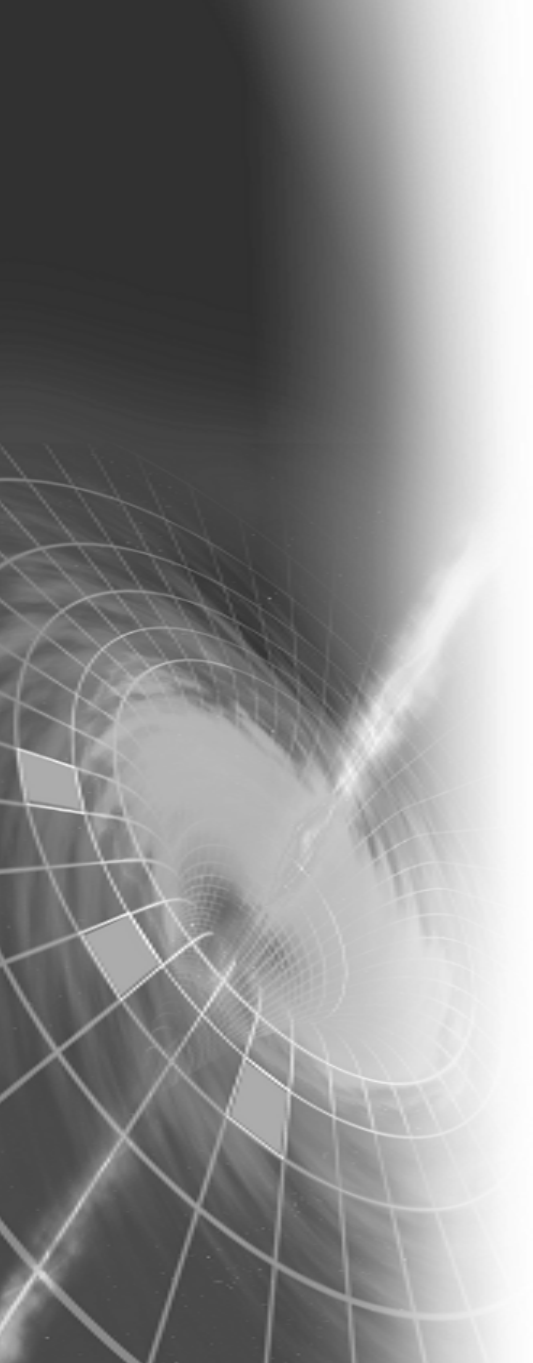
Transformation of EDOC Behavior Models to a Telecommunication Service Provisioning Platform

Joachim Hößler & Dr. Olaf Kath
Technical University Berlin
Dr. Marc Born & Andrej Blazarenas
Fraunhofer FOKUS
Mario Winkler
IKV++ Technologies AG

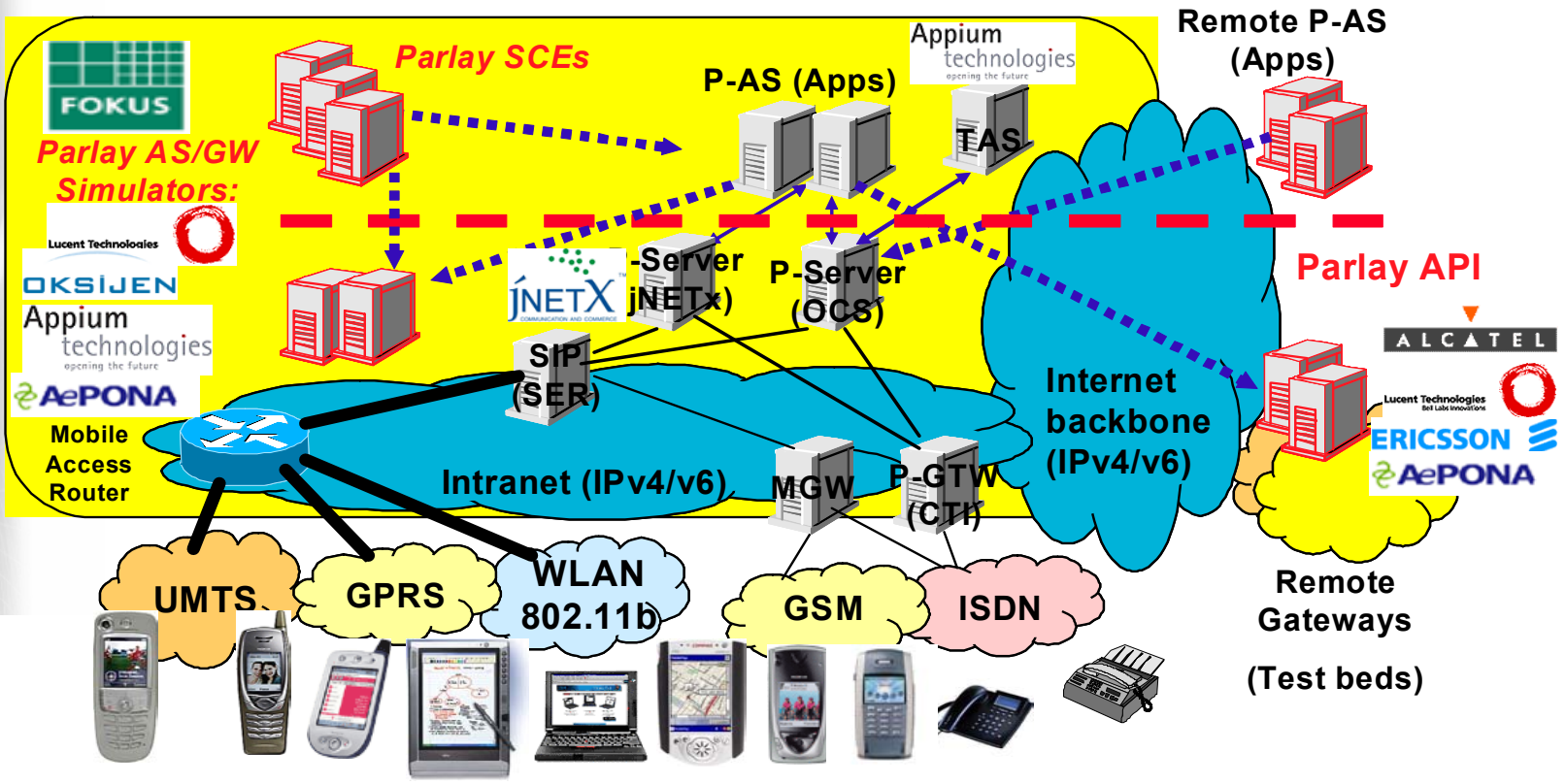


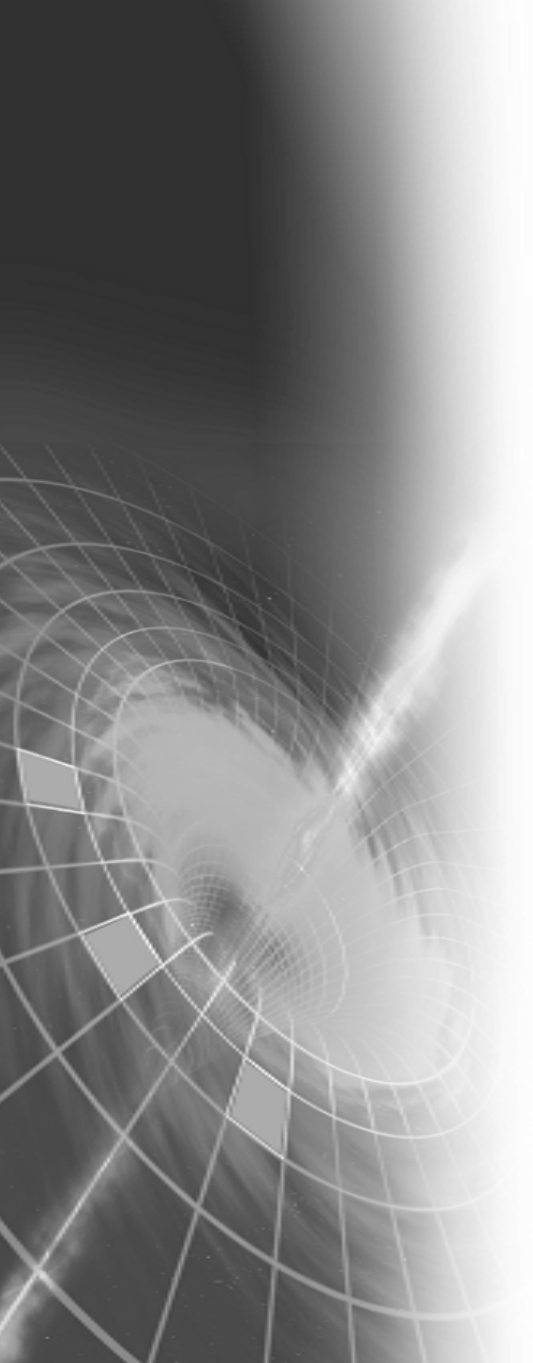
Why Service Creation for 3Gb Networks?

- service offerings differentiate network operators
- time from service idea to deployed service 0 + small ε

But: Networks are to complex for service creators

The 3Gb Playground at FOKUS

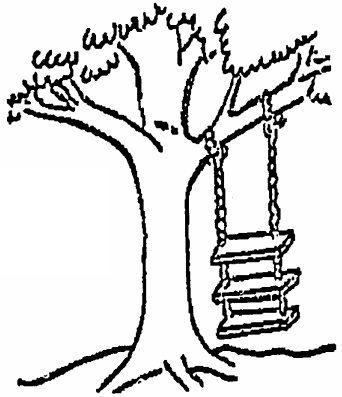




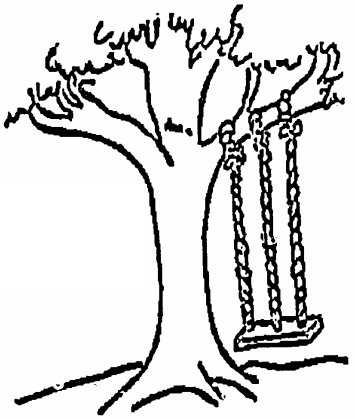
Why Model Driven Service Creation?

Support for service development from service idea down to running solution

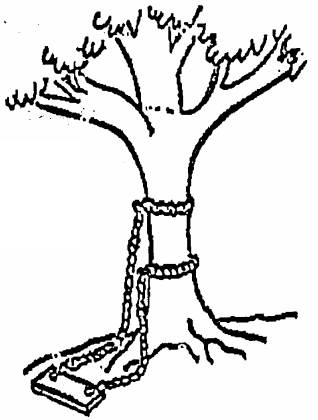
Why Model Driven Service Creation?



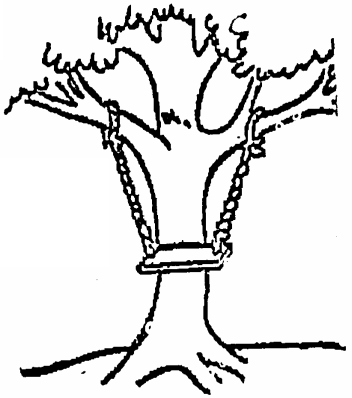
As Management requested it



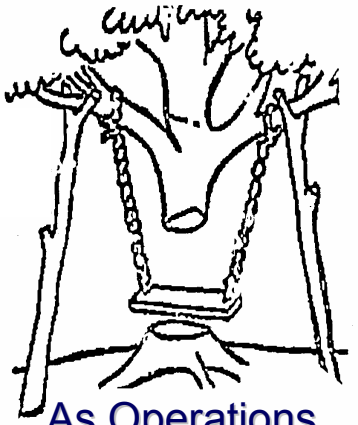
As the Project Leader defined it



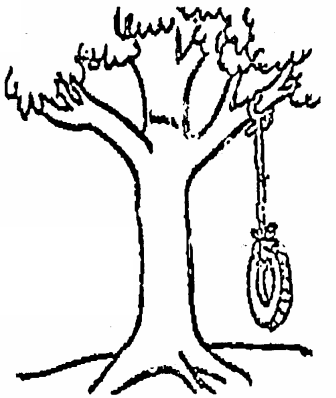
As Systems designed it



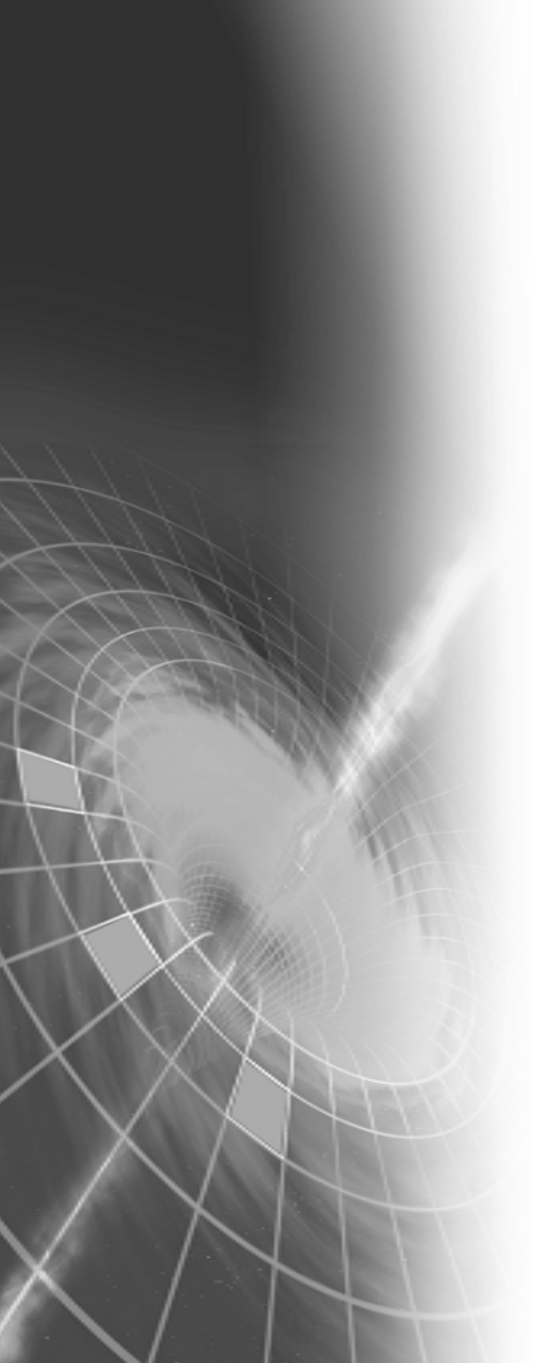
As Programming developed it



As Operations installed it

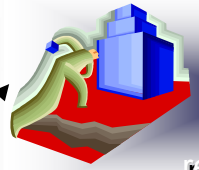


What the customer wanted



Our vision:

service creation tools for 3Gb services running in an environment that consistently connects the output of one with the input of another



service idea and requirements

technology independent service model

technology-independent service model

technology specific service model

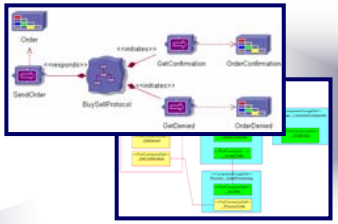
system-components

system integration tests and running system

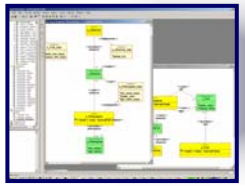
artifacts in an MDA inspired software engineering process

