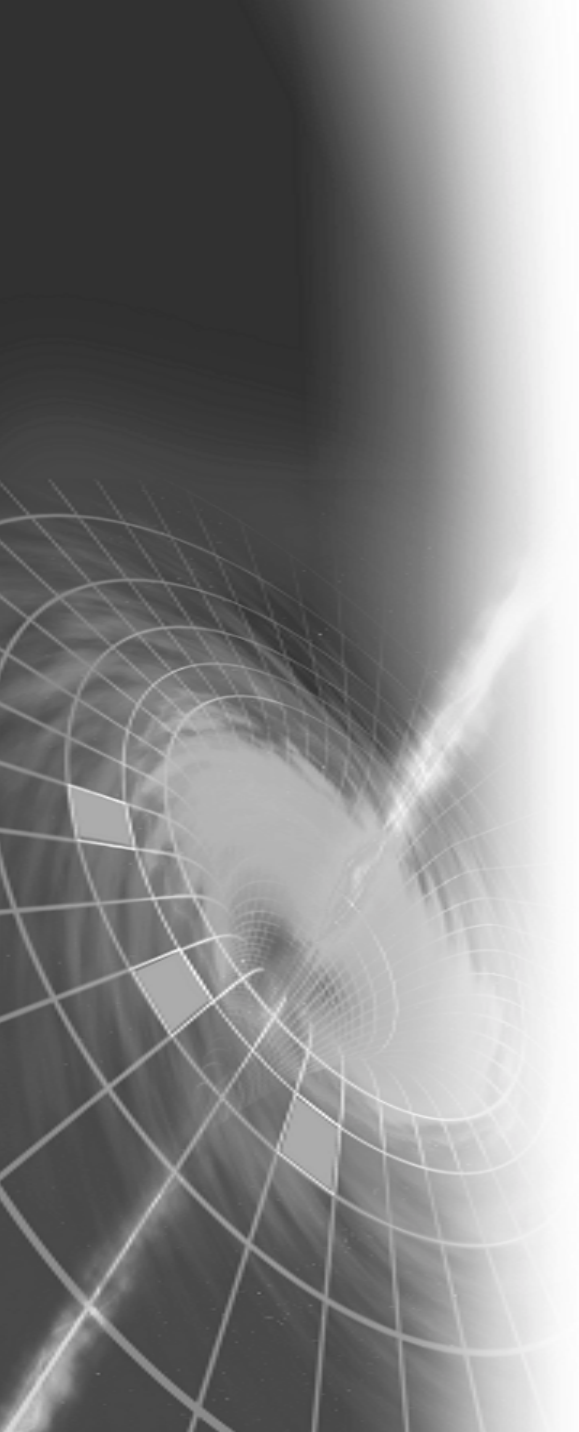
# *Model Driven Service Creation for eGovernment Services*


**Dr. Olaf Kath**
kath@ikv.de

**Dr. Osamu Iwaki**
iwakio@nttdata.co.jp

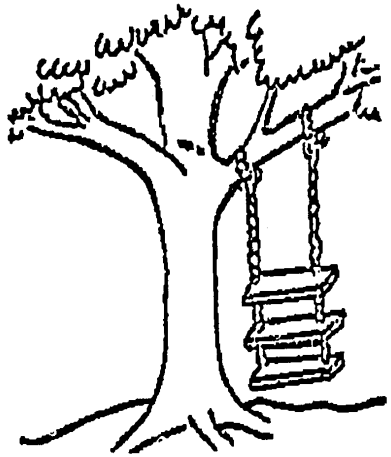
**Dr. Marc Born**
born@fokus.fraunhofer.de

**Shingo Kamiya**
kamiyas@nttdata.co.jp

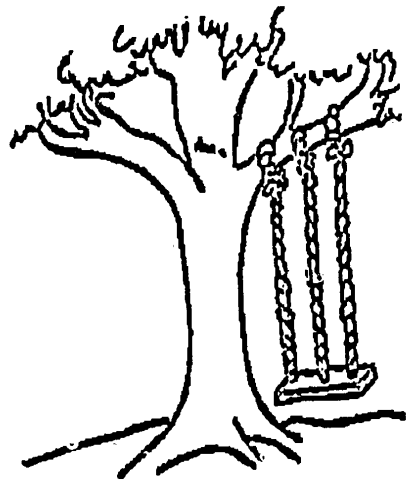


## *Why Model Driven Service Creation?*

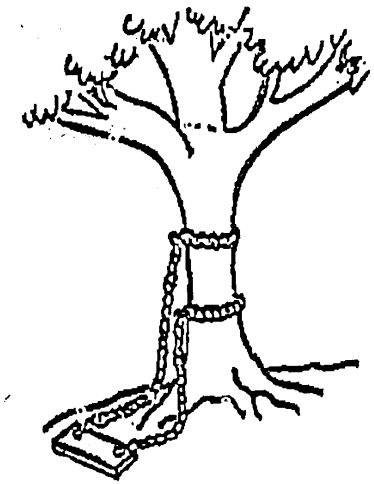
## *Support for systems engineering from requirements analysis down to running solution*



