ModelBus

An Open and Distributed Platform for MDD Tool Interoperability

LIP6, Laboratoire d'Informatique de Paris 6
8, rue du Capitaine Scott
75015 Paris, France
http://www.lip6.fr

Context of this work

- The present courseware has been elaborated in the context of the MODELWARE European IST FP6 project (http://www.modelware-ist.org/).
- Co-funded by the European Commission, the MODELWARE project involves 19 partners from 8 European countries. MODELWARE aims to improve software productivity by capitalizing on techniques known as Model-Driven Development (MDD).
- To achieve the goal of large-scale adoption of these MDD techniques, MODELWARE promotes the idea of a collaborative development of courseware dedicated to this domain.
- The MDD courseware provided here with the status of open source software is produced under the EPL 1.0 license.

Course Objectives

The course aims to explain the following points

- The needs for the interoperability between Model Driven Development (CASE) tools.
- Existing approaches for tool interoperability - the functionalities they offer and their limitations.
- The key features of ModelBus.
- The current implementation of ModelBus
- A guide for using ModelBus to integrate tools

Table of Content

- Introduction 5
- Tool Interoperability
  - Problems 9
  - Existing Solutions 14
- ModelBus
  - Concepts & Design 21
  - Current Implementation 33
  - Tool Integration Guide 36
- Conclusion 42

- Introduction

Model Driven Development

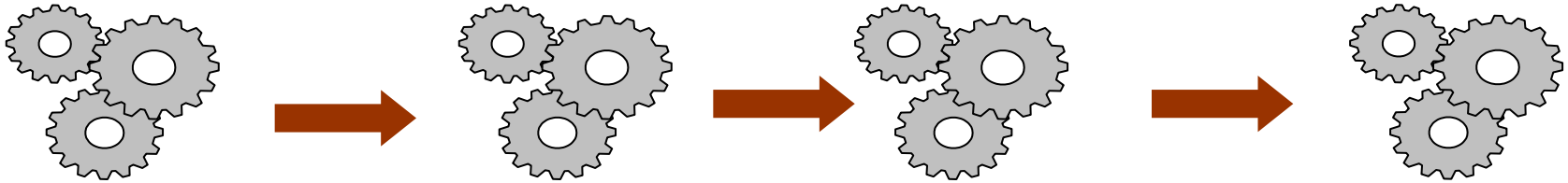
- Model-Centric
  - Models represents software artefacts in various aspects (Business aspect, Implementation aspect, QoS aspect)

  →Different kind of models: Both PIMs and PSMs.

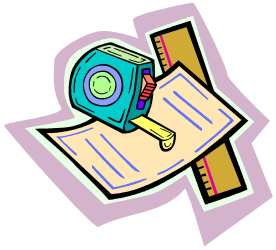
  PIM (UML models, User Interface models, …)

  PSM (EJB models, CORBA models, .NET models, …)
- Model Automation
  - Software development is assisted by automated operations for reducing development cost and human errors.
  - E.g. model visualization, model edition, model transformation, model verification, code generation, …
  - Provided by MDD tools (CASE tools for MDD)

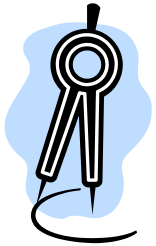
MDD tool chain -> Need for Tool integration



Requirements
- DOORS
- Mantis DB

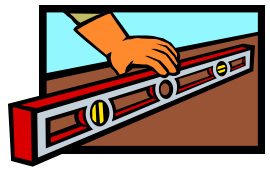


Design
- Objecteering
- IBM RSA



Implementation/ Code Generation
- MOF2Text
- UML2EJB

Validation
- Test Generators



- No Universal Tool -> Need to plug additional functionalities
- Need to support Distributed Tool Chains
- Need to replace Tool Chain tokens for more flexibility
- Need to share models between several Tools (Modelling, etc…)

MDD Tool Interoperability Problem

- MDD software development is complex
  - Involve several kinds of models (PIMs/PSMs)
  - Involve several software development activities (analysis, design, implementation, deployment, test)
- One MDD tool can not handle ALL models and activities → need to use several tools conjointly.
  - Ex. A model transformation engine may not support the visualization of the input/output models. Therefore it needs to be used conjointly with a model editor tool.
- **Integration/Interoperability problems**
  - How can tools share models ?
  - How can tools share functionalities ?