system development

automated artifact transfer between tools

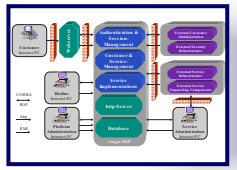


platform-independent Modeling tool



platform-specific Modeling tool

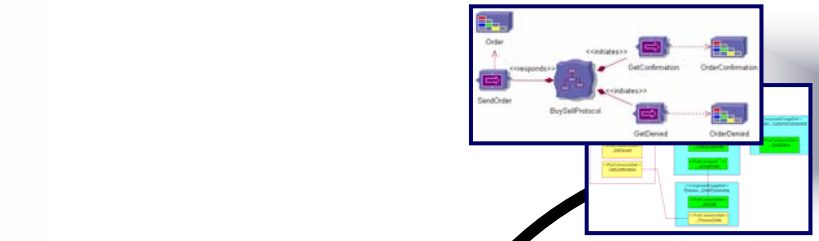
integrated-development environment



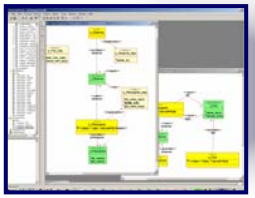
CORBA/Parlay for integration and operation

system operation

Our Tools in an Example Process

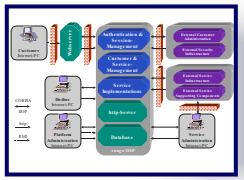


platform-independent modeling



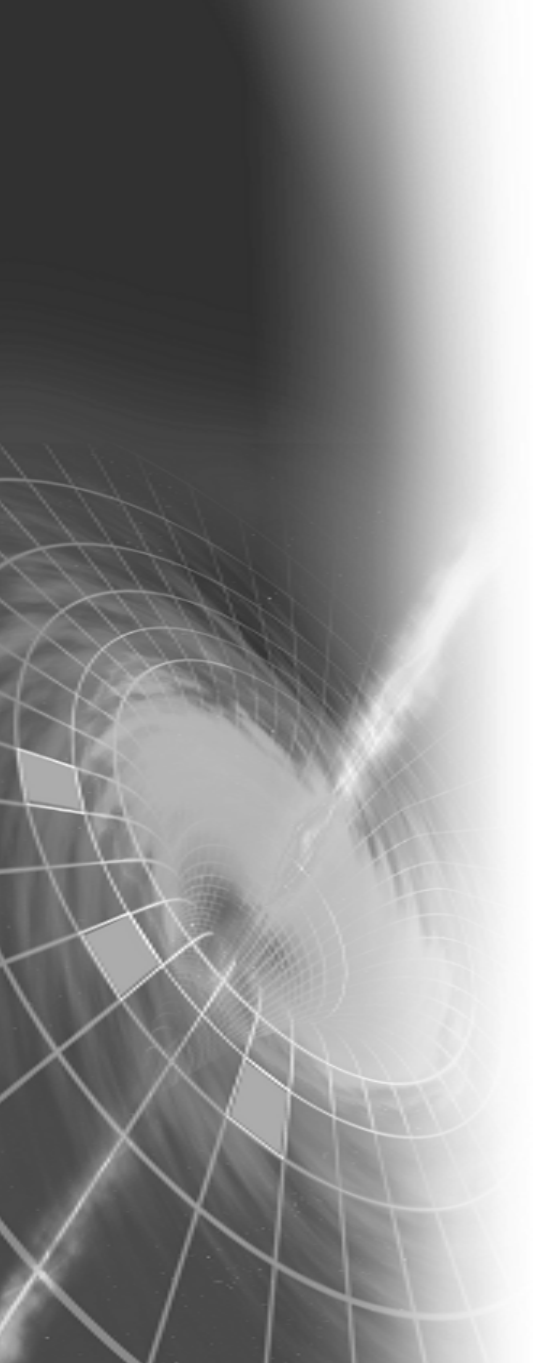
platform-specific modeling

integrated-development environment



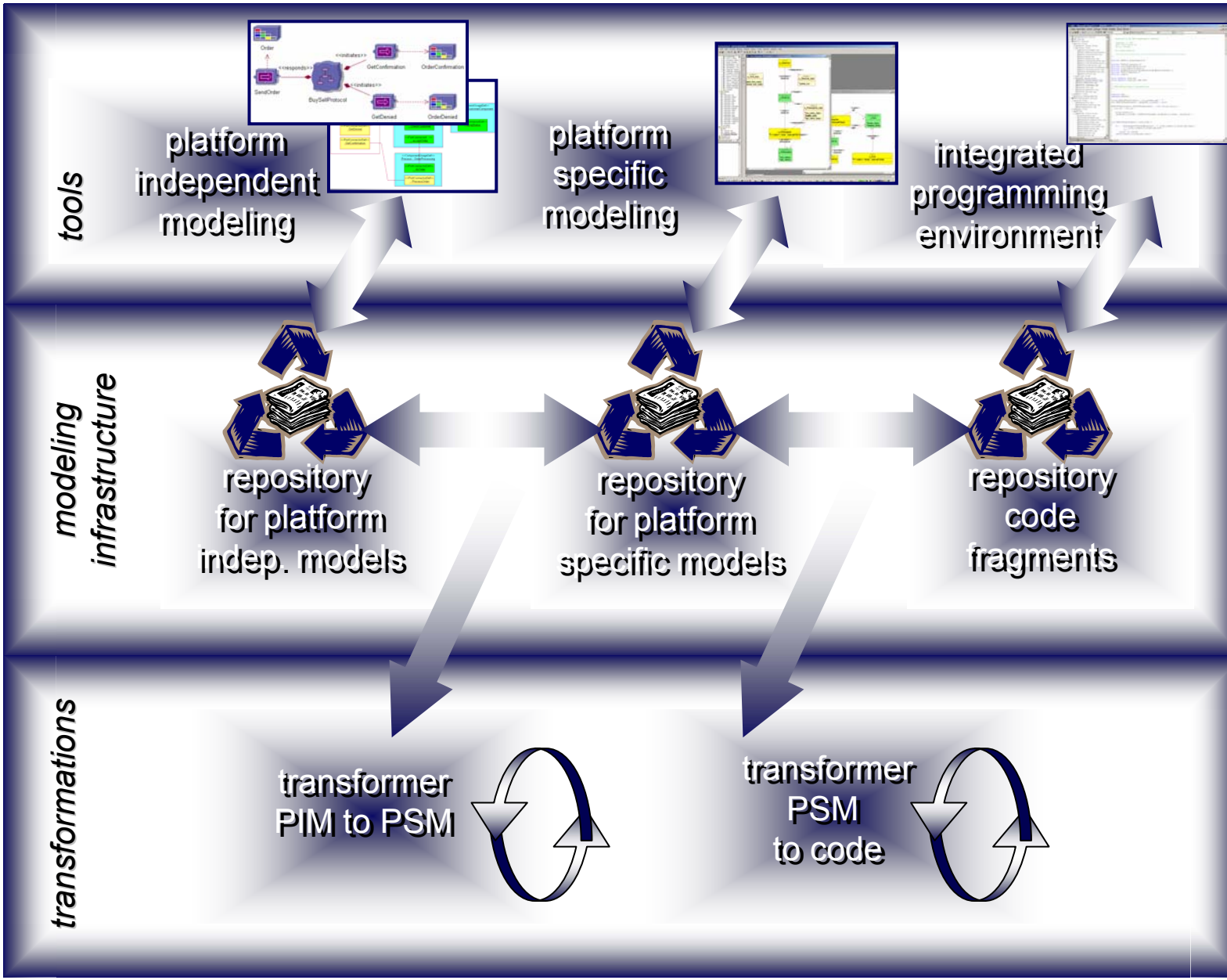
Service platform for integration and operation

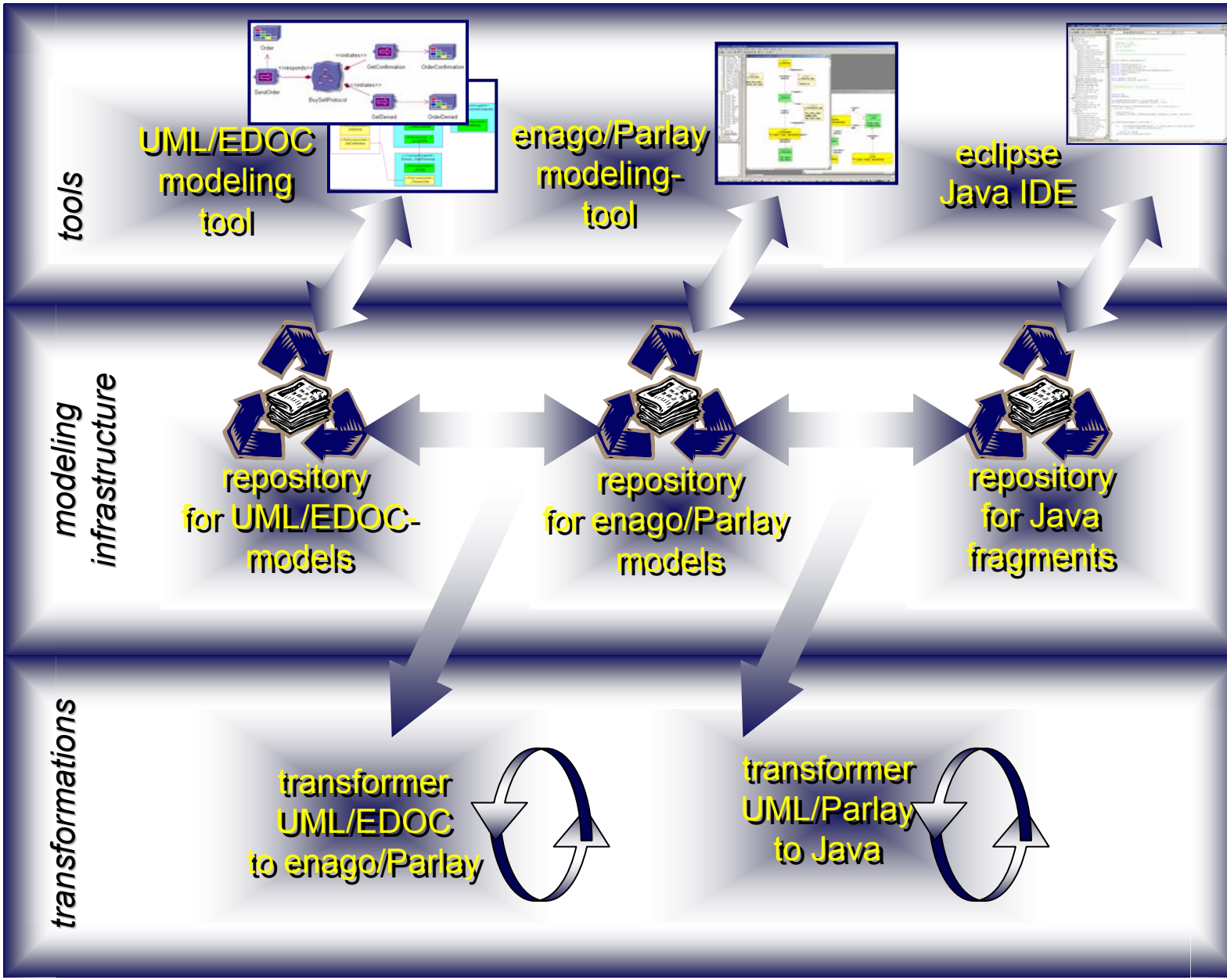
project run time

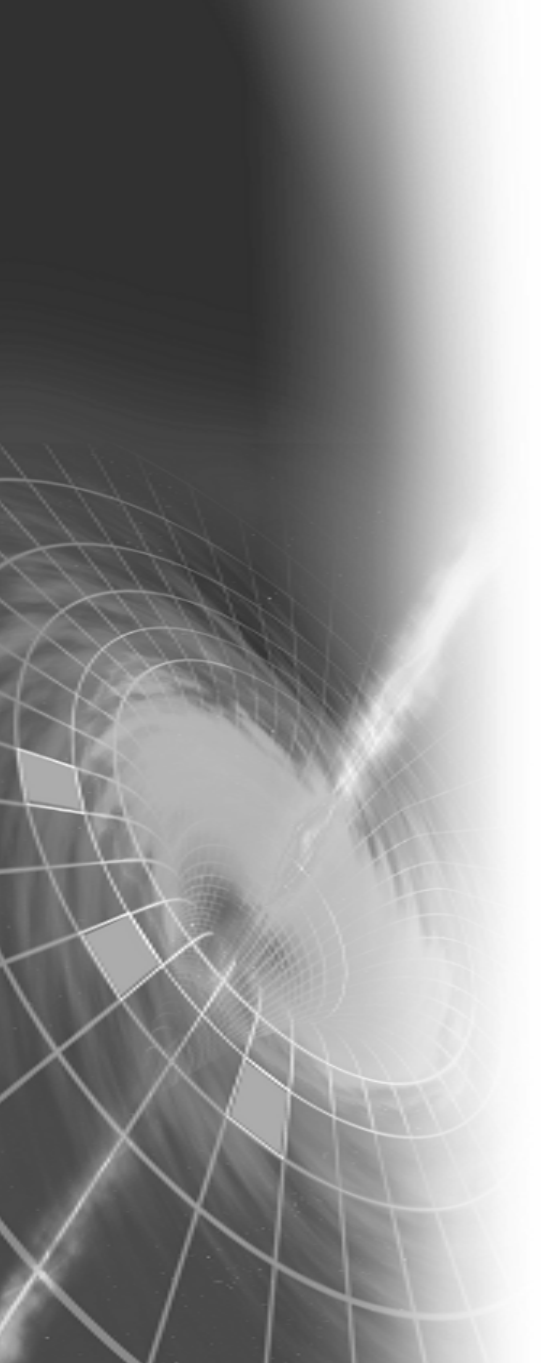


The approach to the vision:

The Big Picture

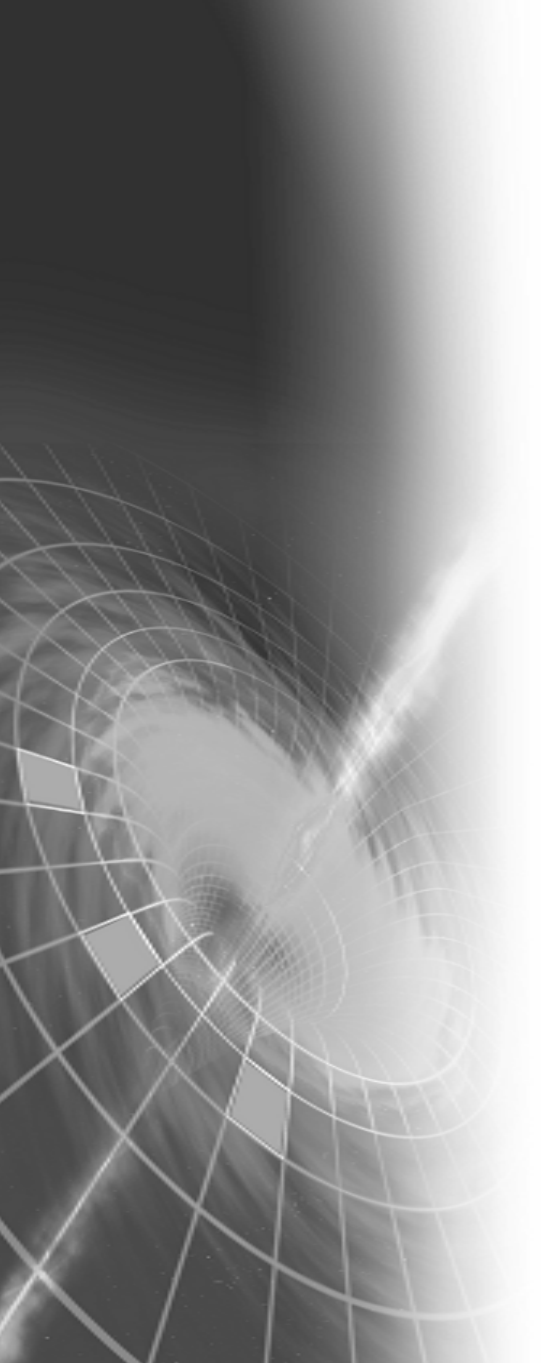

tools
platform independent modeling
platform specific modeling
integrated programming environment
modeling infrastructure
repository for platform indep. models
repository for platform specific models
repository code fragments
transformations
transformer PIM to PSM
transformer PSM to code


tools
UML/EDOC modeling tool
enago/Parlay modeling-tool
eclipse Java IDE
modeling infrastructure
repository for UML/EDOC-models
repository for enago/Parlay models
repository for Java fragments
transformations
transformer UML/EDOC to enago/Parlay
transformer UML/Parlay to Java