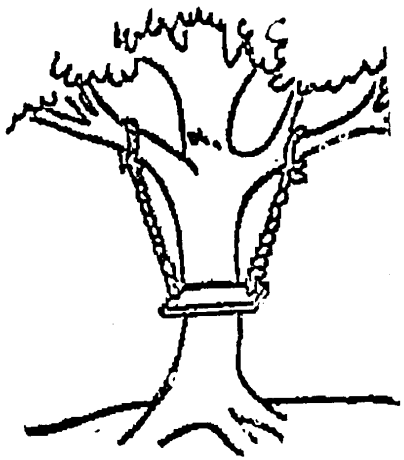
As Management requested it



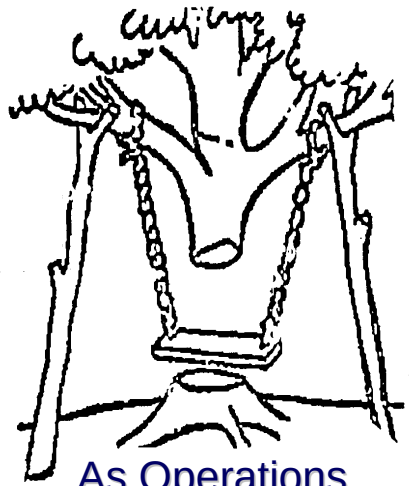
As the Project Leader defined it



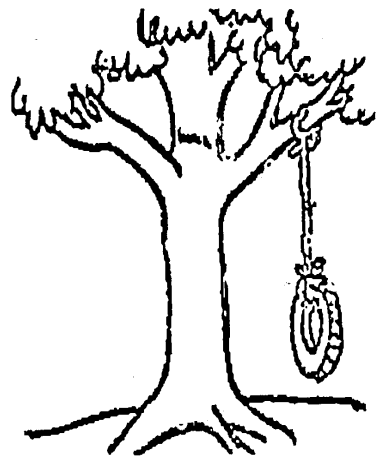
As Systems designed it



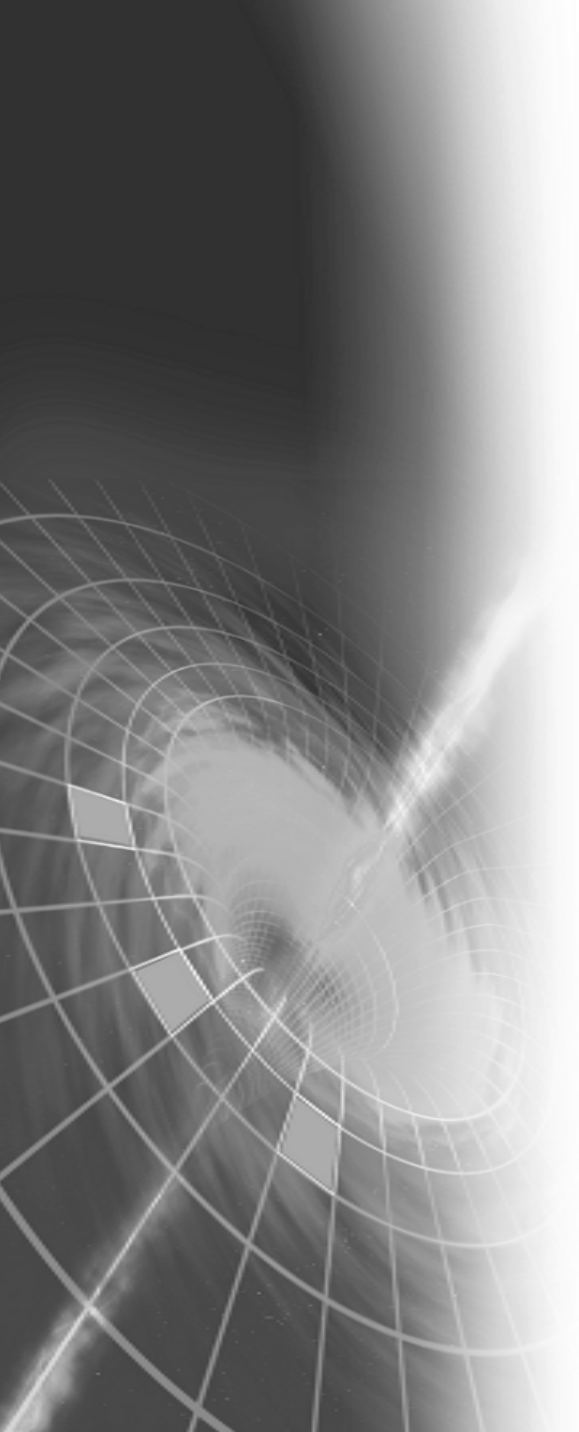
As Programming developed it



As Operations installed it

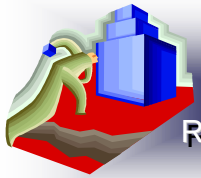


**What the customer wanted**



***The vision of the MDSC project:***

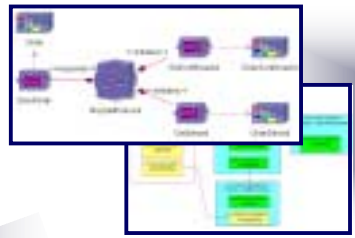
***System development tools running in an environment that consistently connects the output of one with the input of another***



System Requirements

system development

technology independent model (abstract)



platform-independent modeling

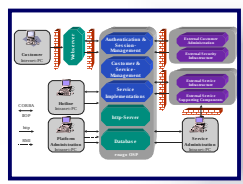
technology-independent model (refined)