- Tool Interoperability Problems

Aspects of tool interoperability

Tool Integration =

- Data Integration ("data sharing")
  - Ex. How can tools share data (models) ?
- Control Integration ("service sharing")
  - Ex. How can a tool use a service of another tool?

  → Enable functionalities sharing
- Presentation Integration
  - Ex. How to unify the user interfaces of different tools in the same environment (workbench) ?
- Process Integration
  - Ex. How to support software engineering processes that involve several tools ?

Our focus: Data sharing/Service sharing

Data sharing

- **Goal: enable tools to read/modify models located in other tools.**
  - Ex: A UML model editor shares a UML model with the transformation engine: The transformation engine can access and modify the model located in the modeler.
- **Problem of model heterogeneity**
  - Different kinds of models
    - CIM, PIM, PSM
    - UML models, Domain Specific models
  - Different model formats / representations
    - Various versions of JMI, EMF, XMI, MDL
- Problem of model location
  - Model discovery (local / remote)
  - Efficient sharing mechanism

Service sharing

- **Goal: enable a tool to invoke a service of another tool.**
  - A tool can have two roles: Service Provider/ Service Consumer
  - Service invocation = request + reply

- **Problems**
  - Need for explicit service description
    - Parameters = Models -> Define Model Type
  - Tool location and service discovery
  - Invocation mechanism heterogeneity
    - Both local/remote service invocation is required

Research goals

=> Need transparency w.r.t.

- local/remote model sharing
- model/service discovery
- local/remote service invocation

- Existing Solutions for Tool Interoperability

Outline

- Eclipse-EMF platform
- MOF-CORBA Repository
- Exchange of XMI files
- Web Services integration

- Summary

Eclipse-EMF platform

- Eclipse
  - Eclipse is local environment for tool integration
  - Tool = "Eclipse plugin".
  - All plugins are registered within Platform
    - A plugin can be discovered via Platform.

- EMF (Eclipse Modeling Framework)
  - Model = Java objects
  - Model can be imported/exported to XMI.

- Reference: http://eclipse.org/emf/

MOF-CORBA Repository

- MOF-to-IDL
  - An OMG standard for generating a set of IDL interfaces for representing models as CORBA objects.
- MOF/CORBA Repository
  - Models = CORBA objects.
  - Tools access = CORBA RPC.
- Reference
  - Kath, O. et al., An Open Modeling Infrastructure integrating EDOC and CCM, Proc. of the 7th Int'l Conf. on Enterprise Distributed Object Computing, IEEE CS, 2003.

Exchange of XMI files

- Model format = OMG XMI
- Exchange of XMI files
  - Tool access = XMI file export/import

- Reference
  - Christian, H. D. et al., Tool Integration: Experiences and Issues in Using XMI and Component Technology, Proc. of the Technology of Object-Oriented Languages and Systems (TOOLS 33), IEEE CS, 2000.

Web Services integration

- Tools functionalities = Web Services.
- Tool access = SOAP/HTTP call.
- Models = XMI entries within SOAP messages.

- References
  - Togni, J.D. et al., Tool integration using the web-services approach, Proc. of the 15th ACM Great Lakes symposium on VLSI, 2005.
  - Mueller, W. et al., Dynamic Tool Integration in Heterogeneous Computer Networks, Design, Automation and Test in Europe Conference and Exhibition (DATE'03), 2003.

Summary

	Eclipse/ EMF	MOF-CORBA : the mostly used version (1.4)	XMI file exchange	Web Services
Data sharing:				
Model representation	Java objects	IDL-compliant representations	conversion to XMI is needed	XMI only
Local/remote model sharing	Only local sharing	Remote sharing	Manual	No support for local sharing.
Service sharing:				
Model Parameter Definition	No	N/A, data sharing only	N/A, data sharing only	No
Tool location & service discovery	Yes (only locally)	N/A, data sharing only	N/A, data sharing only	Transparent
Invocation	Only local	N/A, data sharing	N/A, data	Remote Only