Current achievements:

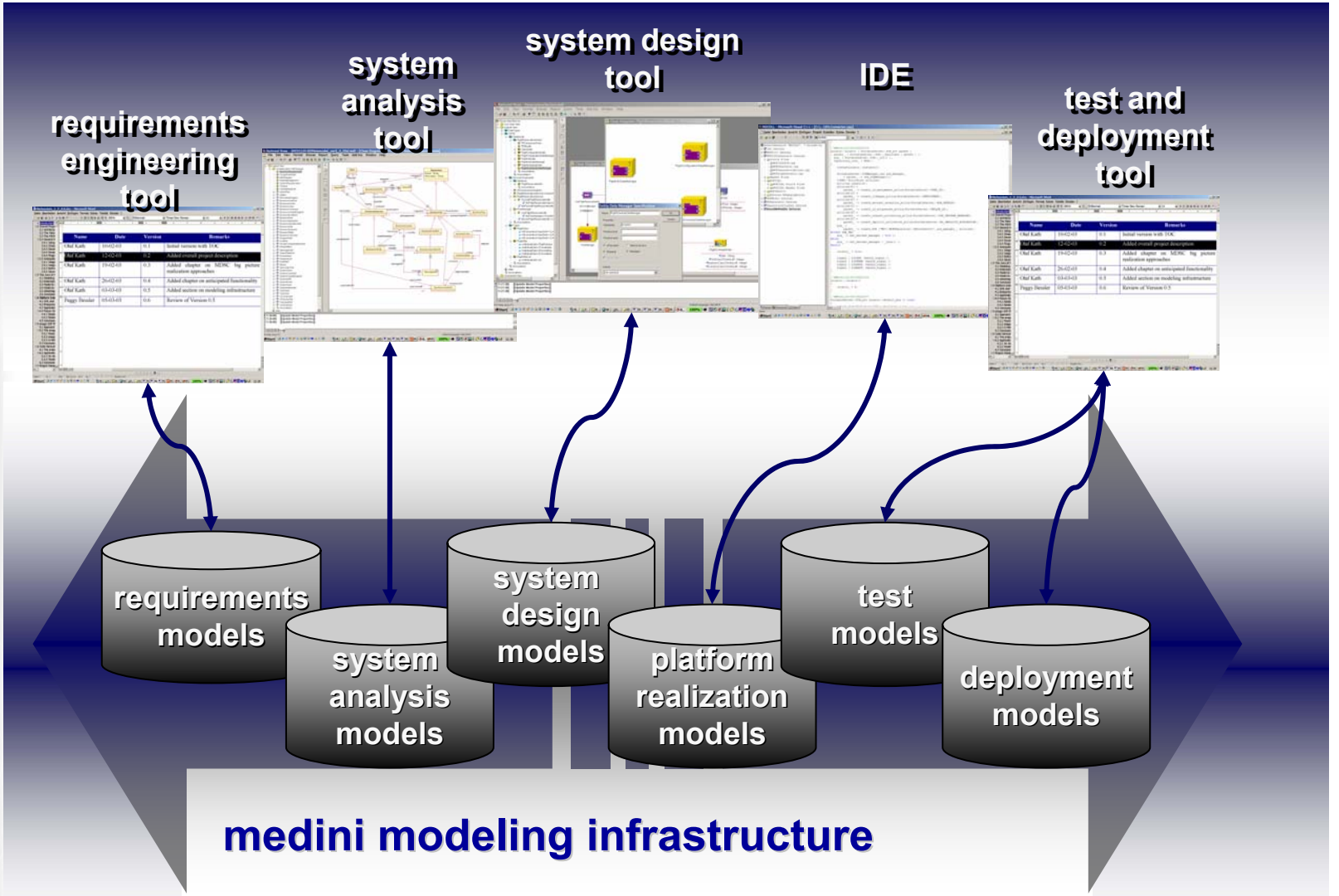
- *MOF and OCL based Meta-Tools*
- *modeling infrastructure for tool collaboration*
- *model transformers*
- *platform independent modeling tool based on UML/EDOC*
- *platform specific modeling tool based on UML/Parlay*
- *eclipse (IBM IDE) integration*



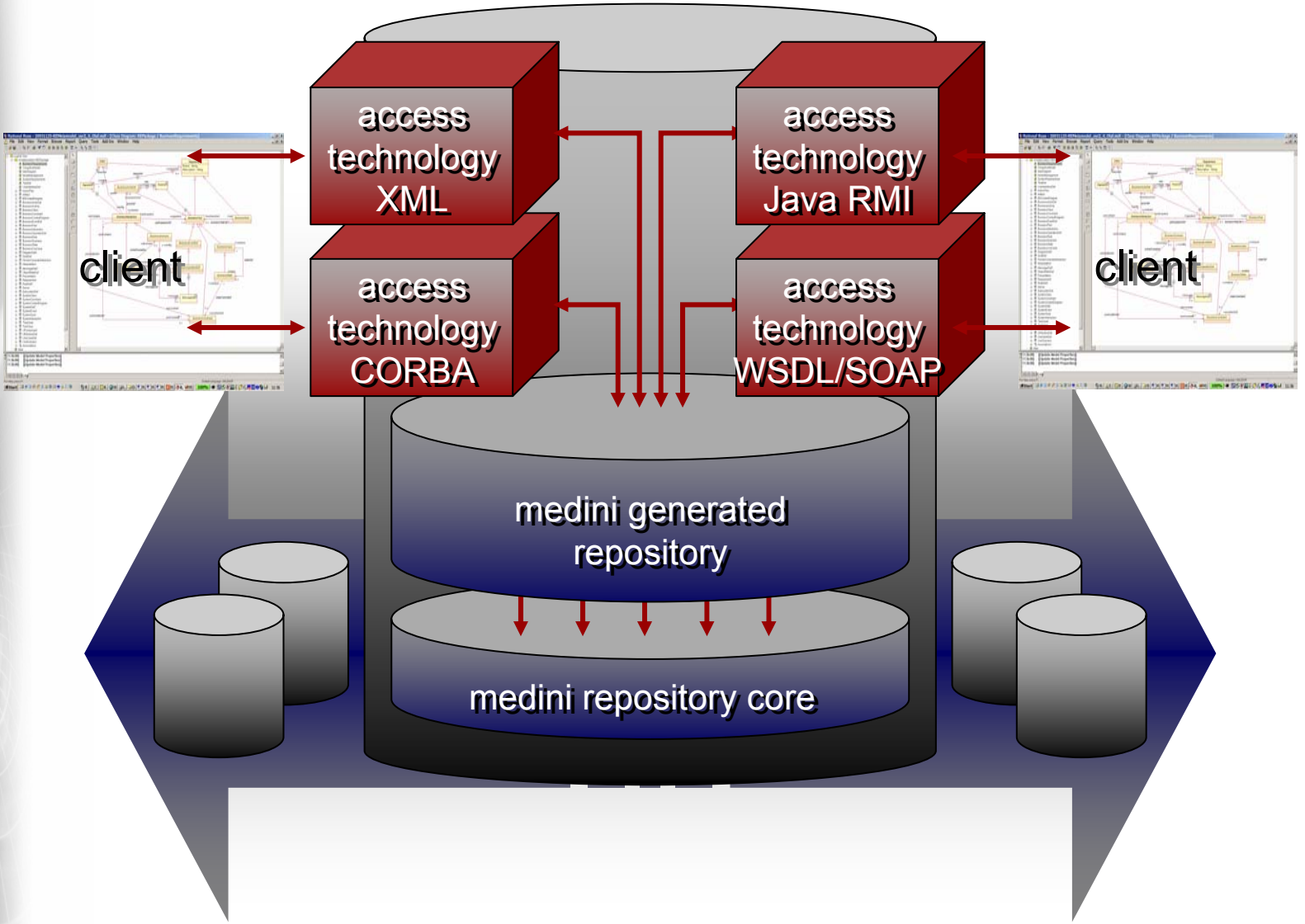
Current achievements:

- *MOF and OCL based Meta-Tools*
- *modeling infrastructure for tool collaboration*
- *model transformers*
- *platform independent modeling tool based on UML/EDOC*
- *platform specific modeling tool based on UML/Parlay*
- *eclipse (IBM IDE) integration*

System Modeling and Development Tools



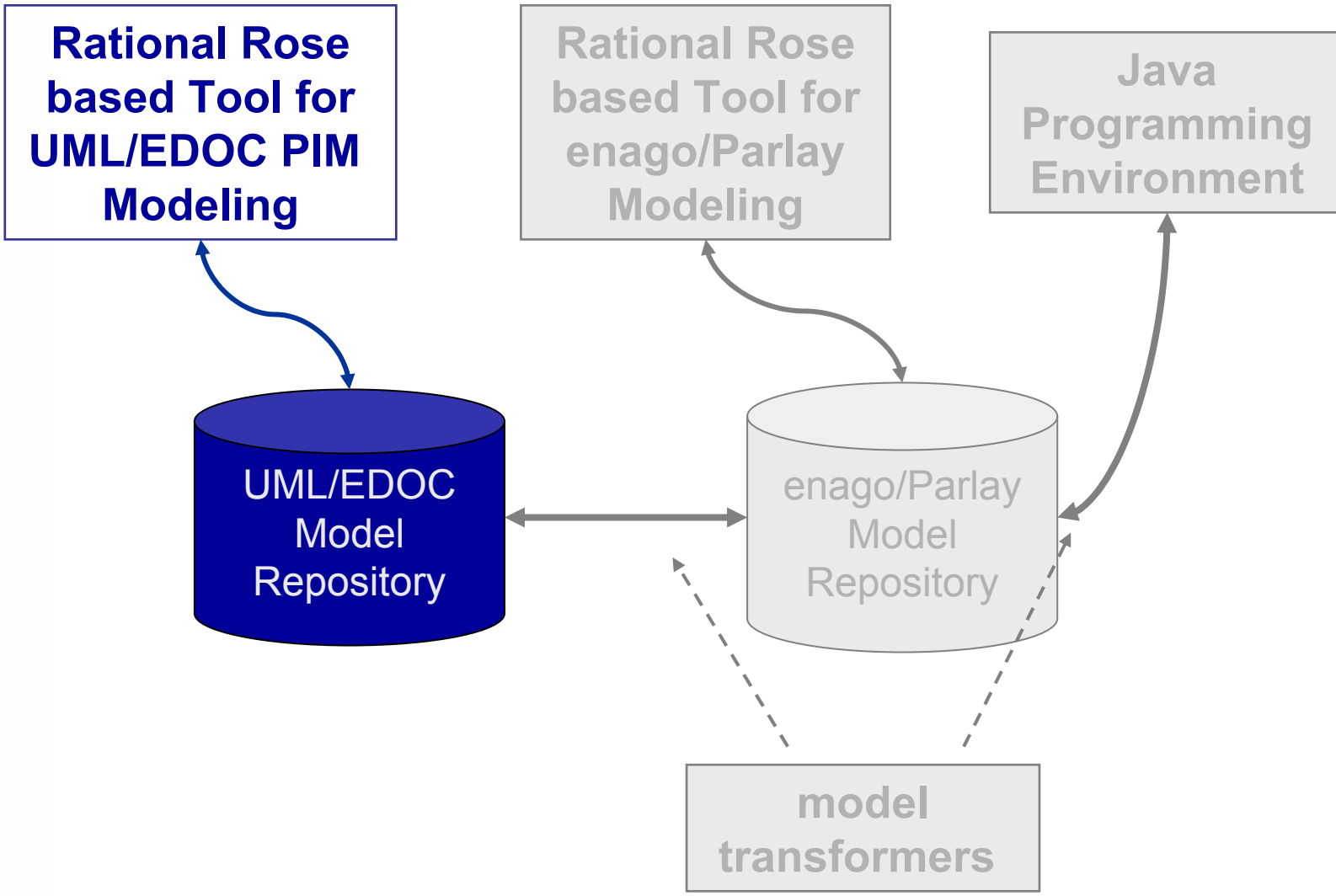
medini repositories



Current achievements:

- *MOF and OCL based Meta-Tools*
- *modeling infrastructure for tool collaboration*
- *model transformers*
- ***platform independent modeling tool based on UML/EDOC***
- *platform specific modeling tool based on UML/Parlay*
- *eclipse (IBM IDE) integration*

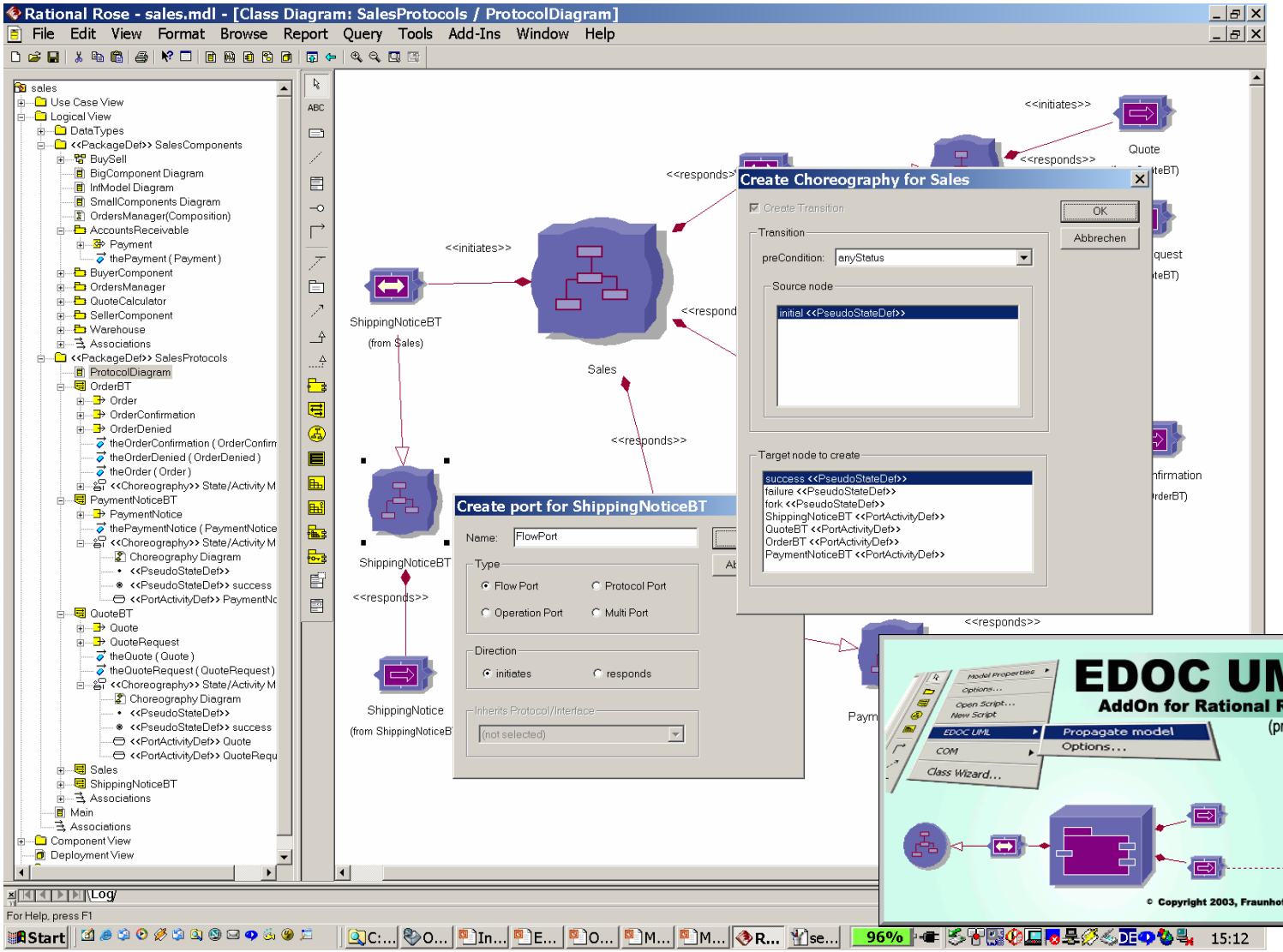
Platform Independent Modeling: Users Perspective