platform-specific modeling

technology specific model

integrated-development environment

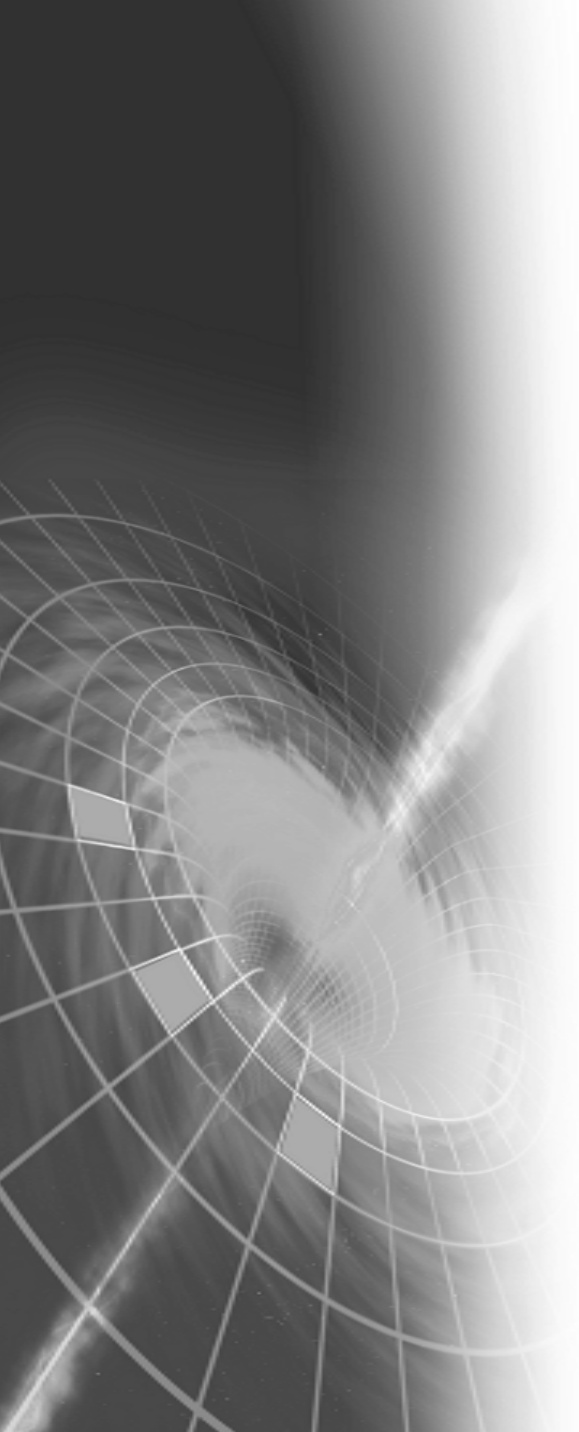
system-components



Service platform for integratin and operaton

system integration, test and operation

system operation

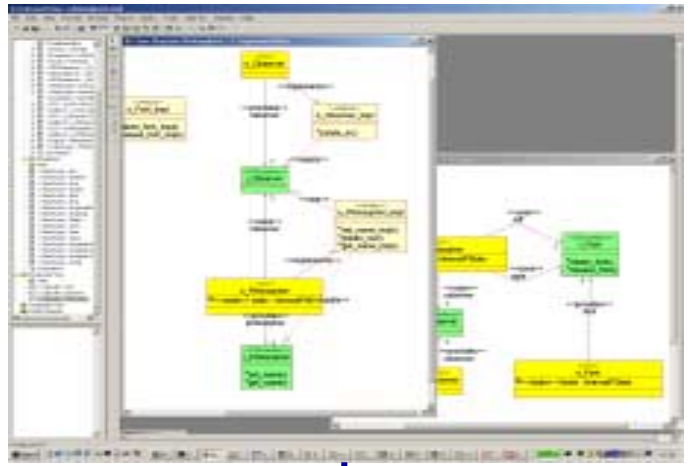


# *Project history - Assessment and Experimentation Results:*

## *3 Fallacies*

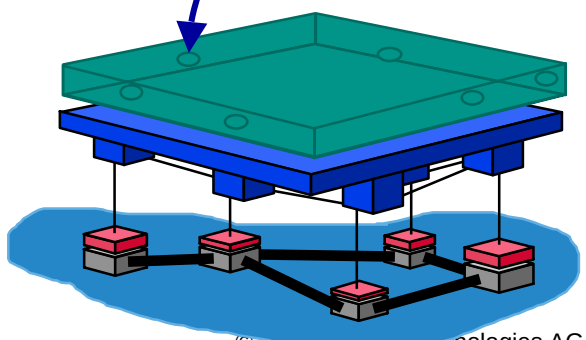
# *Fallacy 1 - One Tool Fits All*

- System Construction is to complex
  - There‘s no ONE tool, that fits to ALL tasks in the process
  - The “We’ve integrated DOORS in our UML tool” isn’t the solution either
- There must be integration of different tools for specific purposes



**tool for modeling**

**one stop code generation**



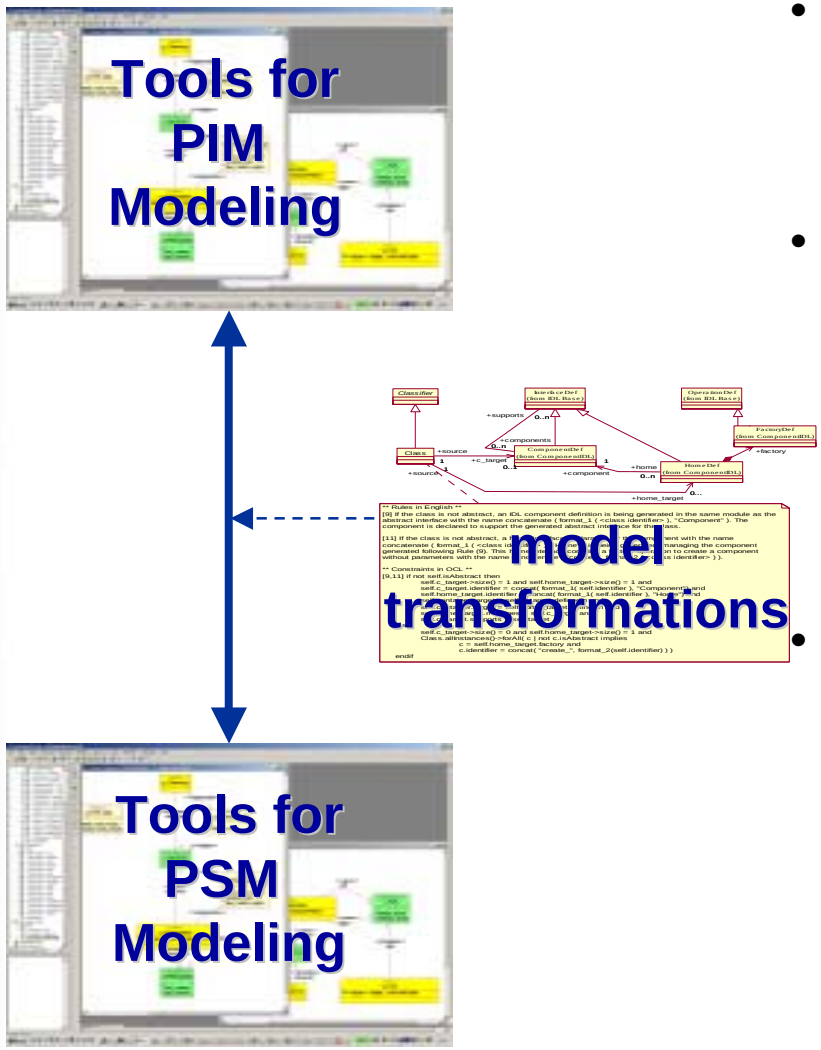
**Platform Components**

# *Fallacy 2 – UML is MDA is UML*

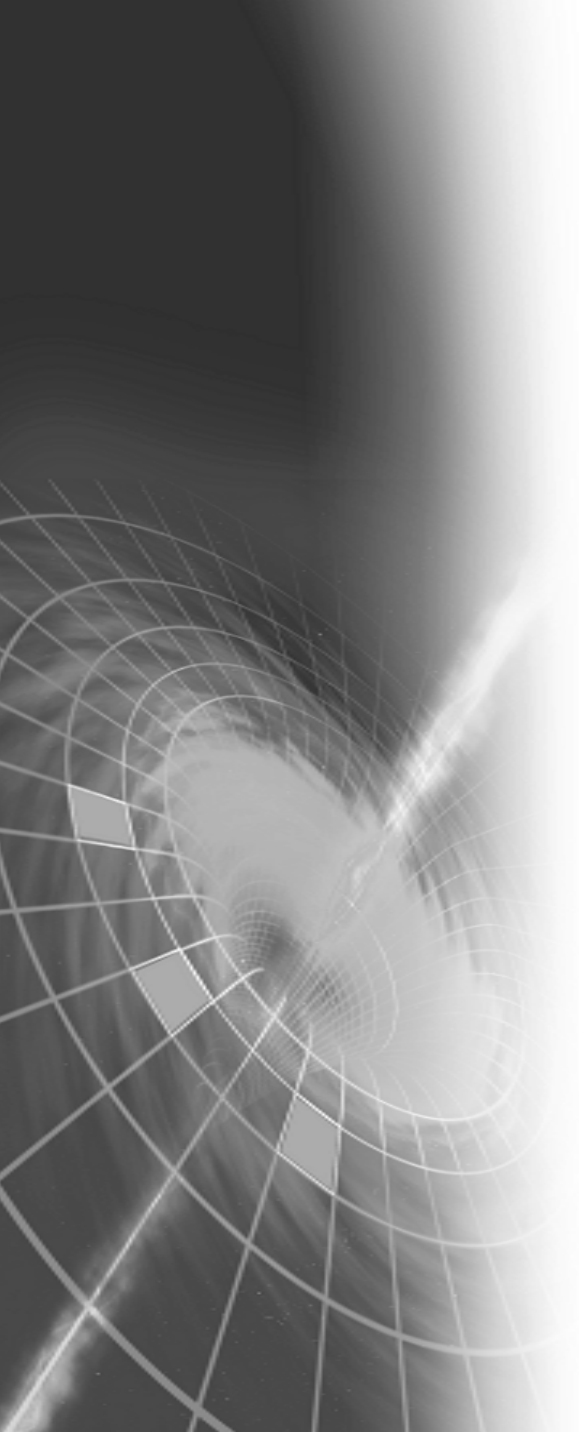
- UML is a general purpose modeling language
- However, UML is not a modeling language for any purpose
- UML Profiles are not sufficient either
  - Define the semantics of business processes on top of the UML Metamodel? Why?