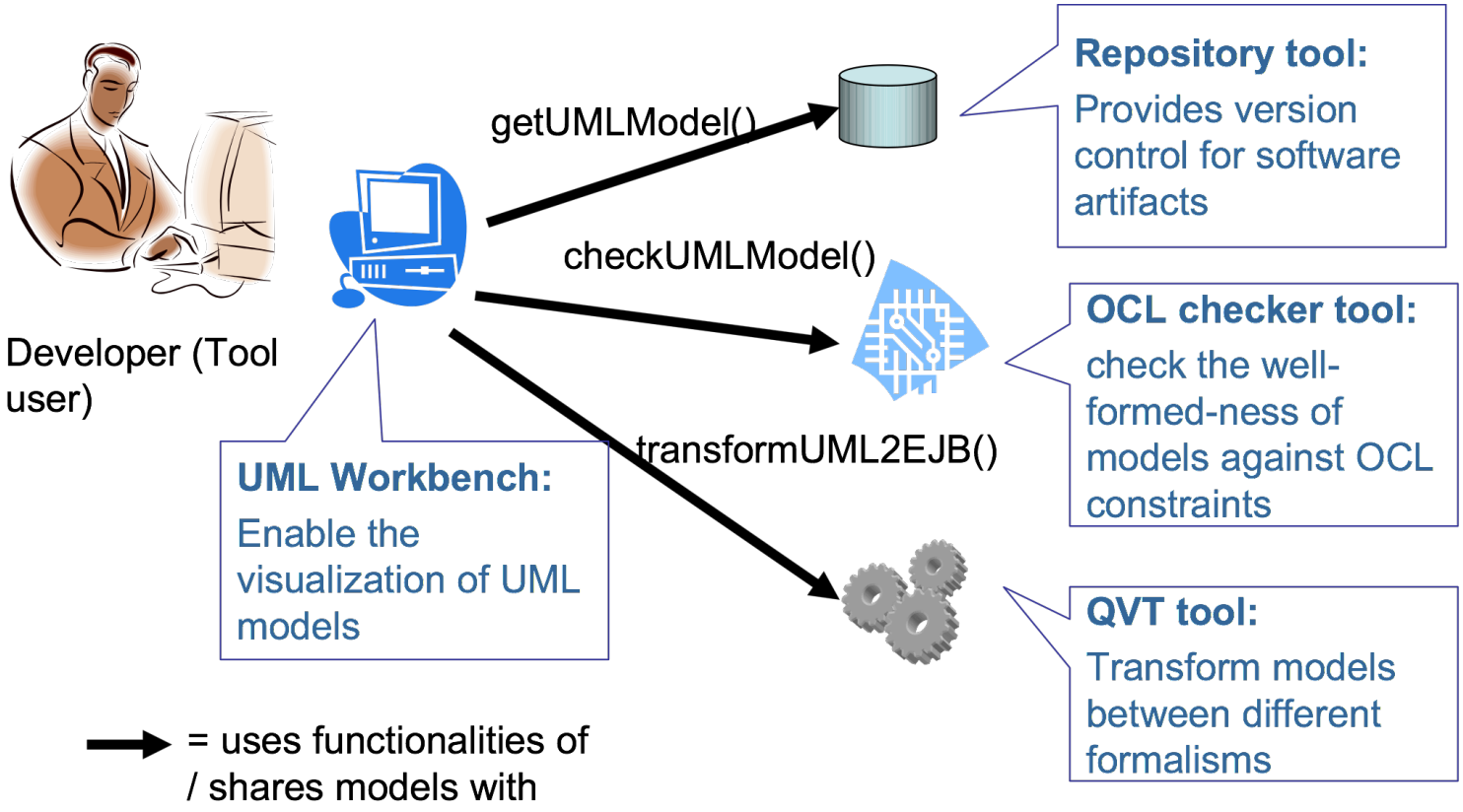
- ModelBus: Concepts & Design

Outline

- ModelBus Principles
- Close look to ModelBus concepts
  - Abstract Modelling Service Description
  - Components: Adapter, Registry and Notification Broker
  - Component Interactions for service invocation and notification
- Current Implementation
  - Features Available
  - Tools plugged

ModelBus Principles

- Model Driven Development is "orchestration" of **modelling services**
- Goal of ModelBus = **Infrastructure** for modelling service **integration** and **interoperability**



ModelBus Principles (2) – Infrastructure

- Based on Service Oriented Architecture
- Reuse conventional middleware – Web Services

- Add new features
  - Conceptual level: → Tool description language
  - Execution level: →Transparent Model/Service Sharing

Close look to ModelBus: New Features

- **Conceptual level**: → Tool description language
  - Specify the services a tool shares
  - Provide an abstraction from tool implementation / transport implementation