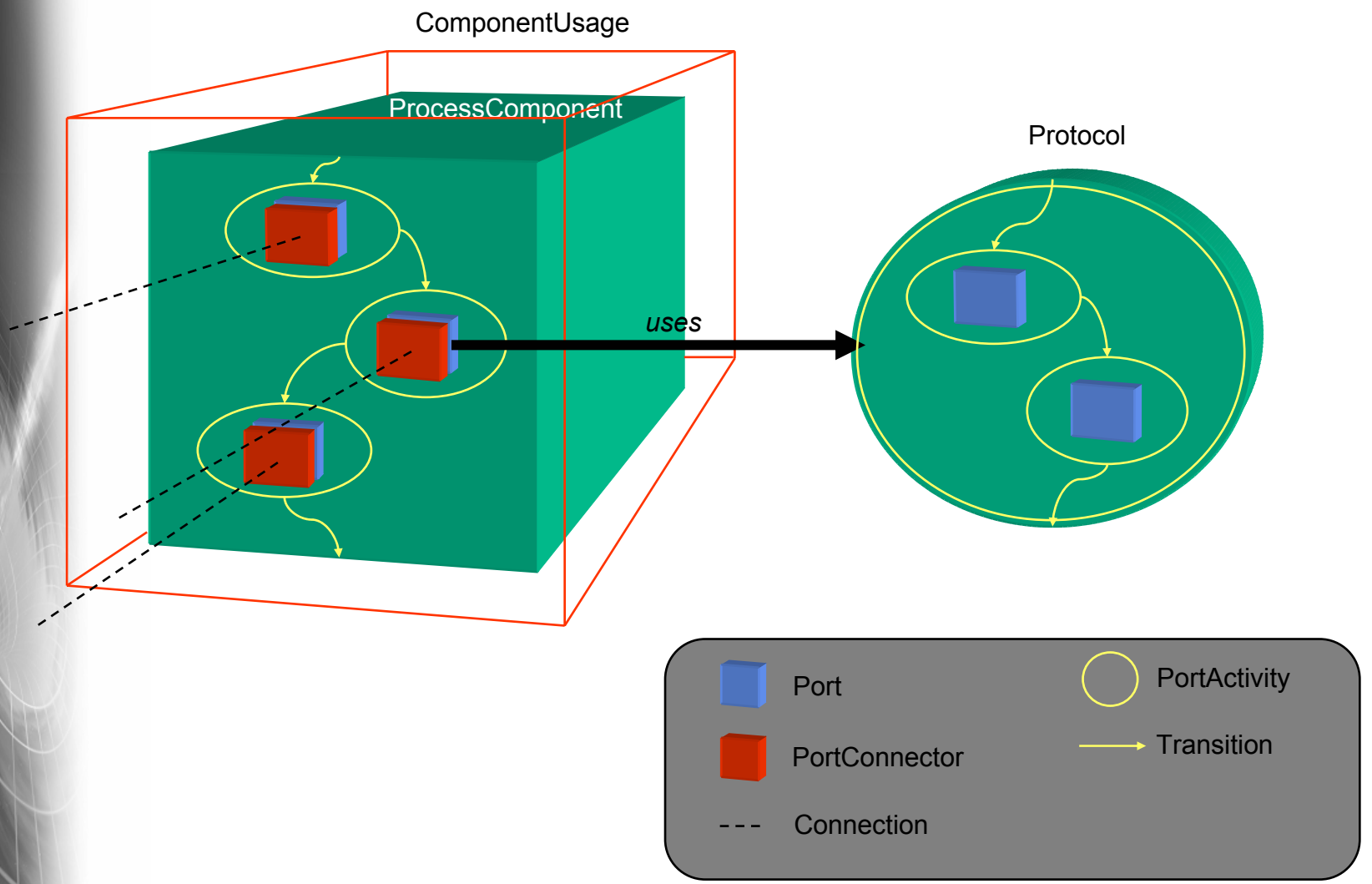
Platform Independent Modeling: EDOC Overview

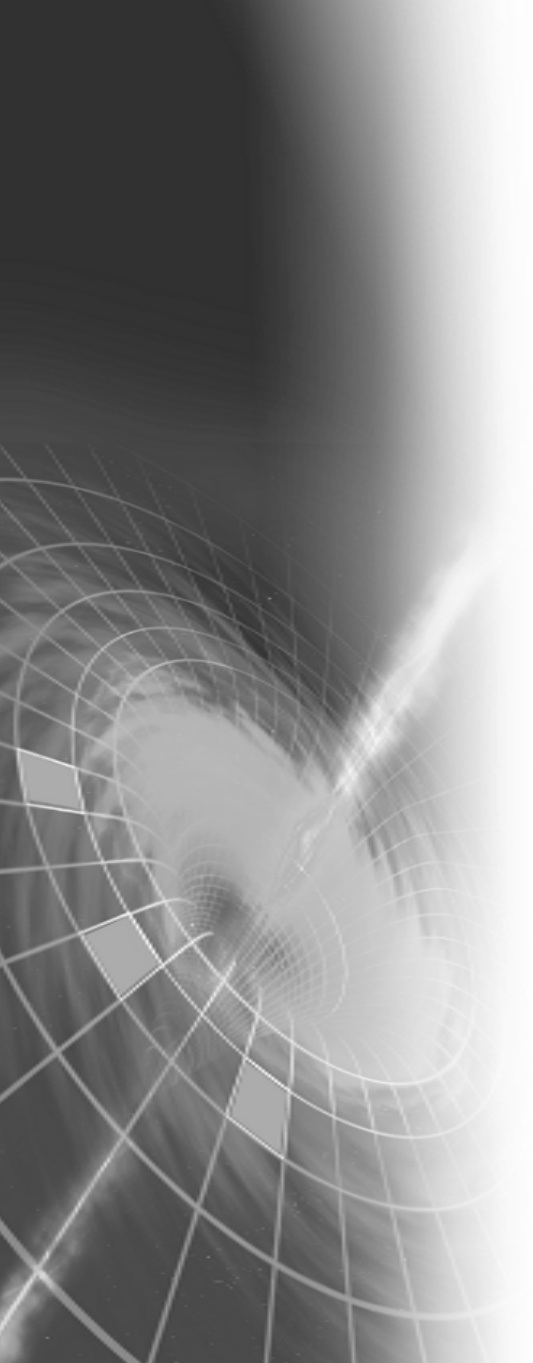
- Enterprise Distributed Object Computing is an OMG modeling standard for enterprise distributed systems
  - **Component based** definition of business process modules, definition of **component collaboration** using choreographies
  - definition of **information models** and their connection to business processes
  - definition of **messaging based interactions** between business components
- EDOC is being realized as an extension to the Rational Rose modeling tool
  - Specific modeling wizards, dialogs, …

Platform Independent Modeling: Our EDOC Front-End for Rational Rose



EDOC Choreography

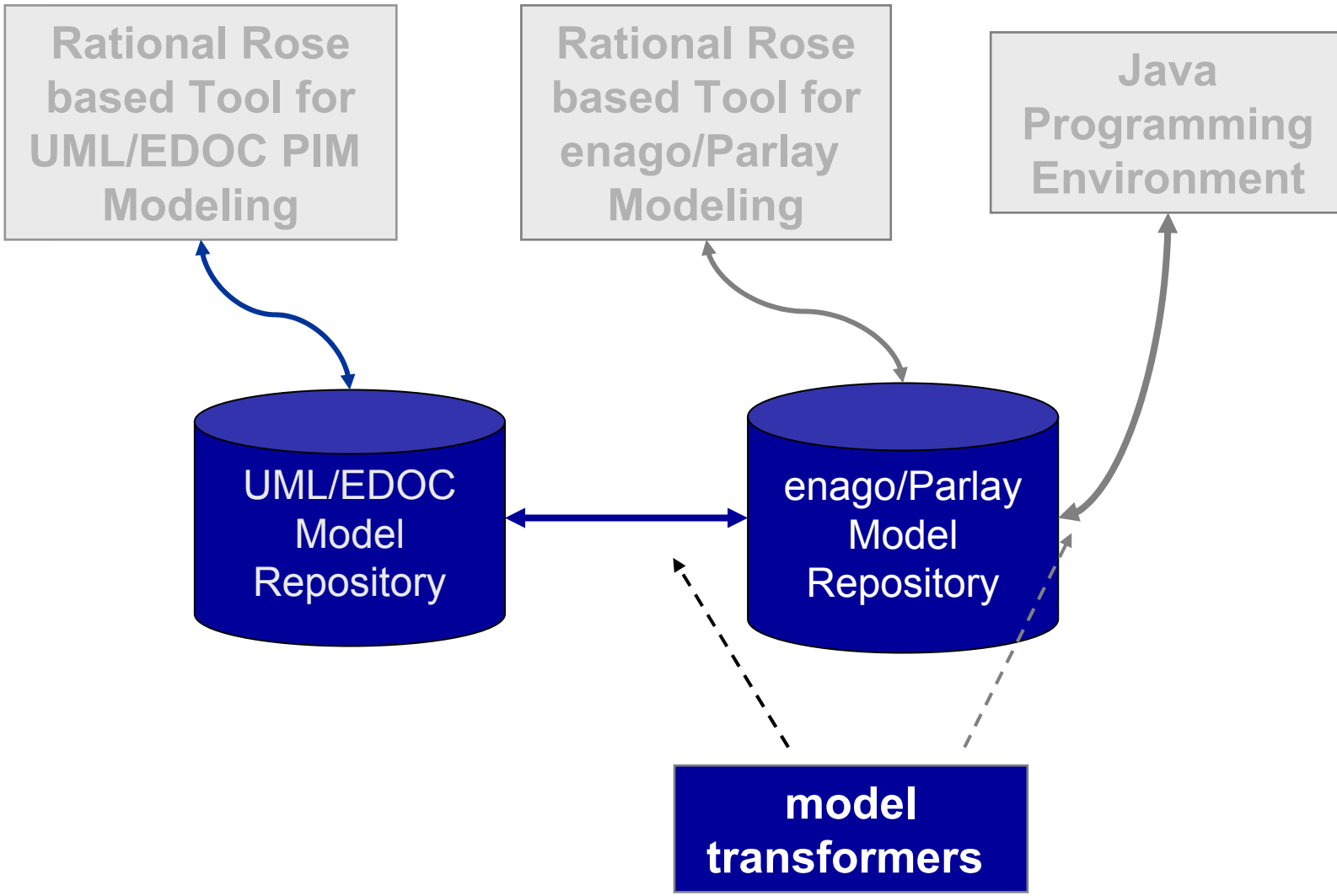




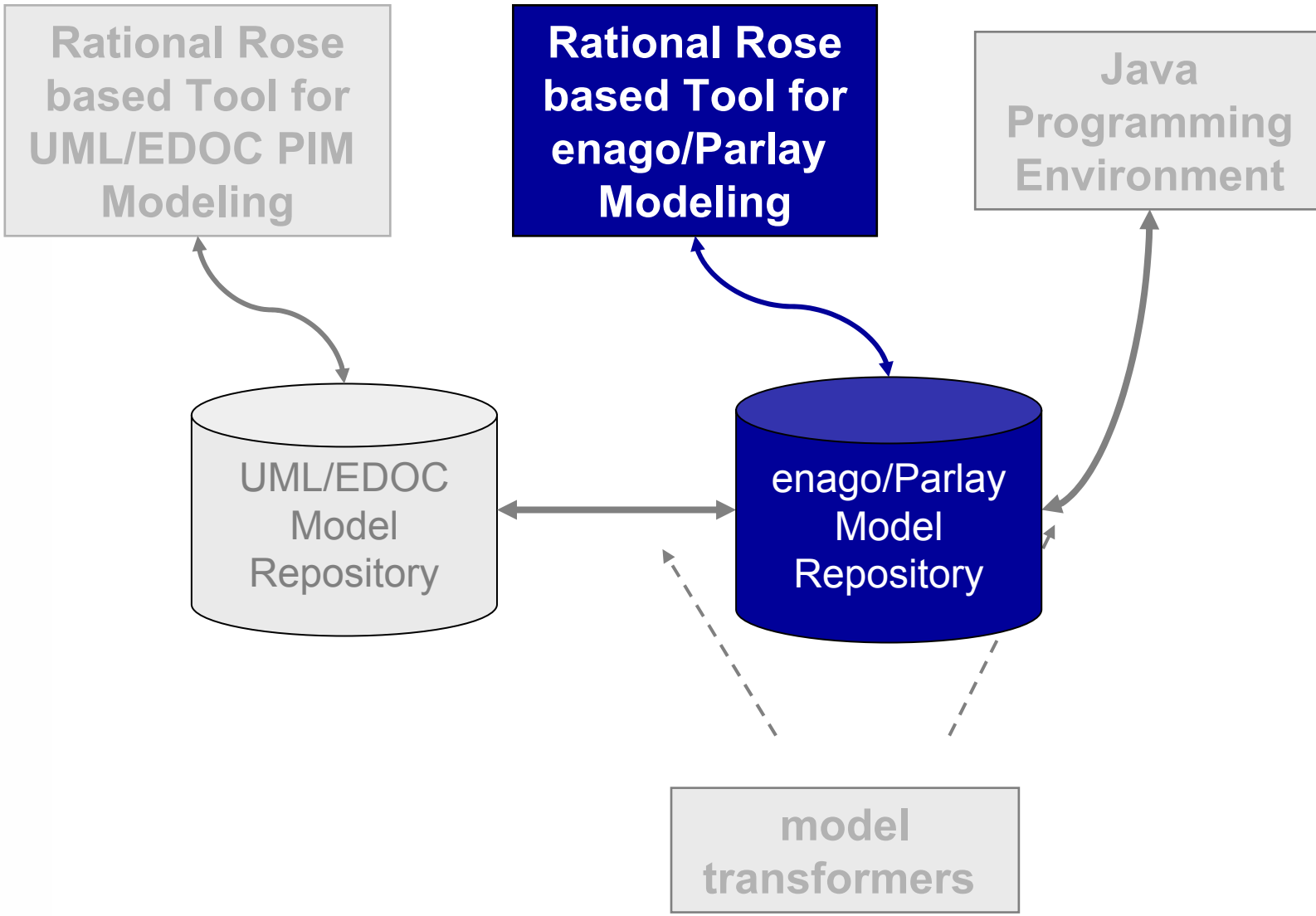
Current achievements:

- *MOF and OCL based Meta-Tools*
- *modeling infrastructure for tool collaboration*
- *model transformers*
- *platform independent modeling tool based on UML/EDOC*
- ***platform specific modeling tool based on UML/Parlay***
- *eclipse (IBM IDE) integration*

Platform Specific Modeling : PIM to PSM Transformation



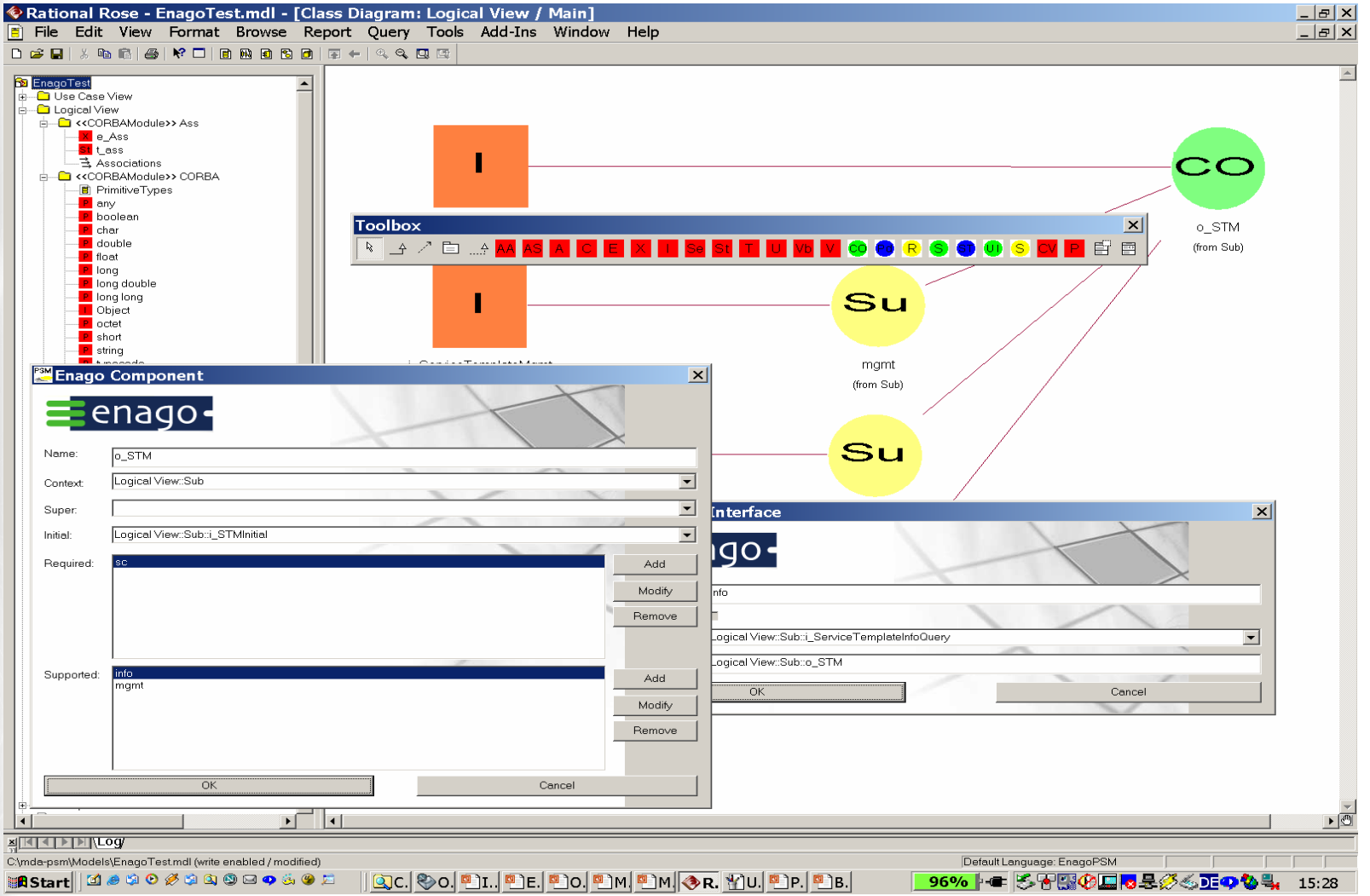
Platform Specific Modeling: Users Perspective



Platform Specific Modeling: The enago Platform

- enago (product of IKV^{++} Technologies AG) is a **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**
- Modeling support for enago is being realized as an extension to the Rational Rose modeling tool
  - again specific modeling wizards, dialogs, …

Platform Specific Modeling: Our enago Front-End for Rational Rose



Behavior modeling (1)

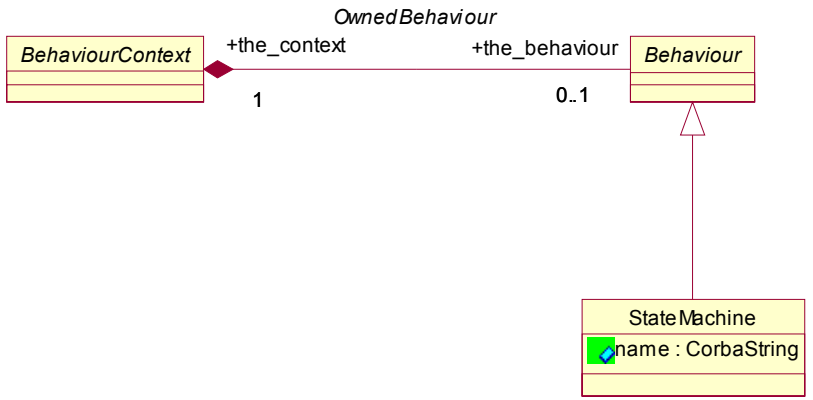
- **enago structural modeling comprises:**
  - Services
  - Components
  - CORBA interfaces …
- **Platform independent modeling supports choreographies**
  - Model the behavior of the structural concepts
  - Describe the interactions between the model elements

Behavior modeling (2)

- **enago behavior modeling**
  - The behavior descriptions are derived from the choreographies of the PIM
  - Refinement and extension of the resulting model with enago specific aspects
  - Description of the interaction between services and enago components
  - Interaction via CORBA interfaces and the offered operations

Behavior Context (1)

- Behavior is modeled using state machines
- A Behavior is always defined in a specific context (BehaviorContext)

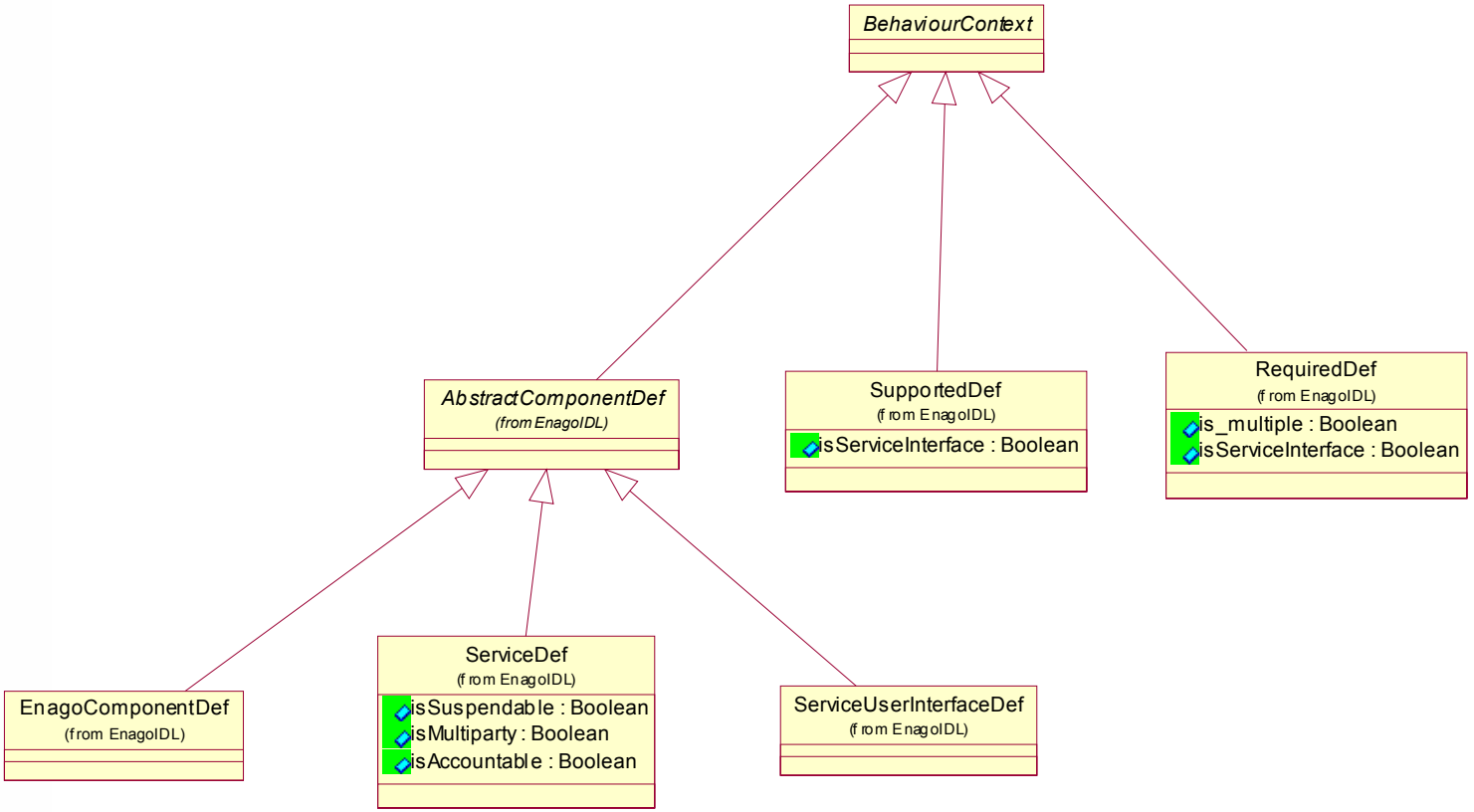


Behavior Context (2)

- **Behavior contexts in enago:**
  - Services
  - enago components
  - Service user interfaces

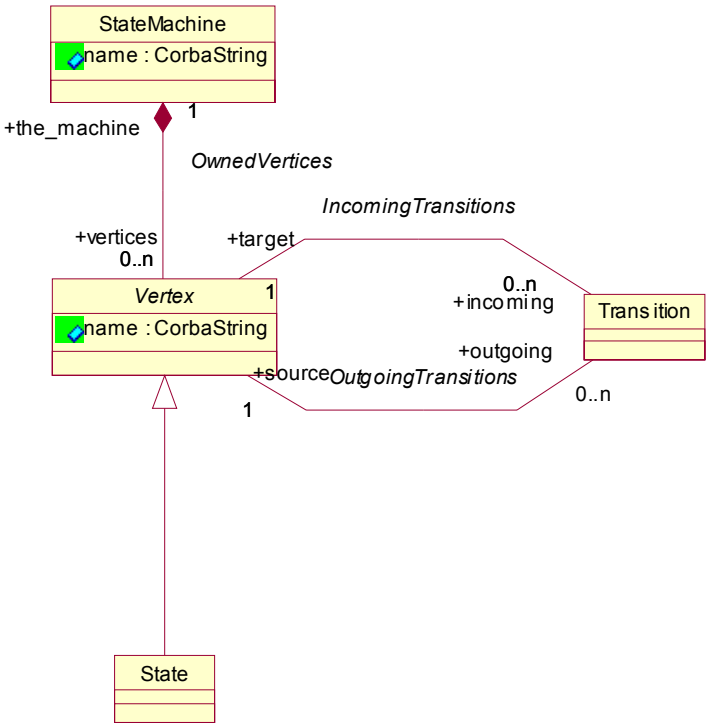
- **More detailed modelling**
  - Supported interfaces
  - Required interfaces

Behavior Context (3)



States and Transitions

- **Behavior description by means of states and transitions**
- **State:**
  - Condition during the lifetime of an object
  - Perform actions
  - Wait for an event
  - Outgoing transitions
  - Incoming transitions



Transition

- **Represents a change from an originating state to a successor state**

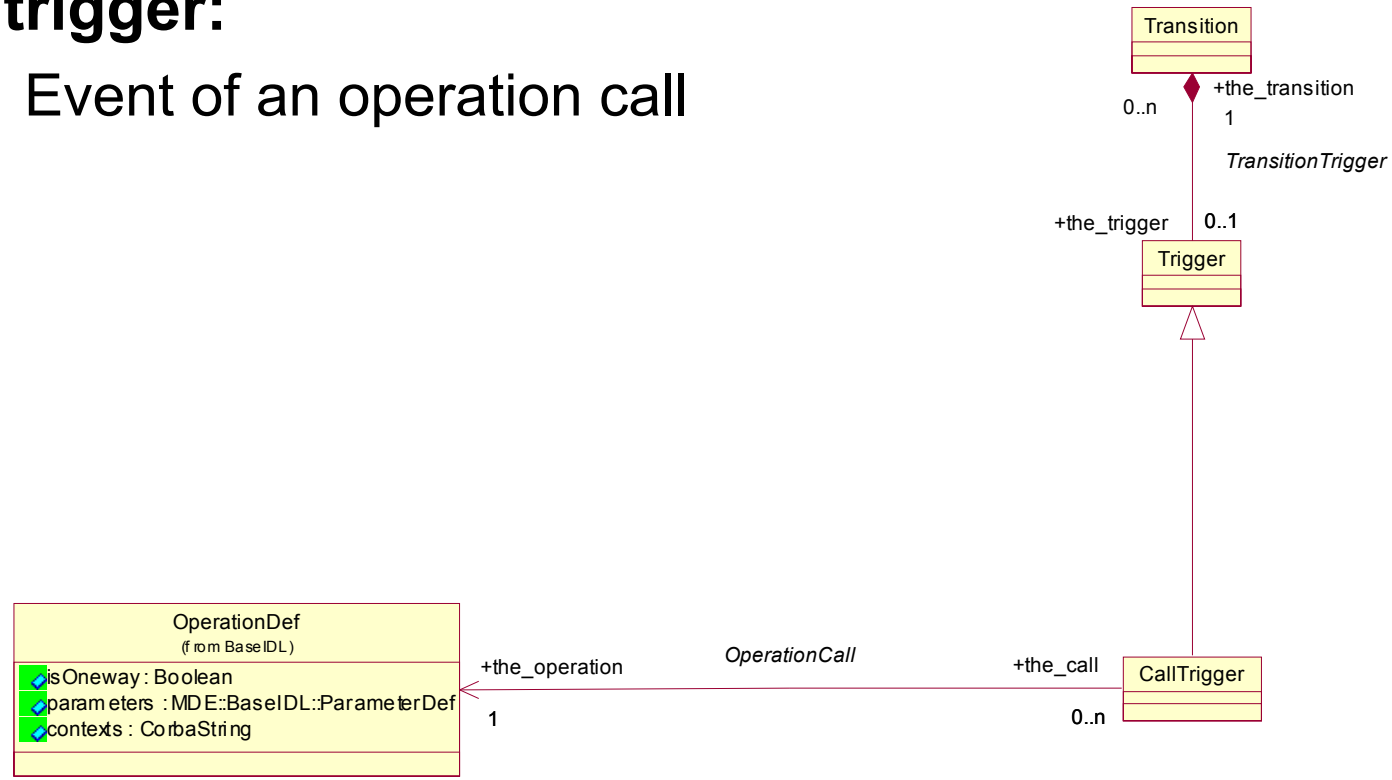
- **Two different kinds of transitions:**
  - Automatic transition:
    - Occurs when the activities of the originating state completes
  - Non-automatic transition:
    - Occurs when a named event is received by the system
    - Named events are especially operations of the CORBA interfaces