- There must be other techniques to
  - Define the semantics of elements to model e.g.
    - Business processes
    - Platform concepts
  - Define transformation rules between models
  - Define different notations for the same concepts

# *Fallacy 3 – XMI is the Solution for Tool Integration*

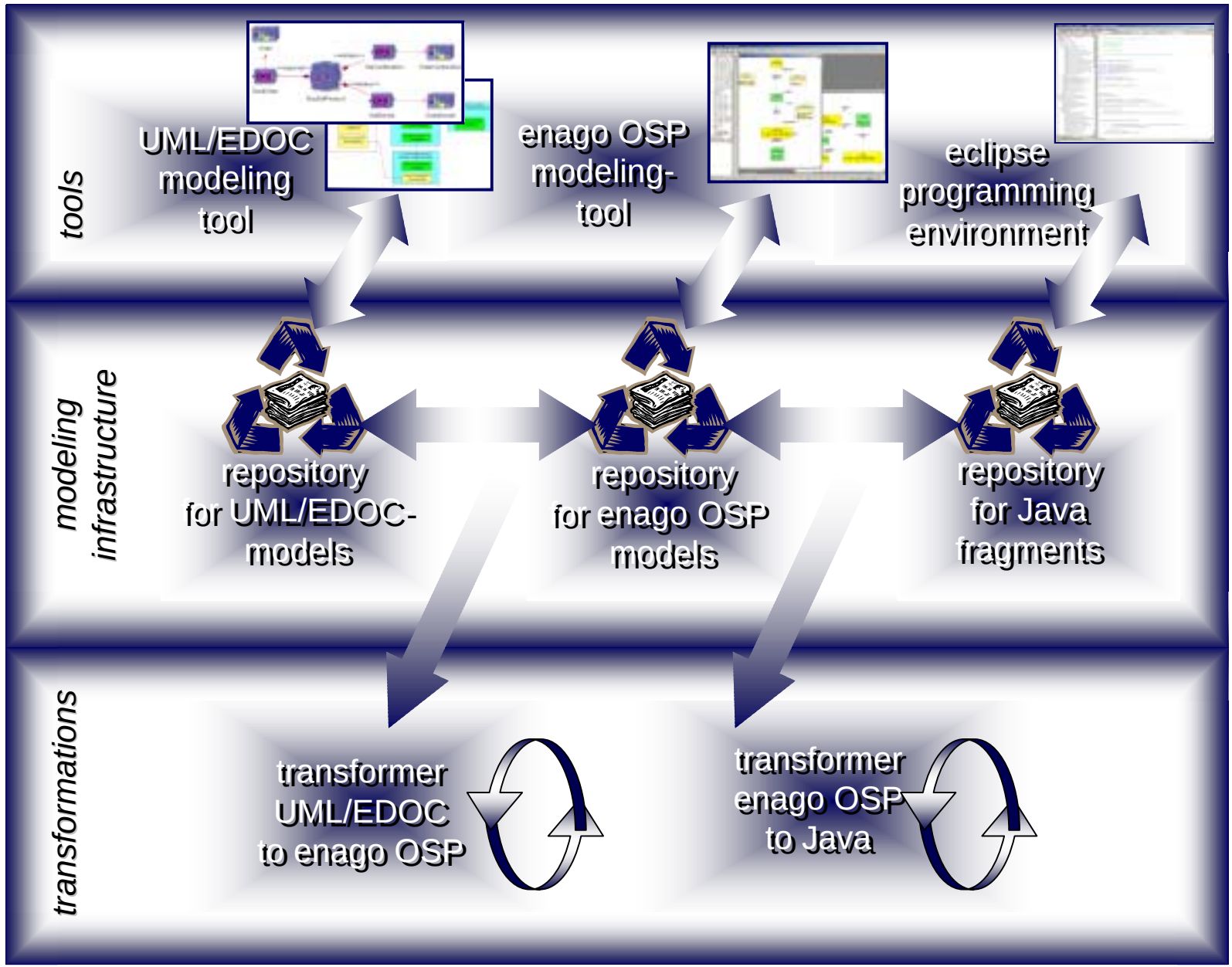


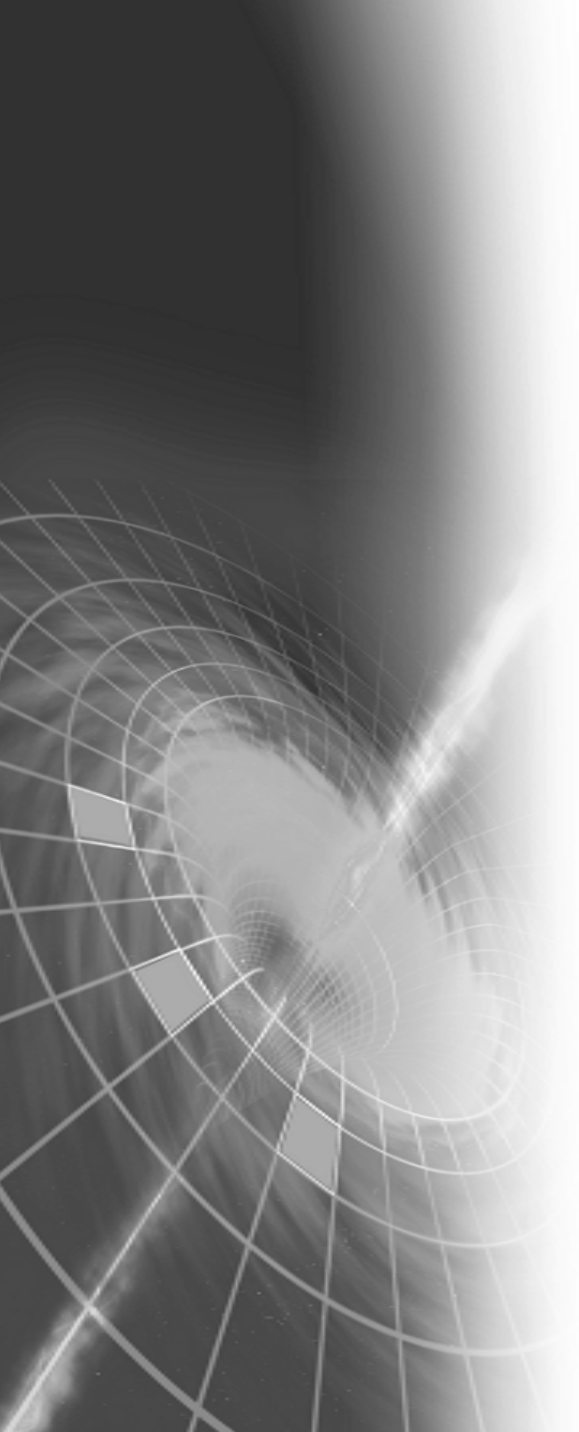

- XMI is a technique to exchange any kind of models between tools
  - specializations for e.g. UML, MOF, CORBA, …
- It can really only be used for this purpose!
  - The “we use XMI and XSLT for transformations between models” won’t work
  - The “we store all models as XMI” won’t work either
- There must be other techniques to manage e.g.
  - tracebility
  - consistency
  - forward/backward transformations