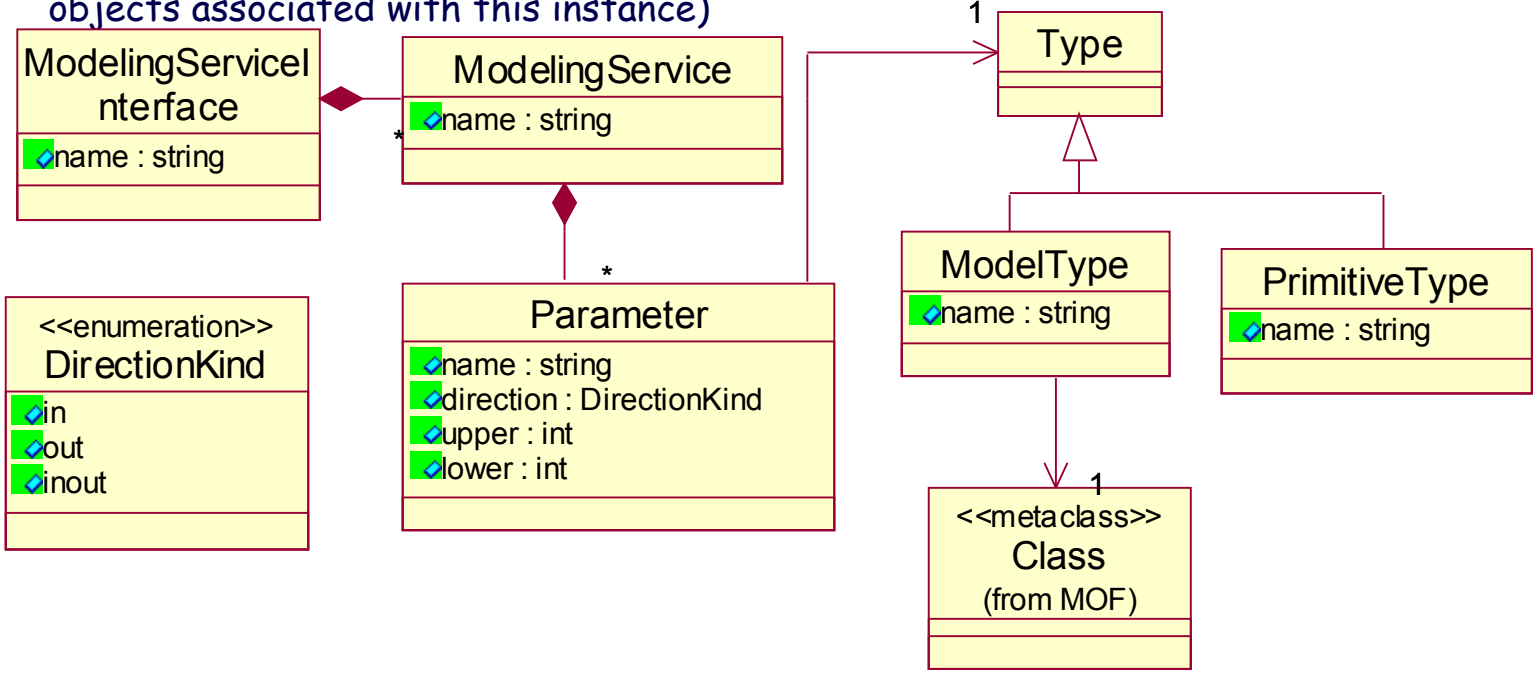
- **Execution level**: → Adapted Run-time Infrastructure
  - Transparent model sharing
    - Automated model format conversion
    - Support several model transmission granularities
      - Model fragment/Complete model transmission
  - Transparent Service sharing
    - Automated modelling service discovery
    - Automated transport selection (local/remote)

Conceptual level: Tool description language (1)

- Abstract tool description
  - Concept of **modeling services**
  - Service parameters (input/output): content of the request/ reply messages
  - Services parameters are defined by ModelTypes
    - Define the models inputs/outputs of services
  - Concept of **ModelType** : Based on metamodels
    - Ex. "UML Model Type" define models that conform to the UML metamodel
  - A **tool description** : a document defining the services of a tool.