Trigger

- **Non-automatic transitions will be modeled by triggers**
- **A trigger is always associated with a transition**

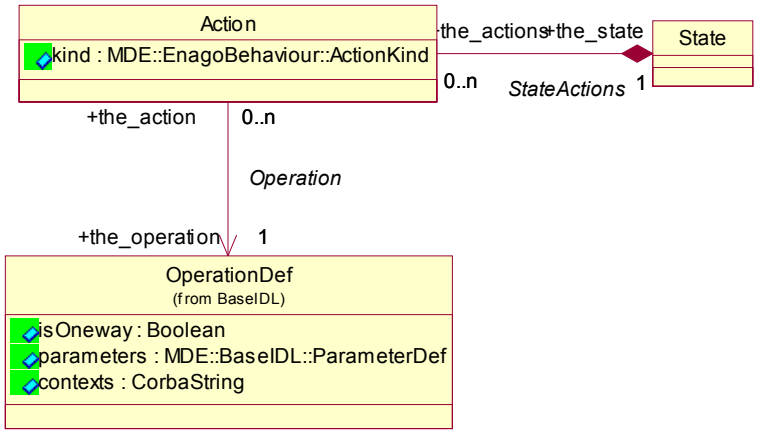
Call trigger:

– Event of an operation call



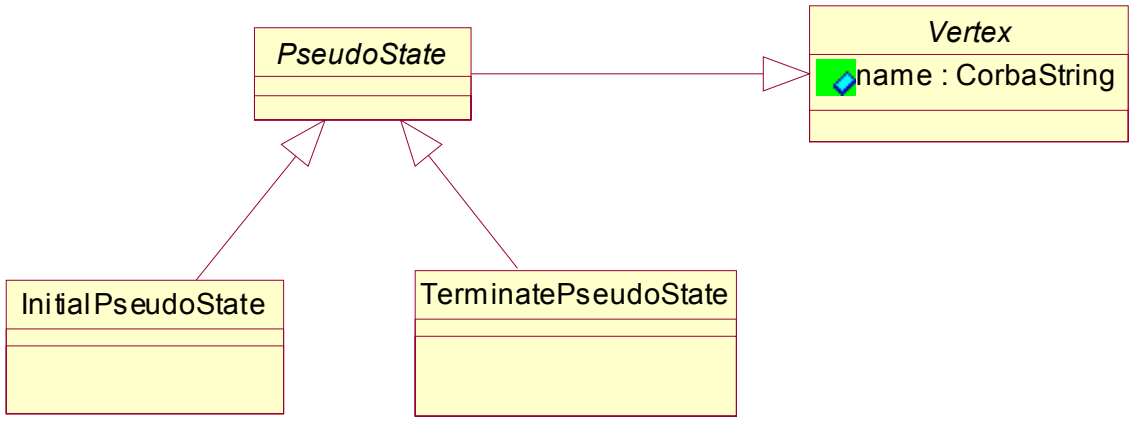
Actions

- **If a state is entered a number of actions can be performed**
- **Actions are ordered**
- **Actions result in operation calls**
- **Three kinds of actions:**
  - Entry actions
  - Do actions
  - Exit actions



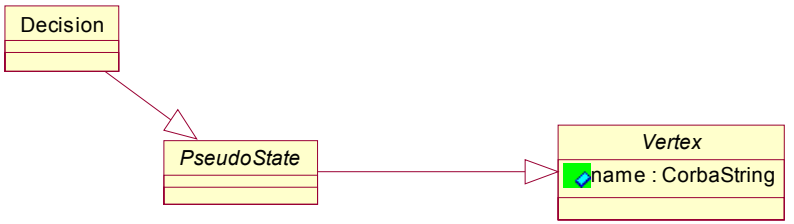
Pseudostates

- **One initial pseudo state**
- **One or more terminate pseudo states**
  - Execution of the state machine terminates
  - Success or failure result

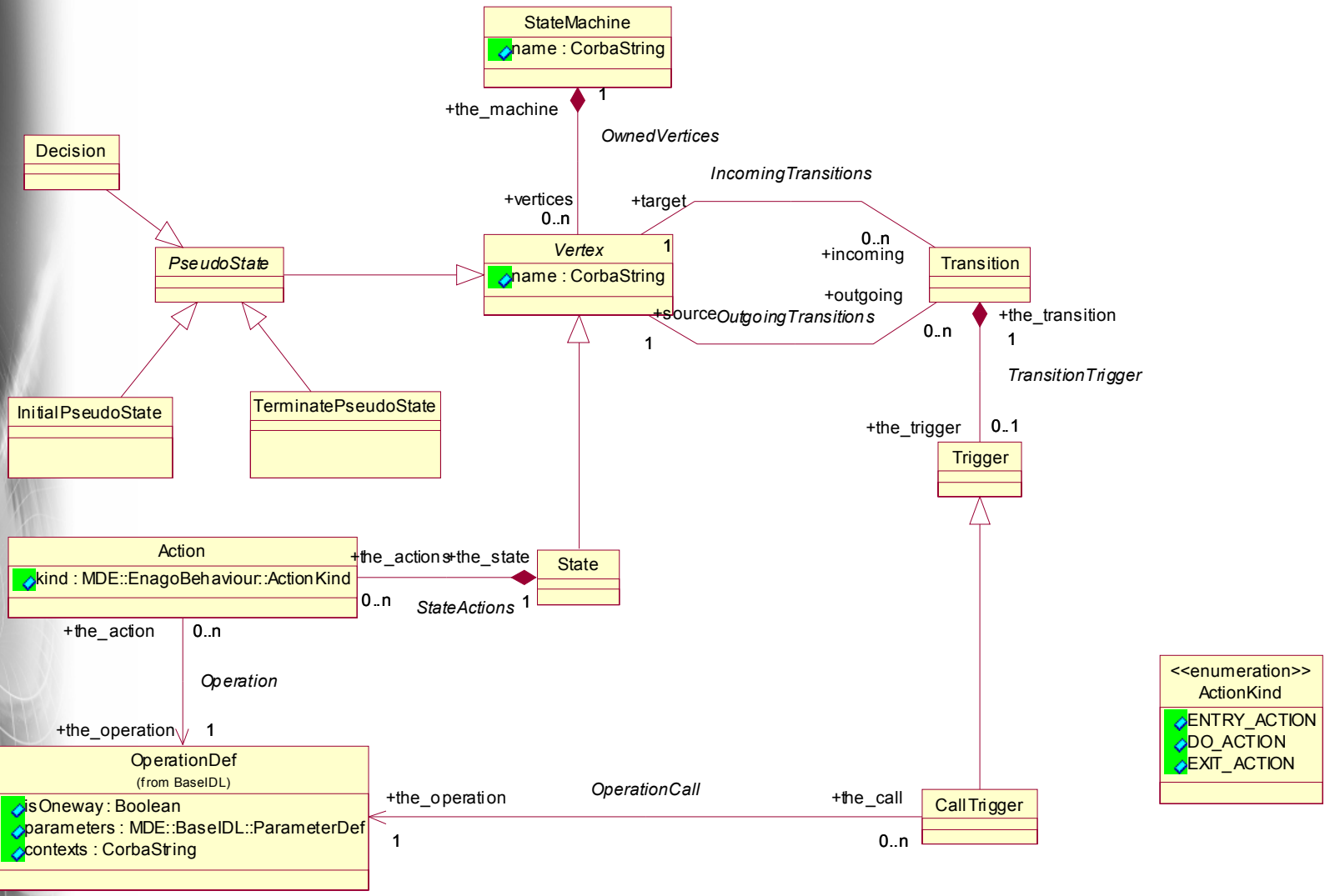


Decision

- **A decision is a special state**
  - Contains no actions
  - Contains a decision condition
  - Depending on the decision condition, one of the outgoing transitions is used

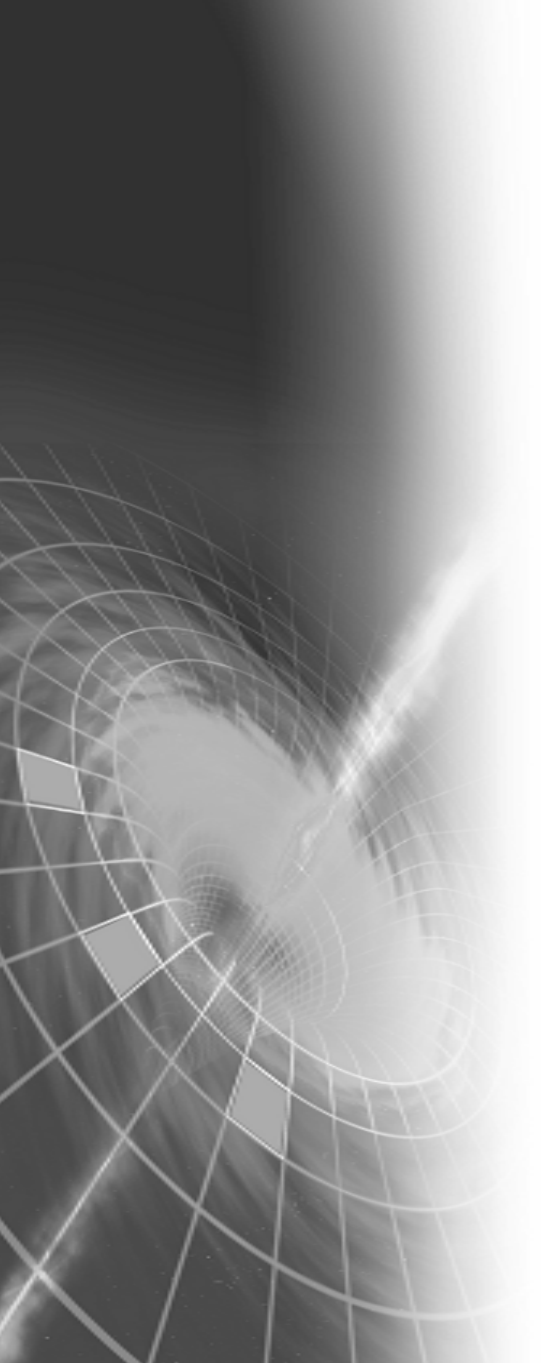


The meta model



Platform Specific Modeling: enago and Parlay

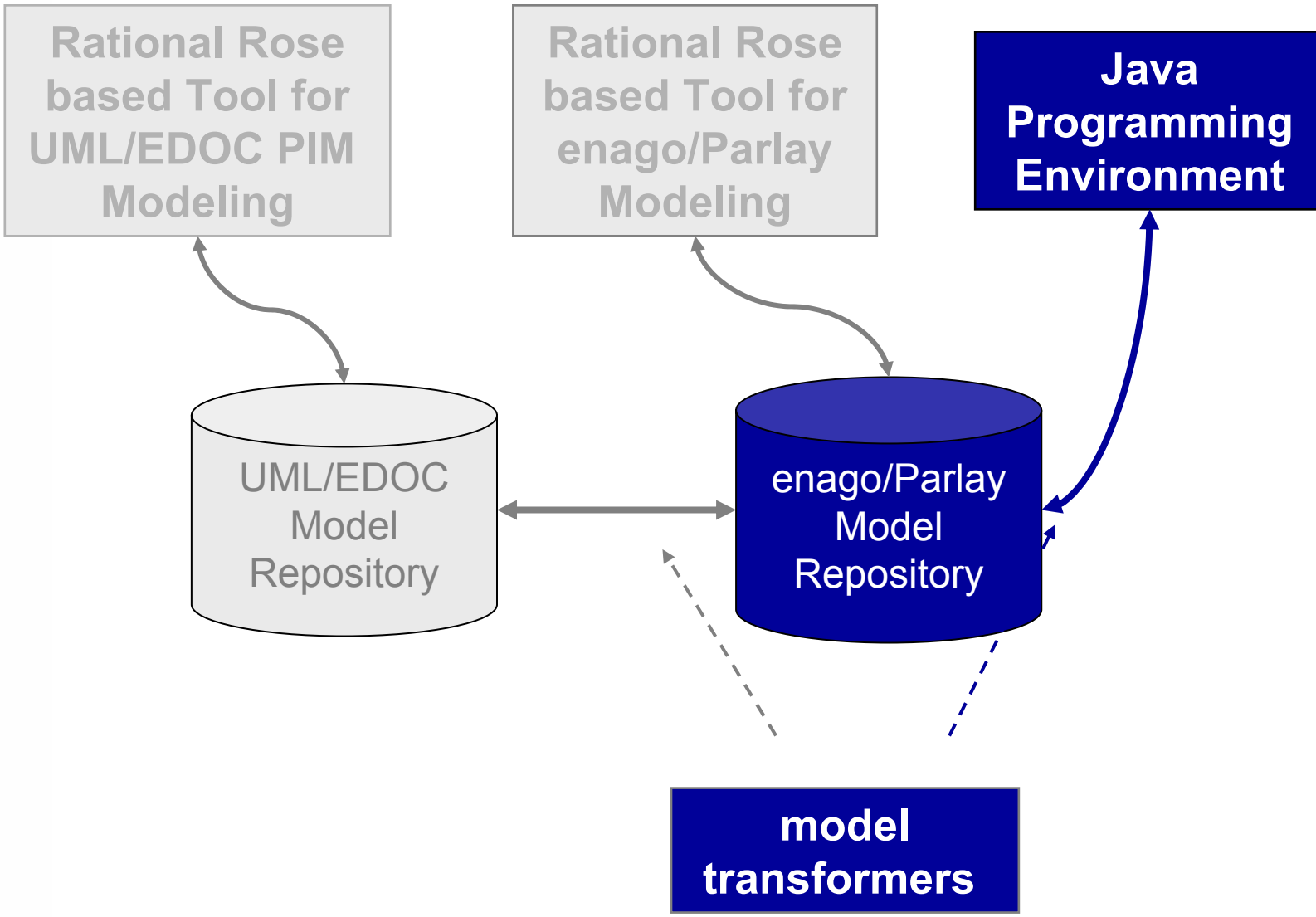
- Parlay services are visible as a library of models
  - Containing structure and behavior of the Parlay API and its implementation in the Parlay platform of FhI FOKUS
- A transformer from EDOC to enago + Parlay
  - is parameterized by the modeler, and maps interactions between EDOC components to Parlay services usages
  - both structure and behavior of EDOC models are mapped to such enago/Parlay models



Current achievements:

- *MOF and OCL based Meta-Tools*
- *modeling infrastructure for tool collaboration*
- *model transformers*
- *platform independent modeling tool based on UML/EDOC*
- *platform specific modeling tool based on UML/Parlay*
- ***eclipse (IBM IDE) integration***

eclipse Integration: Users View



eclipse Integration: Purpose and Achievements

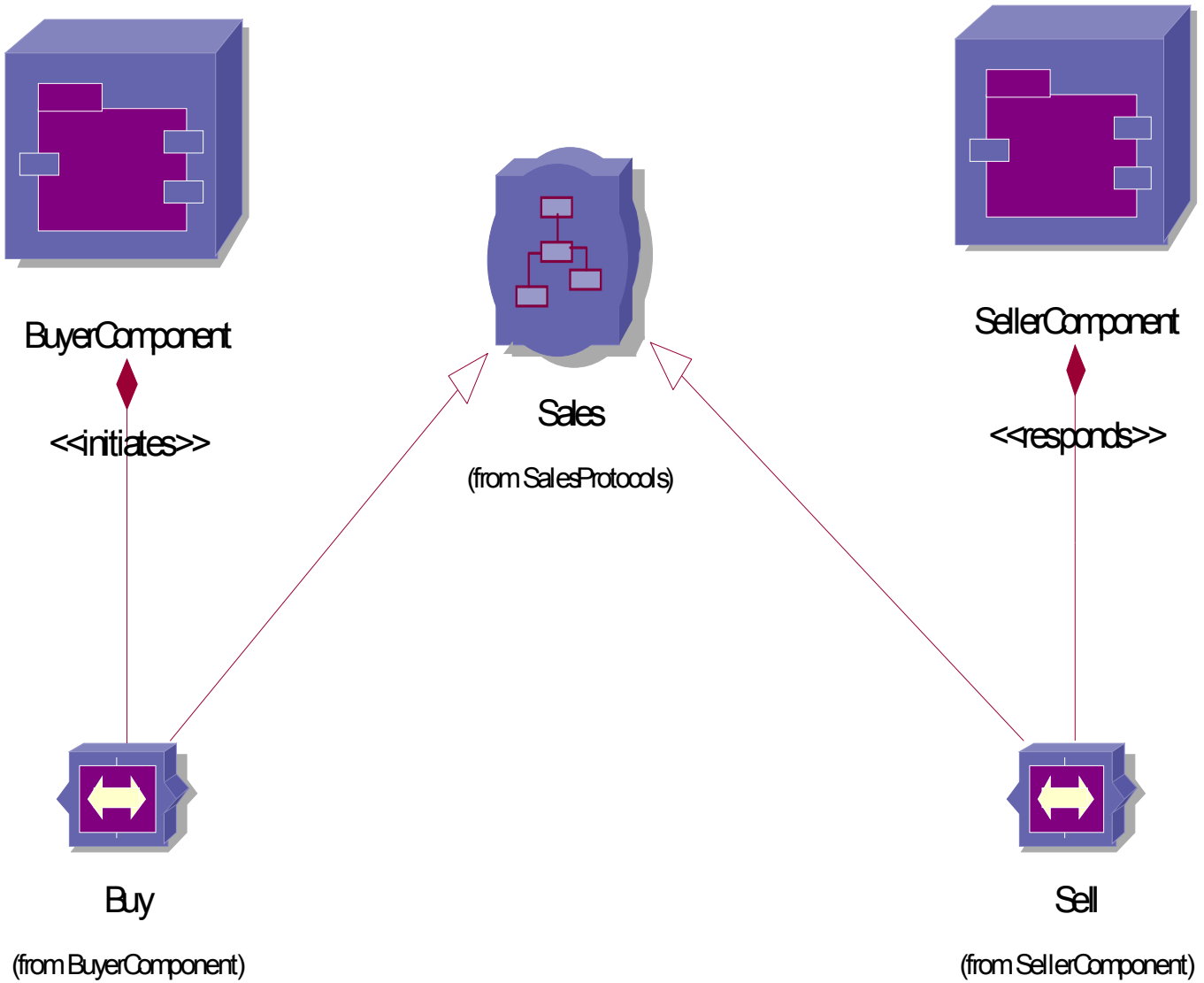
- Purpose: **automatic generation** of Java programming language code out of enago/Parlay models
  - The modeler refines the enago/Parlay model of collaborating services
  - The services are being realized by software components
  - These components are implemented in the Java language
- eclipse (IBM's open source development platform) provides a Java integrated development environment
- We use **eclipse as target for code generation**

Results enable

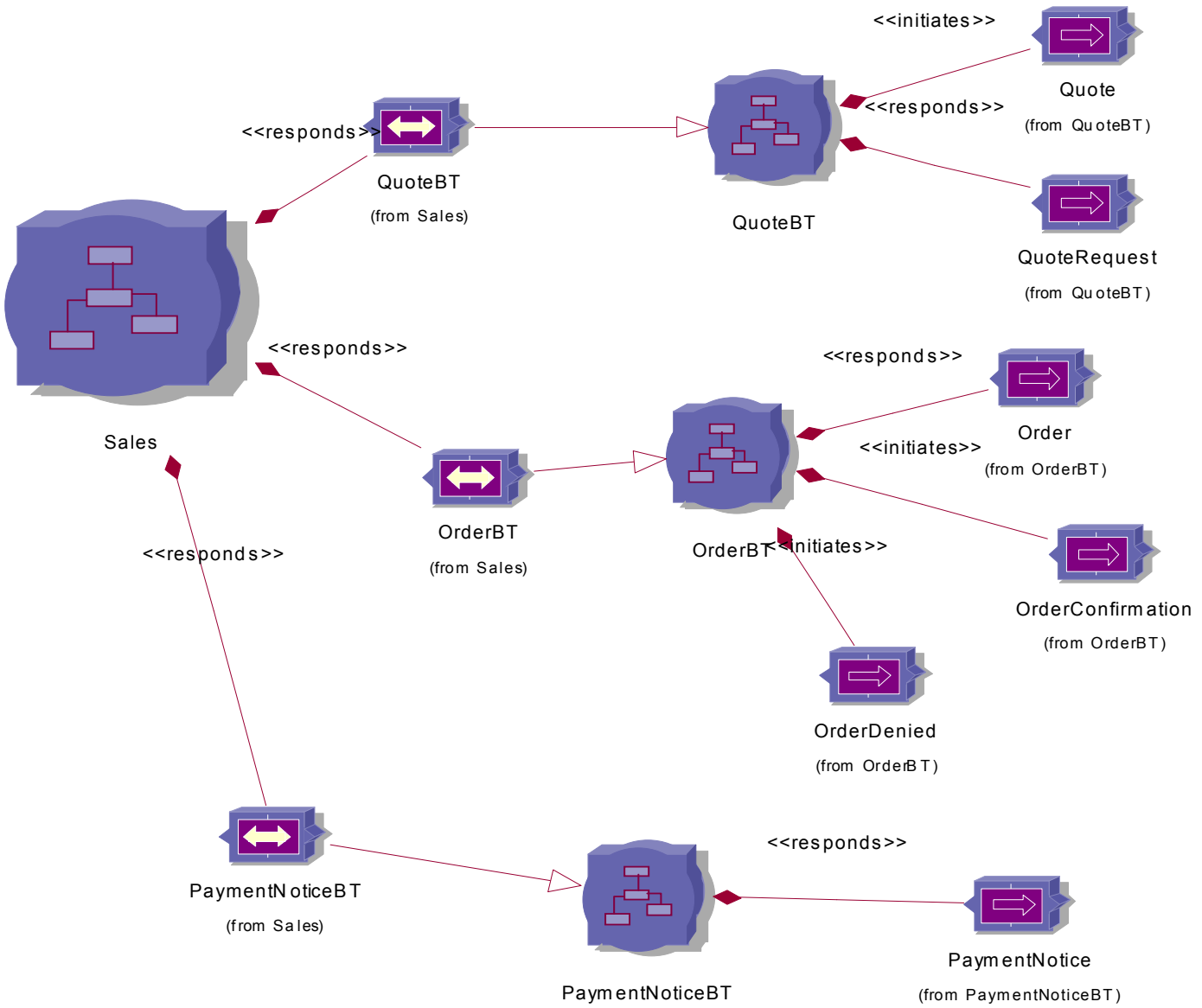
- platform independent definition of business processes
  - with well defined semantics, based on EDOC
  - supported by graphical modeling tools
- platform specific definition of enago/Parlay services
  - with well defined semantics
  - Supported by graphical modeling tools and programming environments
- transformations between platform independent and platform specific models as well as code generation
  - with well defined semantics, based on formal definition of model transformation rules
- a modeling infrastructure, supporting
  - consistency and traceability of models
  - the integration of different modeling languages and techniques into one environment