# *The approach to the vision:*

# *The MDSC Big Picture*





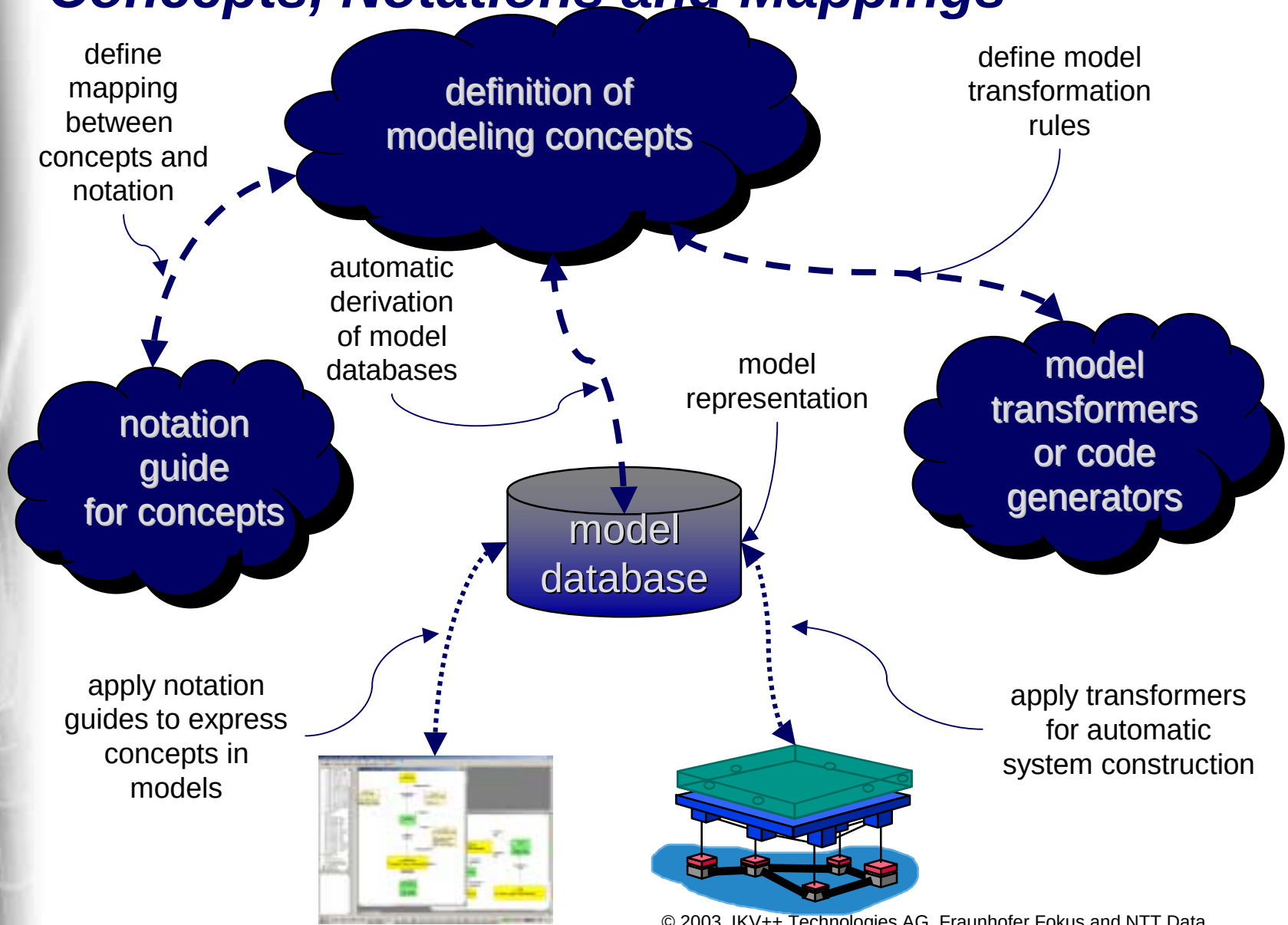
***Some aspects behind the Big Picture:***

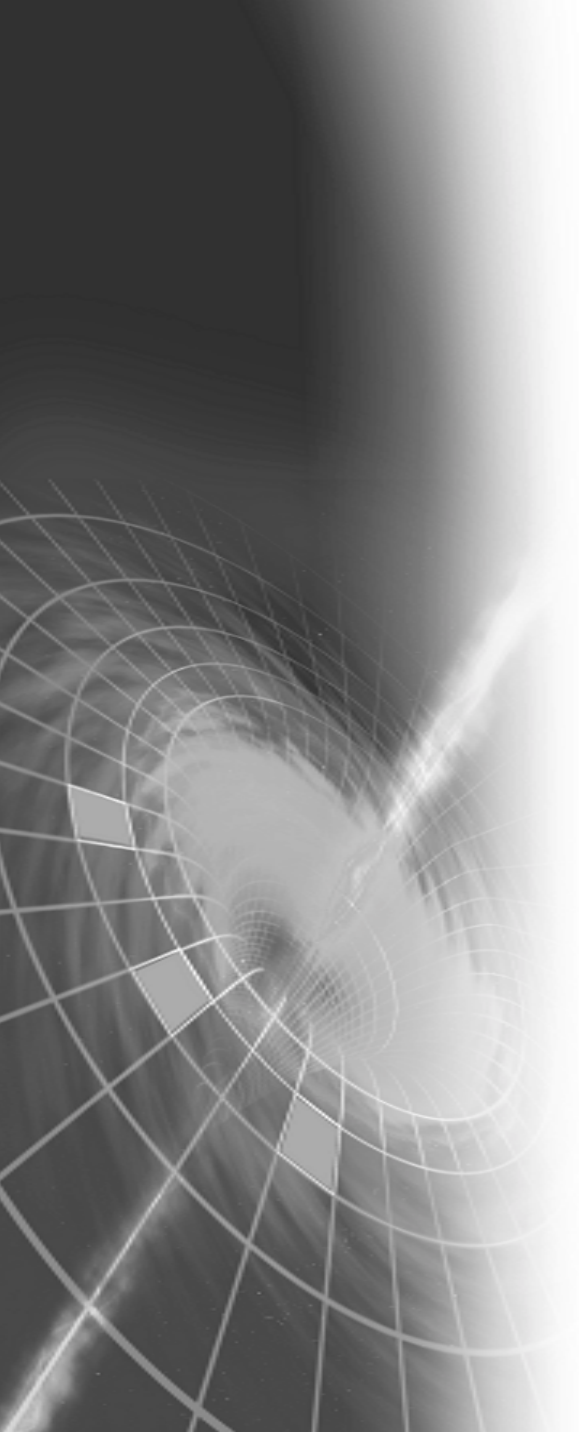
***Architectural guidelines, a standards mess and technology pieces …***