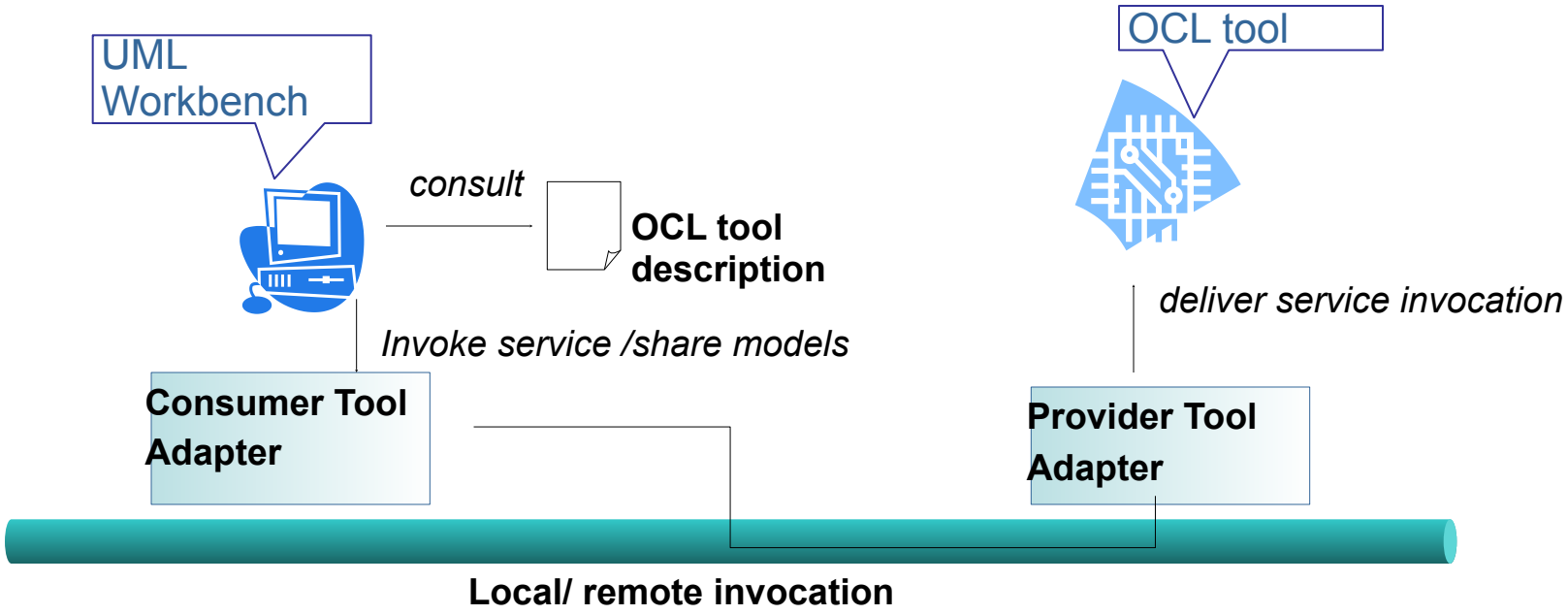
Conceptual level: Tool description language (2)

- **ModelingServiceInterface** define the services provided by a tool.
- A **ModelingService** contains parameters that define the input/output models for this service.
- Model sharing role is specified by the direction of the parameter.
  - **in**: The service consumer tool shares a model for read-only.
  - **inout**: The service consumer tool shares a model for mutable access.
  - **out**: The service provider tool shares a model for read-only.
- **ModelType** defines the models that can be passed as the parameter.
  - The model to be passed is an instance of the specified metaclass (and all the objects associated with this instance)



Execution Level: Infrastructure

- Service discovery mechanism
- Model format conversion mechanism
- Transport mechanism:
  - Local/Remote invocations,
  - Notification: event propagation
- Service execution transparency



Execution Level: ModelBus Components

Adapter

- Adapter – built-in component, makes a Tool to be ModelBus enabled
  - Invocation: (1) Service selection; (2)Model format adaptation;