Example

The Buyer-Seller-Example

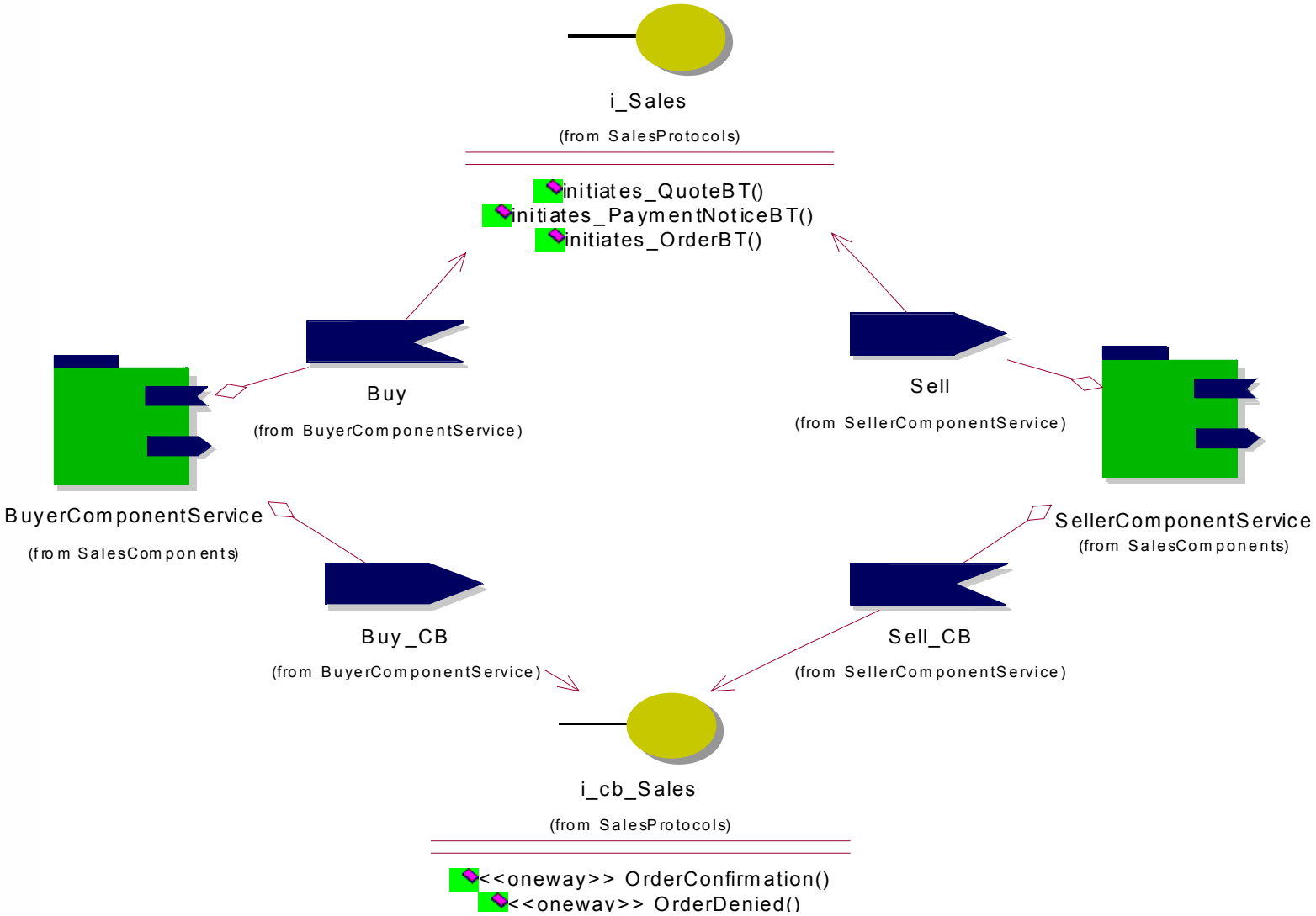


The Buyer-Seller-Example



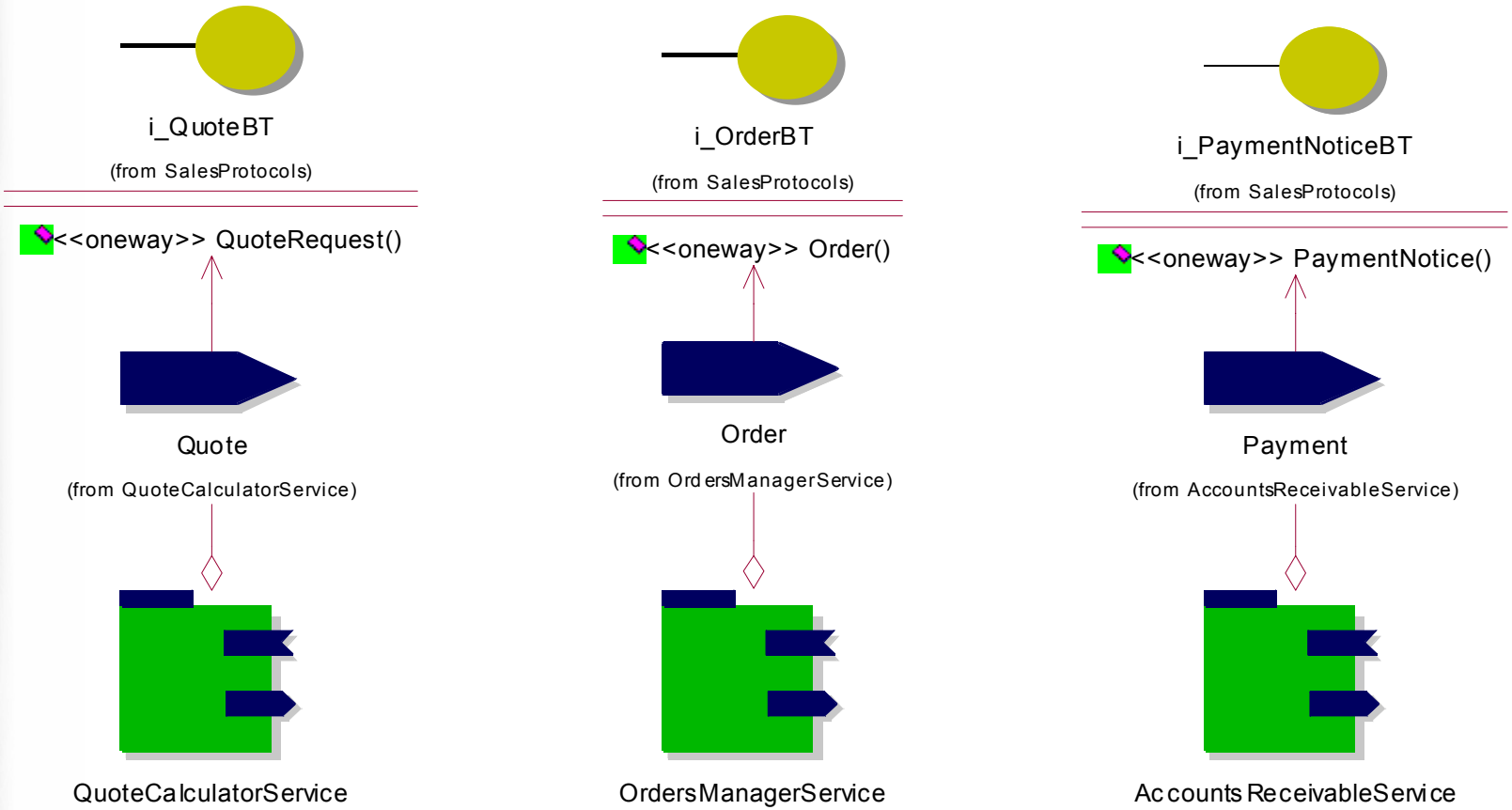
The Buyer-Seller-Example

The Mapping Result



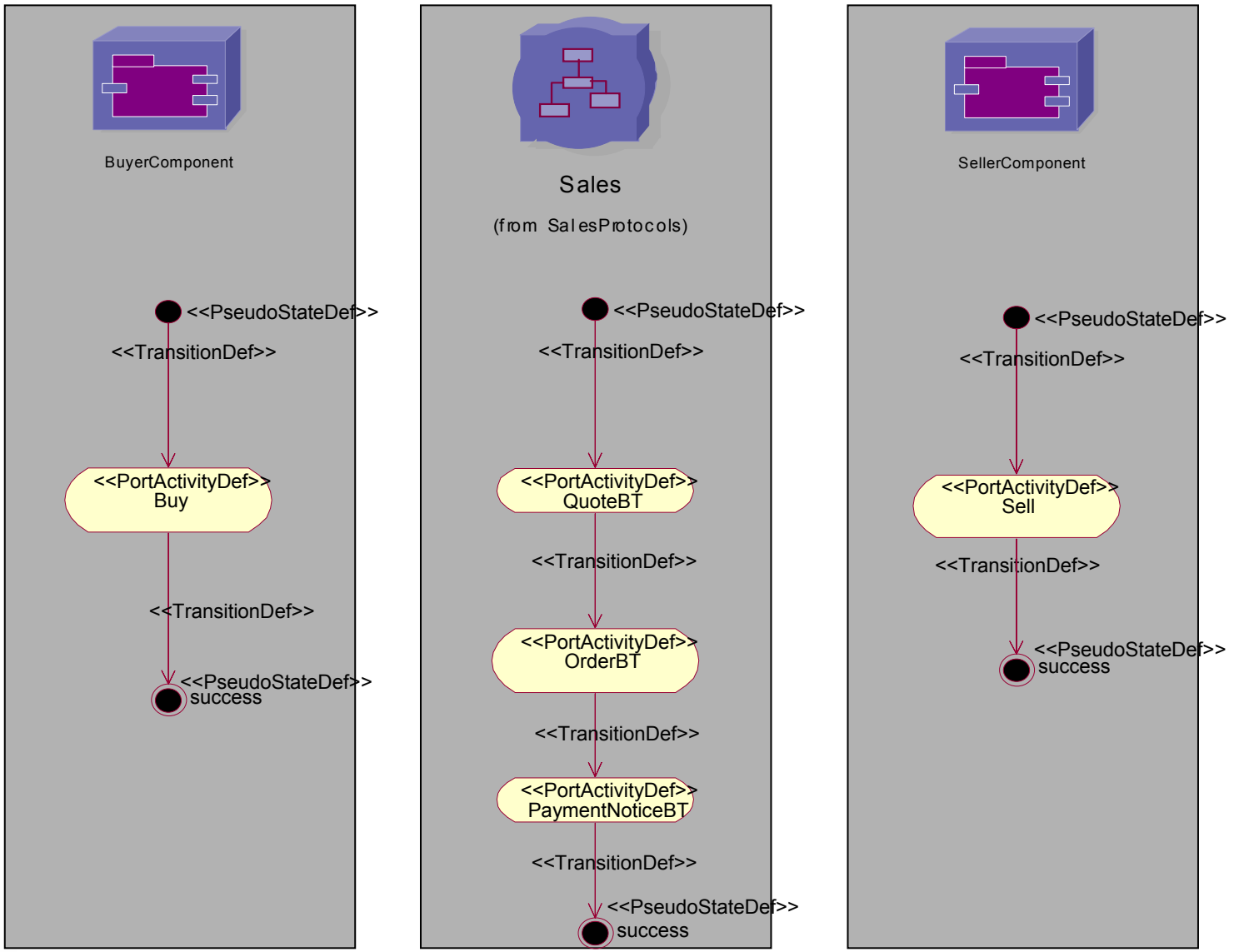
The Buyer-Seller-Example

The Mapping Result



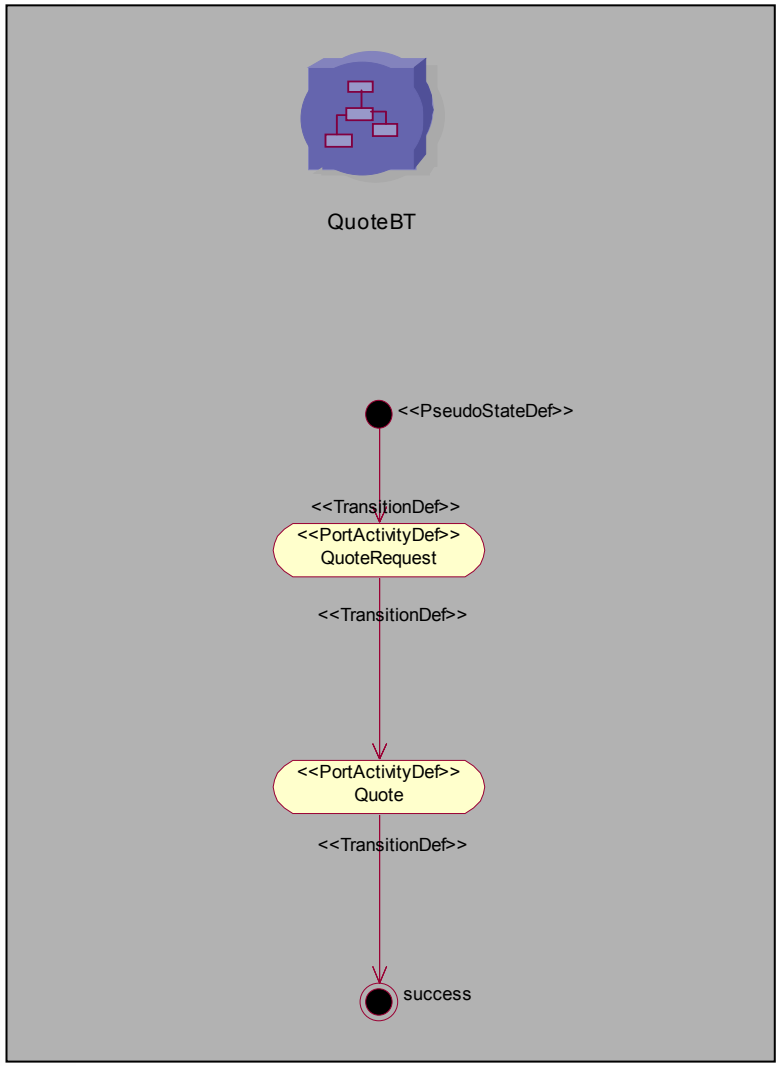
Behaviour

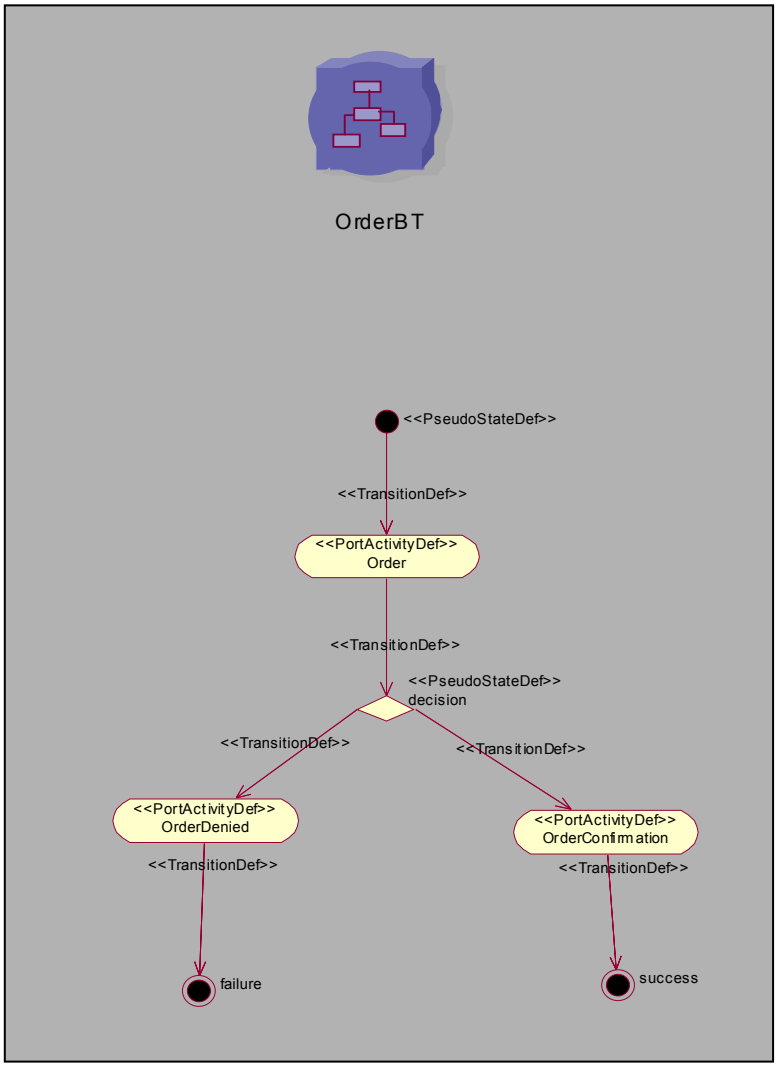
Choreographies



Behaviour

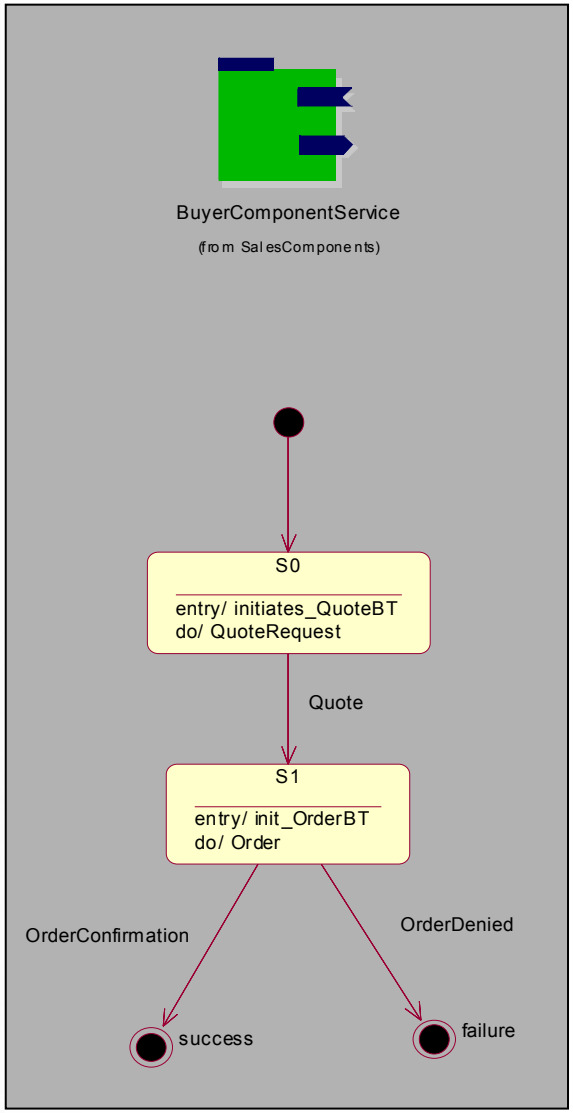
Choreographies

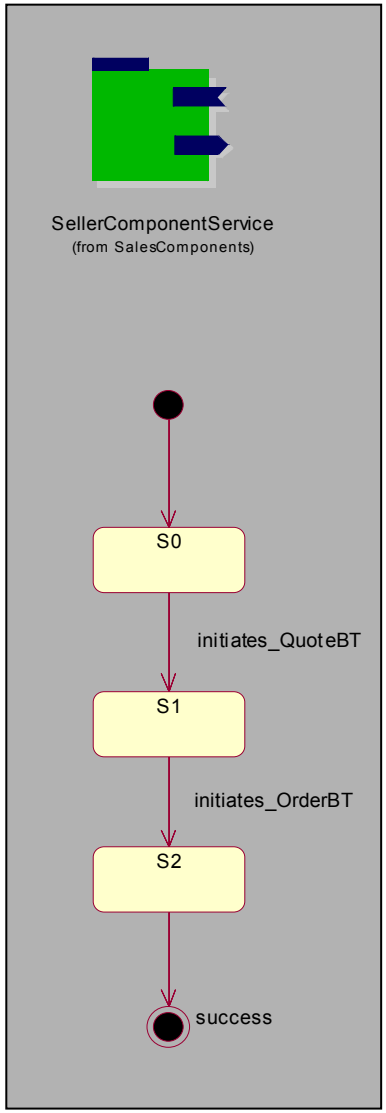


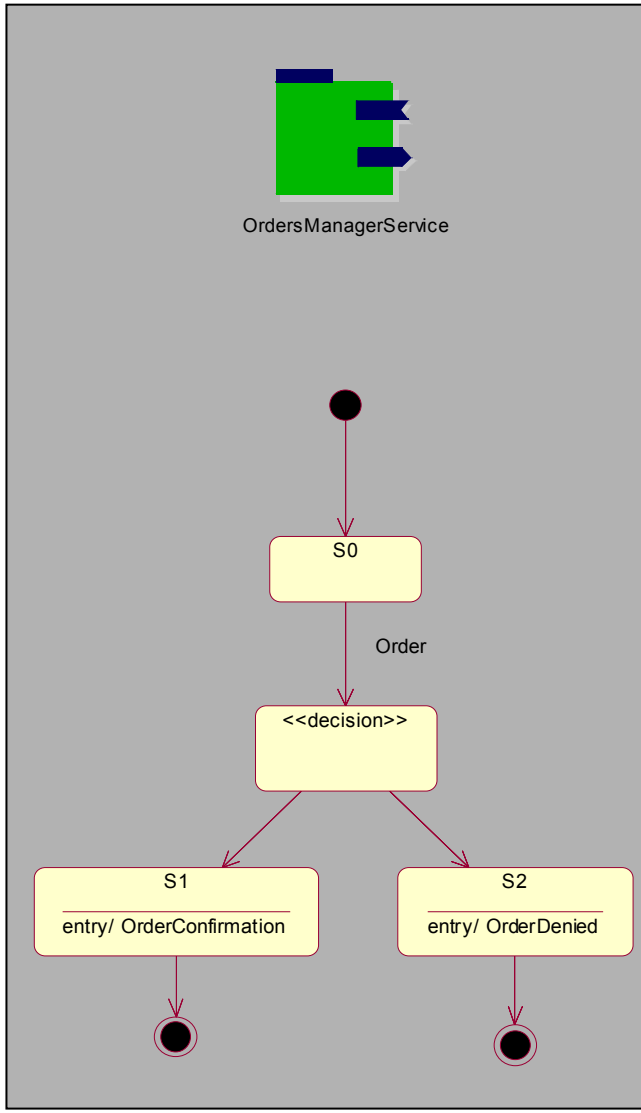


Behaviour

enago OSP State Machines







Behaviour
Java Code

```
public class BuyerComponentServiceSSM extends SinglePartySSM {
…
// this operation will be called when this SSM is started
   public void initialAction {
        // here the implementor has to assign a valid reference
           for sales_interface
        i_QuoteBT result1 =
                m_BuyReq.initiates_QuoteBT();
        result1.QuoteRequest();
  }
…
```

Behaviour

Java Code

```
public class i_cb_SalesImpl implements … {
    public void Quote () {
        stateTable.put(„i_cb_SalesImpl_ Quote“,
                Boolean.TRUE);
        i_OrderBT result1 =
                m_BuyReq.initiates_OrderBT();
        result1.Order();
    }
}
}
```

Behaviour

Java Code

```
public class i_cb_SalesImpl implements … {

   public void OrderConfirmation() {

         if (stateTable.get(„i_cb_SalesImpl_ Quote“) == null)
                  throw new RuntimeException(„incorrect state“);

         stateTable.put(„i_cb_SalesImpl_ OrderConfirmation“,
                  Boolean.TRUE);

   }

   public void OrderDenied() {

         if (stateTable.get(„i_cb_SalesImpl_ Quote“) == null)
                  throw new RuntimeException(„incorrect state“);

         stateTable.put(„i_cb_SalesImpl_ OrderDenied“,
                  Boolean.TRUE);

   }

}
```

}