# *Modeling Infrastructure*

- All modeling instruments and principles are defined by models themselves, model databases are derived
  - We use the Meta Object Facility (MOF) for this purpose
- Based on these metamodels, domain specific visual modeling languages are defined
  - We use different techniques here, e.g. UML based languages, textual languages, new graphical languages
- Also based on the metamodels, further processes like code generation or model transformations are defined
  - We keep notation and these processes separate

# Architecture Guidelines: Concepts, Notations and Mappings



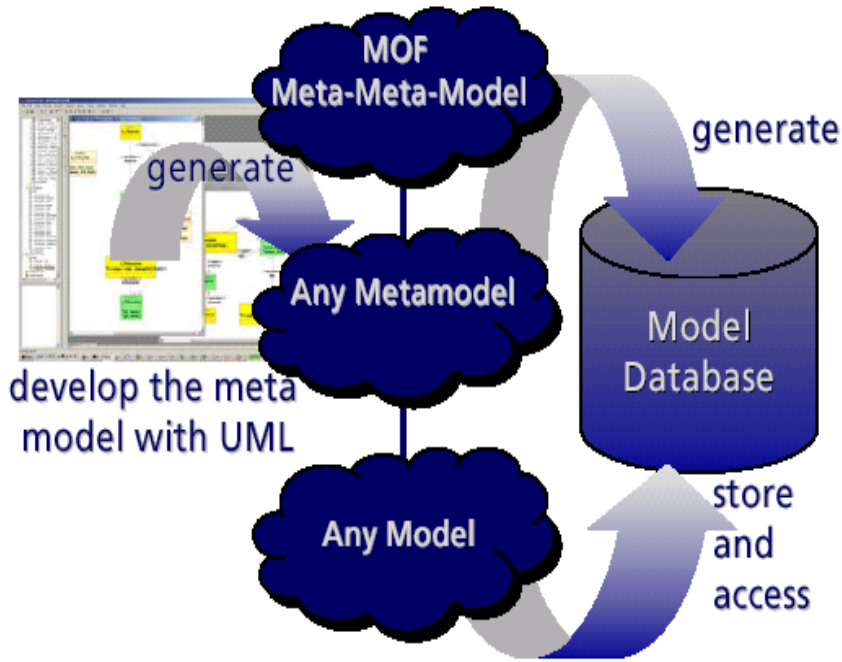


***Project results:***

***- PIM Modeling Tool based on EDOC***
***- enago OSP modeling tool***
***- MDSC eclipse integration***
***- modeling infrastructure for tool collaboration***
***- model transformer generator***
***- concrete model transformers***

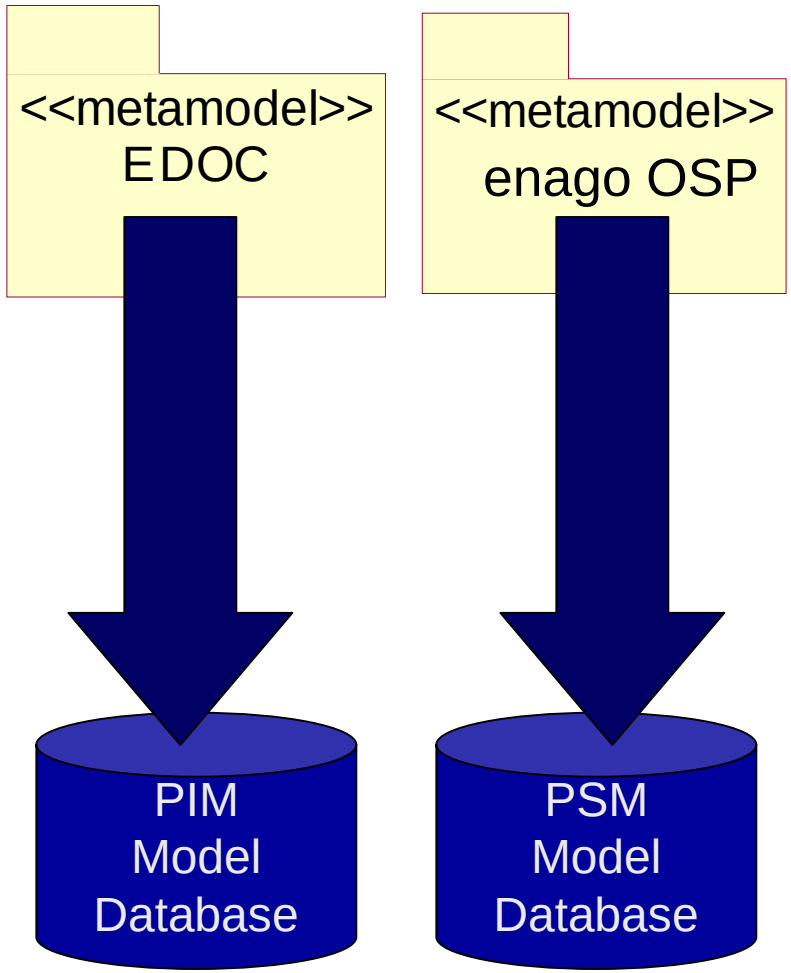
# *MDSC modeling infrastructure*

- Meta Object Facility (MOF) provides a technology to share models based on their metamodels
- The sharing of models is to be fined between tools
- There exist a number of technology mappings of the MOF:
  - MOF to XML → XMI
  - MOF to Java → JMI
  - MOF to IDL → CORBA
- Out of a metamodel, a set of CORBA interfaces is being generated that can be used to exchange models




- ☞ *the medini tool of IKV realizes the MOF to IDL technology mapping*
- ☞ *medini is being applied to connect modeling tools to the modeling infrastructure and*
- ☞ *to establish connections within the infrastructure*