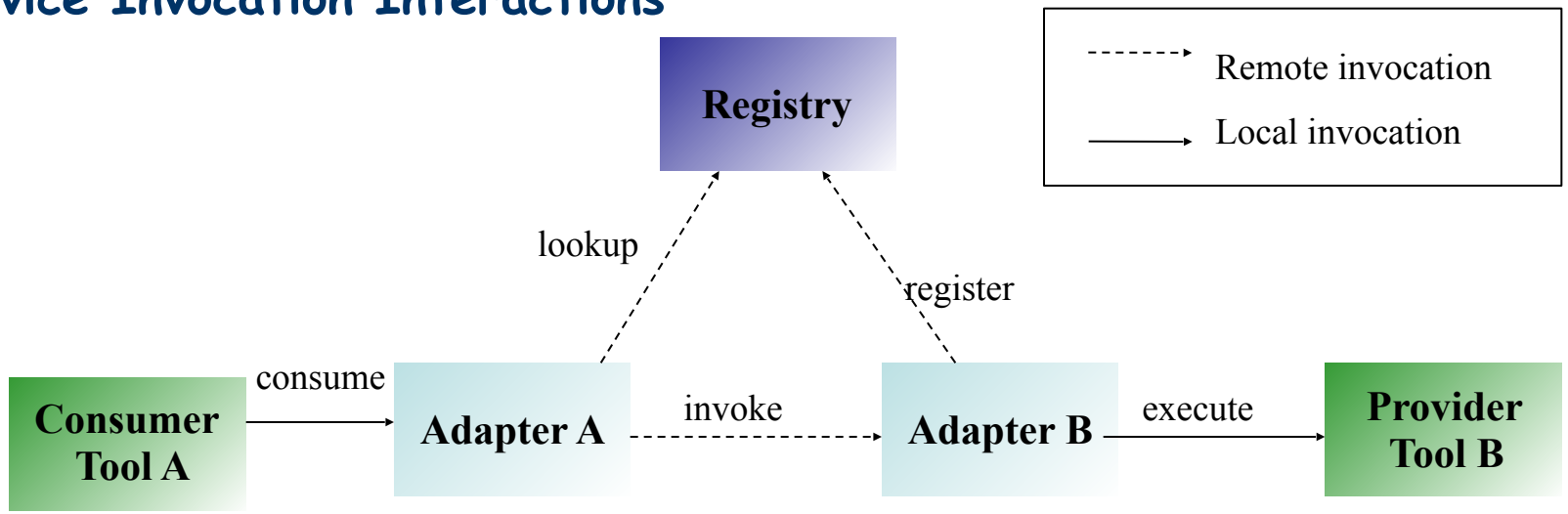
Registry

- Registry – service discovery component
  - Register Modelling Service description
  - Lookup service

Notification Broker

- Notification Broker – mechanism for instant asynchronous messaging
  - Manages subscriptions
  - Broadcast events