# *Application of medini*




Automatically generate model databases out of the metamodels, using *medini*

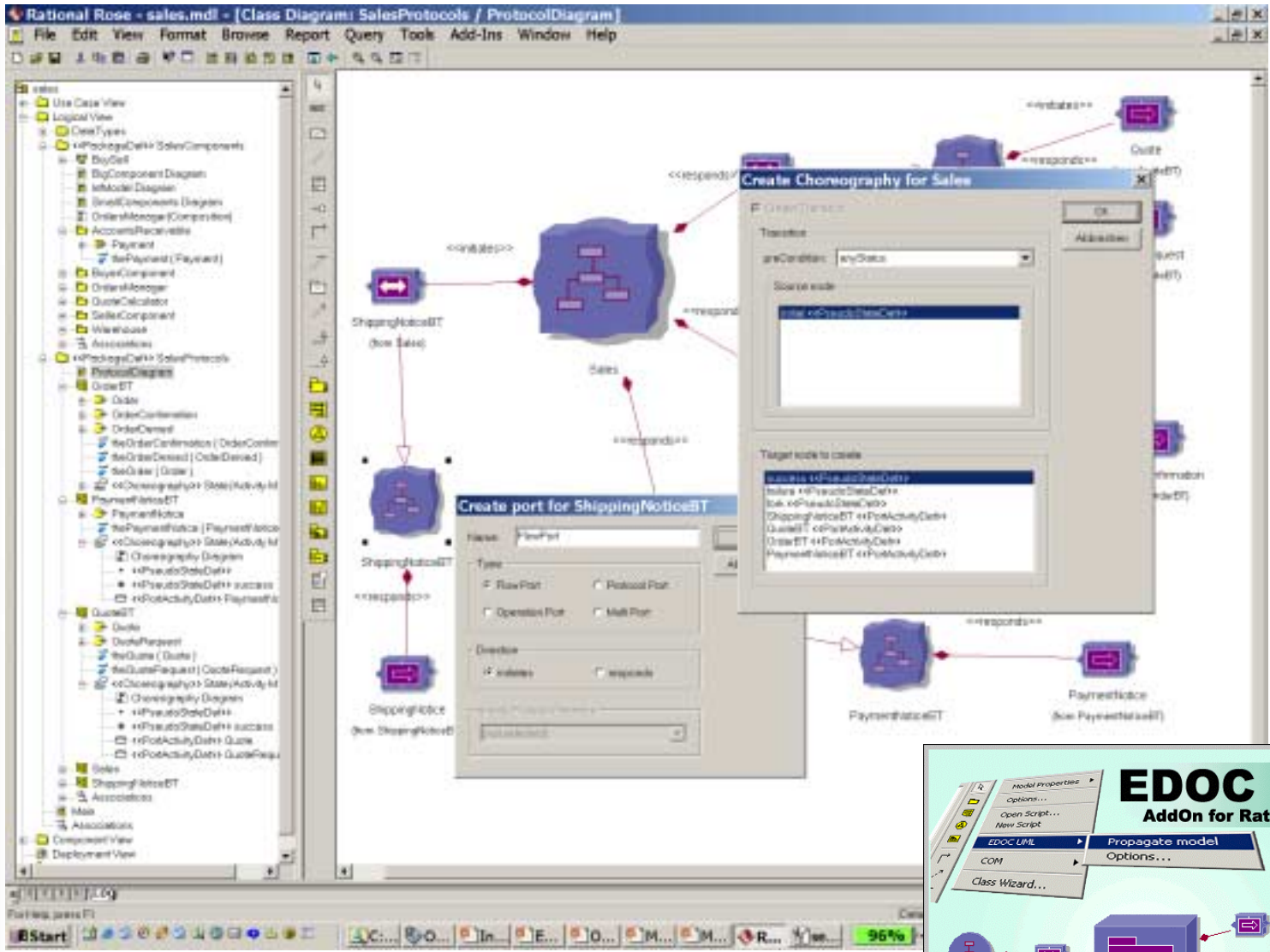
These model databases support mechanisms for

- CORBA access
- transactions
- model change notifications
- XML import/export
- consistency check
- ...

# *EDOC based PIM Modeling*

- EDOC is an OMG standard for modeling of enterprise distributed systems
  - *Component Collaboration Architecture* - Component based definition of business process modules, definition of component collaboration using choreographies
  - *Entity Profile* – definition of information models and their connection to business processes
  - *Events* – definition of messaging based interactions between business components
- In MDSC, EDOC is being realized as an extension to the Rational Rose modeling tool
  - Specific modeling wizards, dialogs, …

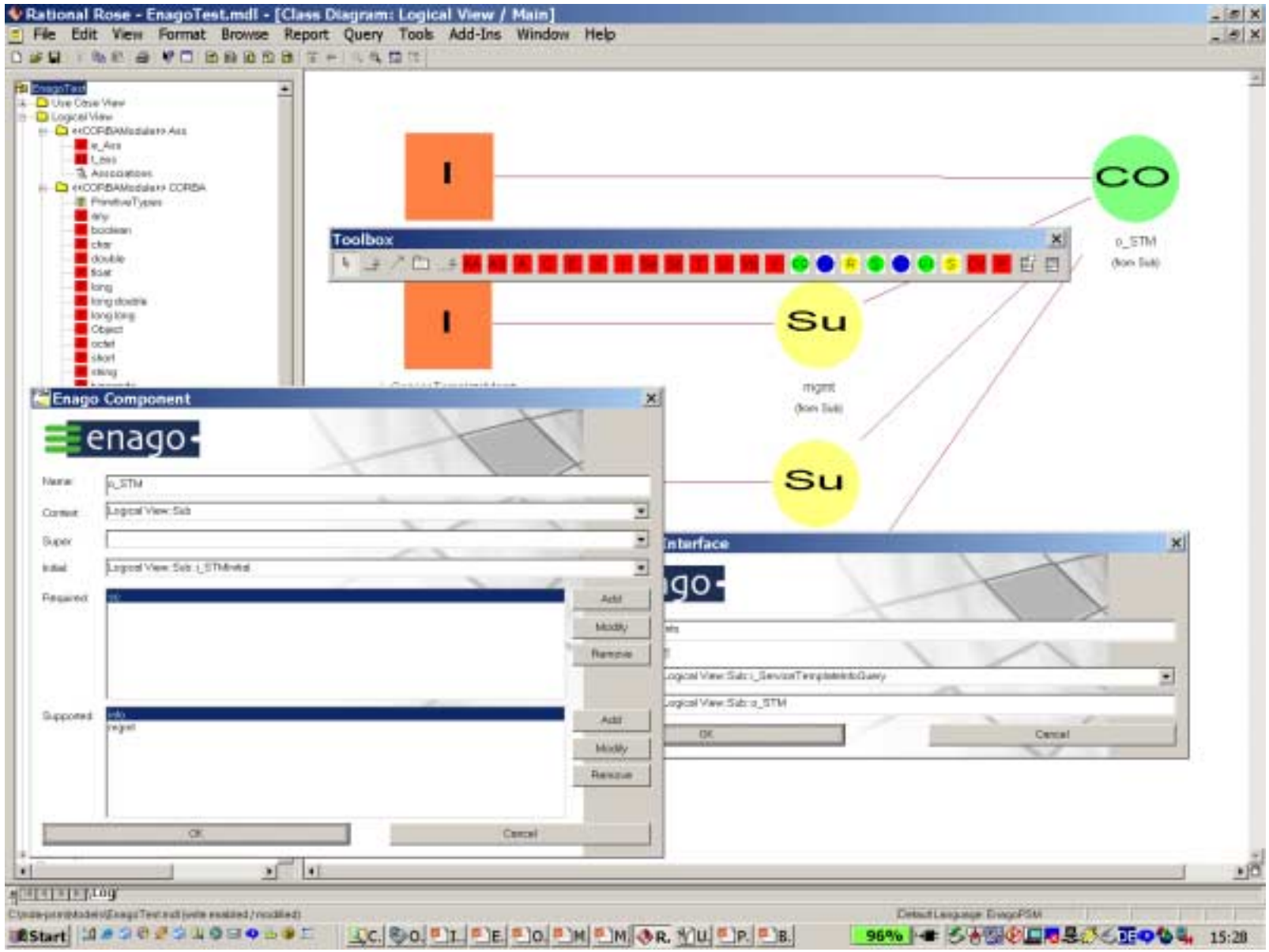
# EDOC based PIM Modeling



# *enago OSP Modeling*

- enago OSP is IKVs CORBA based platform for rapid definition, realization, deployment and operation of distributed services
  - introduces the service concept, *services* can be provided, used, sold, accounted on and so forth
  - Services are being realized by software components
- based on the CORBA metamodel, we developed a metamodel and a UML profile *for modeling of enago services*
  - realized again as Rational Rose extension