Service Invocation Interactions

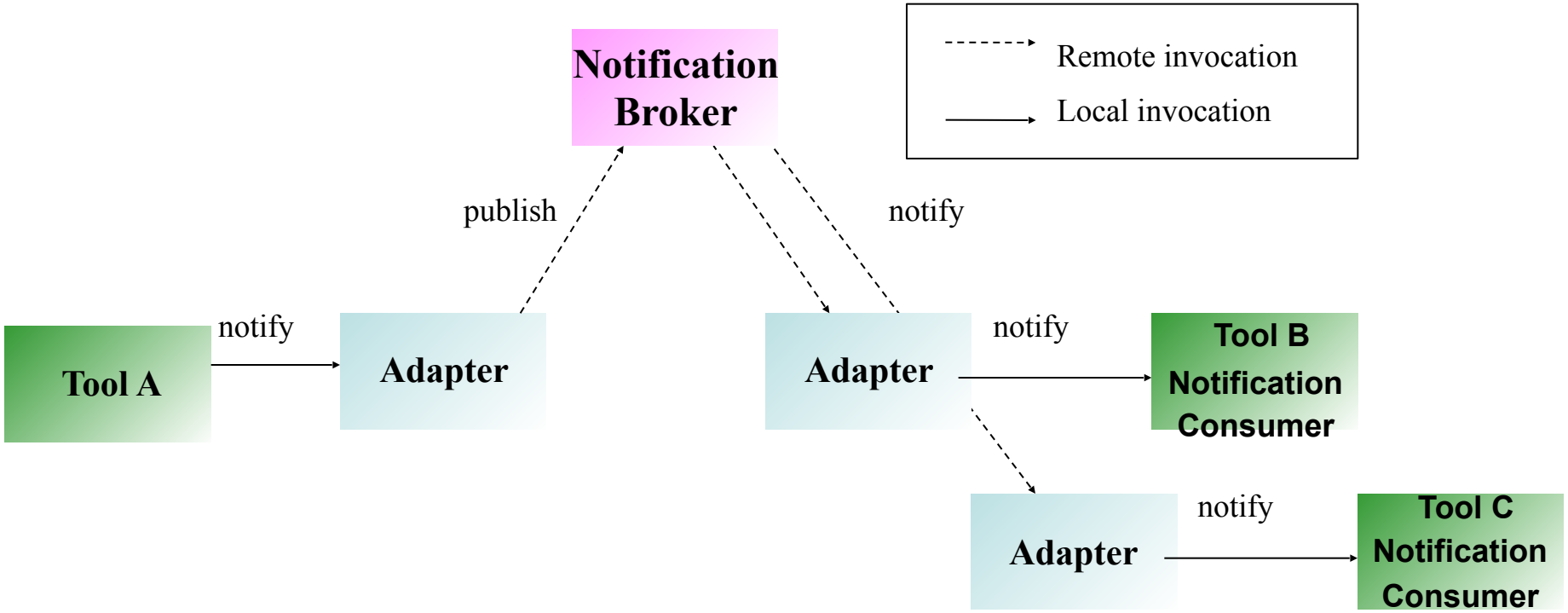


1. Tool B deploys adapter
2. Tool A consumes a service

Behind:

1. Adapter B registers Tool B description
2. Adapter A looks up for a service
3. Adapter A invokes Adapter B, which executes corresponding service
4. Adapter A returns a result to the tool

Notification Interactions



1. Tools B and C subscribes to a topic of interest
2. Tool A publishes a notification to one of topics of interest

Behind:

1. Adapters provide a simplified façade for Notification Broker and manage remote communication
2. Notification Broker manages subscriptions /event propagation

More Information

- ModelBus Architecture
  - Functional Architecture

http://www.eclipse.org/mddi/D3.1%20ModelBus%20Architecture%20Specification%20-%20Volume%20I.pdf

  - Design Views

http://www.eclipse.org/mddi/D3.1%20ModelBus%20Architecture%20Specification%20-%20Volume%20II.pdf

- ModelBus Implementation

What is available in ModelBus

- Adapter for Java tools
  - Adaptable transport
    - Remote calls using Web Service interface
    - Local calls via Java run-time
  - Model Format Adaptation
    - Pluggable Serializers
  - Flexible deployment
    - Stand-alone version
    - Servlet
    - Eclipse
- Registry
  - Service Discovery
  - Web Interface
- Notification Broker
  - fully WS-Notification compliant
- ModelBus Integration Toolkit
  - Tool Description Editor
  - Consumer-side stub generator (see Integration Tutorial)
- Distributed within Eclipse MDDI: http://eclipse.org/mddi/

Tools plugged = Services Available

- IBM - http://www.ibm.com/
  - RSA Modeller and Repository
  - Model Simulator, Test Generator
- Softeam Objecteering - http://www.softeam.com/
  - Repository
  - MDA Modeller
  - Orchestration
- Adaptive Repository - http://www.adaptive.com/
- INRIA ATL Engine - http://www.eclipse.org/gmt/atl/
- France Telecom QVT Engine - http://www.francetelecom.com/
- FHG OSLO - OCL Checker - http://oslo-project.berlios.de/
- SINTEF MOFScript - http://www.eclipse.org/gmt/mofscript/
- Maven adapted by ZEG - http://www.zuehlke.com/
- Together adapted by SINTEF - http://www.sintef.no/

- Integration Tutorial

Outline

- **Scenario:**
  - You have a UML Modeller tool
  - Make use of "OCL check" services exposed via ModelBus

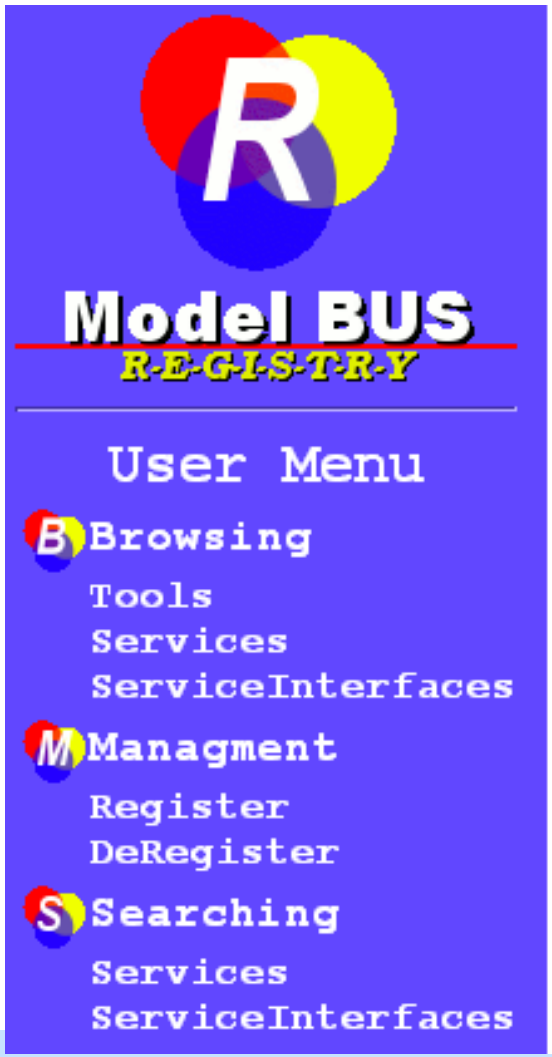
- **Steps to follow**
  1. Browse Registry and get Tool Description
  2. Generate a consumer Stub
  3. Use it in your tool