# *enago OSP Modeling*



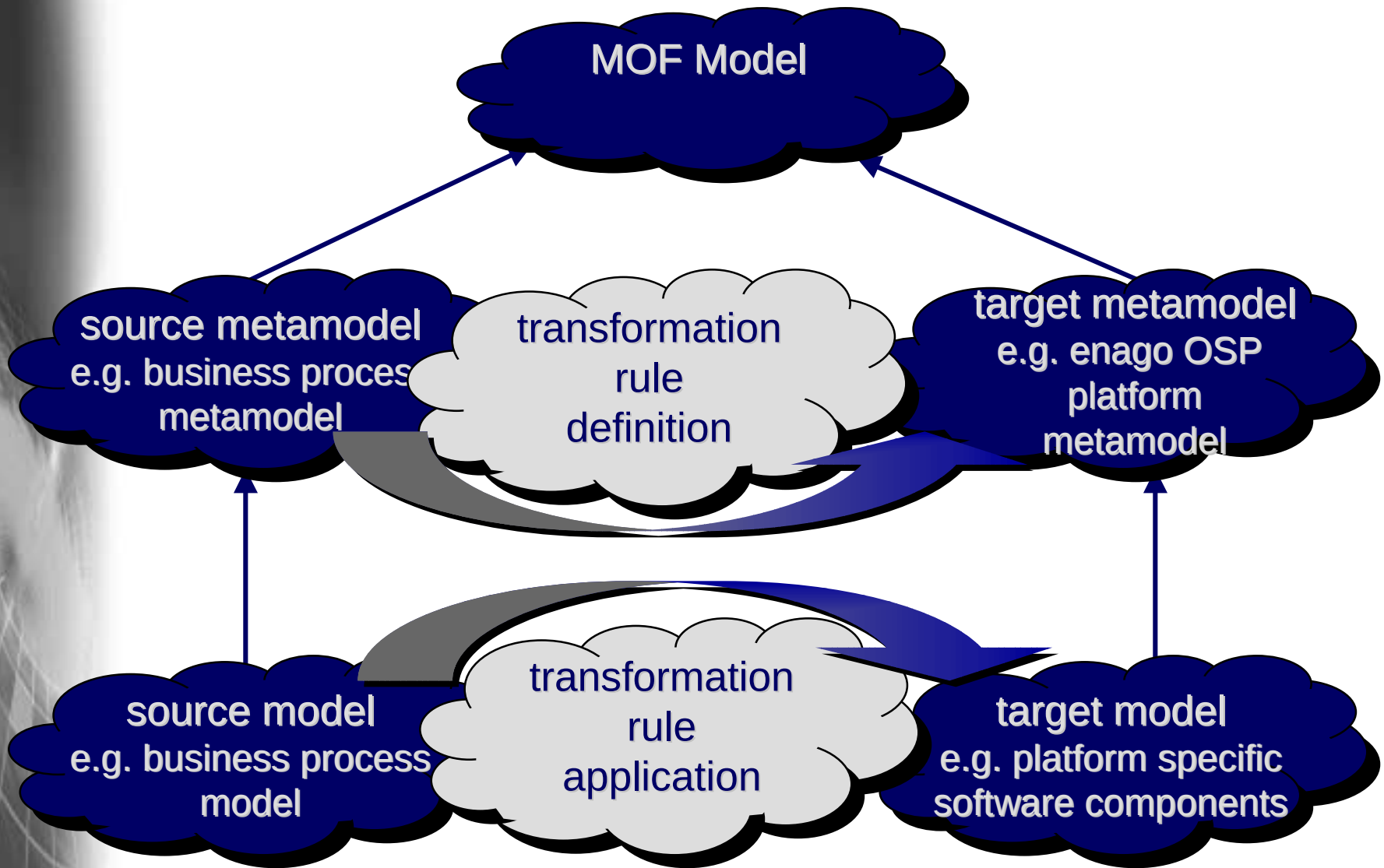
# *MDSC eclipse integration*

- Purpose: generation of Java code out of enago OSP models
  - The modeler defines an enago OSP model of collaborating services
  - The services are being realized by software components
  - These components are implemented in the Java language
- eclipse provides an ideal Java development environment
- in MDSC, we use eclipse as target for code generation

# *Model Transformation*

- Model transformation is the new software construction paradigm
  - Systems are being constructed by transforming a model of higher abstraction into a model of lower abstraction
  - Programming language code is also being treated as a (part of a) model of a system
- In MDSC, we find two categories of model transformations
  - Transformation of EDOC models into enago OSP models
  - Transformation of enago OSP models into Java programming language code

# Model Transformation

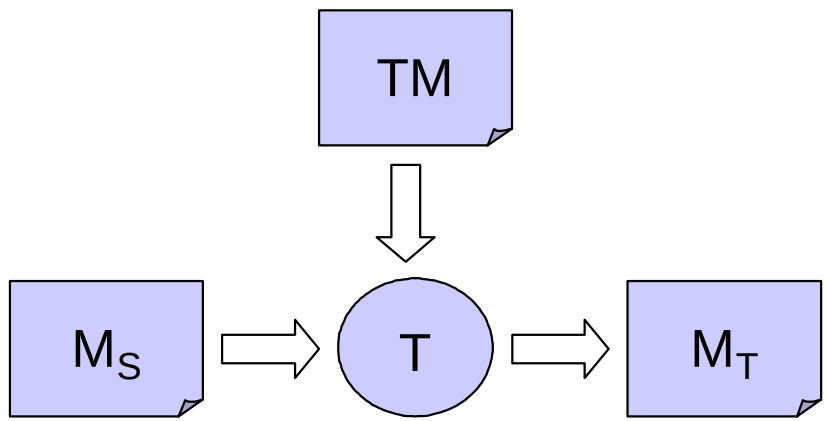


# *Model Transformations in MDA*

- PIM -> PSM is most mentioned transformation
  - Model refinement (partially also PIM->PIM and PSM->PSM)
  - Different abstraction levels
- Also important: PIM->PIM, PSM->PSM on same abstraction level
  - Version upgrading
  - Models of the same domain with similar content but slightly different modeling concepts
  - ...

# *When talking about Model Transformation Techniques ...*

- ... no need to stick to abstractions like PIM and PSM
- ... no need to consider abstraction lelvels
- ... no need to distinguish between model transformation and code generation
- ... just use terms source, target, transformation model, graph, etc.



# We are on the right track!