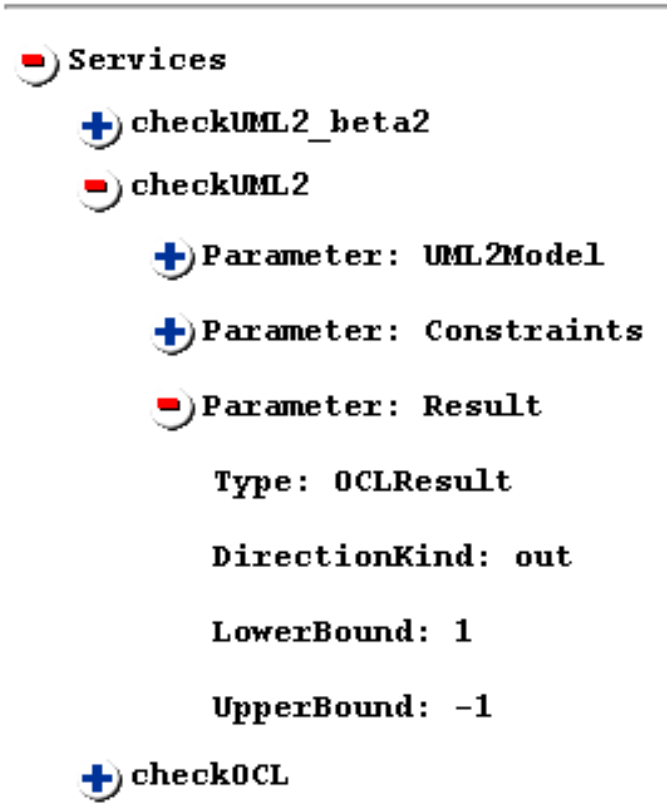
- Usage of ModelBus is as simple as calling a single java method is.

```
consume_ServiceIlike(required_parameters);
```

Browse Web Registry

- Get tool description from the Web Registry
- http://monarch.fokus.fraunhofer.de:8080/WebRegistryClient/





Generate Consumer Stub

- Run Toolkit Generator => OCLToolServiceInterface, OCLToolServiceInterfaceStub

- It contains:

```
consume_checkUML2(Collection UML2Model,
                                  String[] Constraints) throws
                ModelBusCommunicationException,
        ServiceUnknownException,
    NoToolAvailableException,
    ModelTypeMismatchException;
```

Use it in your tool

- Adapter Setup: specify registry location

```
OCLToolServiceInterface ocltool = new
    OCLToolServiceInterfaceStub(properties);
```

- Stub Usage: prepare parameters and call stub

```
result =
    ocltool.consume_checkUML2(testmodel,constraints);
```

More Information

- ModelBus Tool Integration using Generic Adapter
  - in the docs of
  - http://www.eclipse.org/downloads/download.php?file=/technology/mddi/mb-adapter-1.zip
- See Tutorial for ModelBus Integration using Toolkit
  - http://www.eclipse.org/mddi/ModelBusToolkitTutorial_0.7.pdf

- Conclusions

Conclusions

- MDD needs modelling tool integration
- The current approaches have certain drawbacks
  - Local environments => need infrastructure for distribution
  - Conventional middleware => need adaptation for the modelling domain
- ModelBus proposes Service Oriented Architecture for modelling tools
  - Tool Description in an abstract manner
  - Flexible Transport Infrastructure based on Web Services
  - Tool Integration Toolkit
- ModelBus implementation is available
  - More than 10 leader tools have been already integrated

- Integrate your tools and use available modelling services !