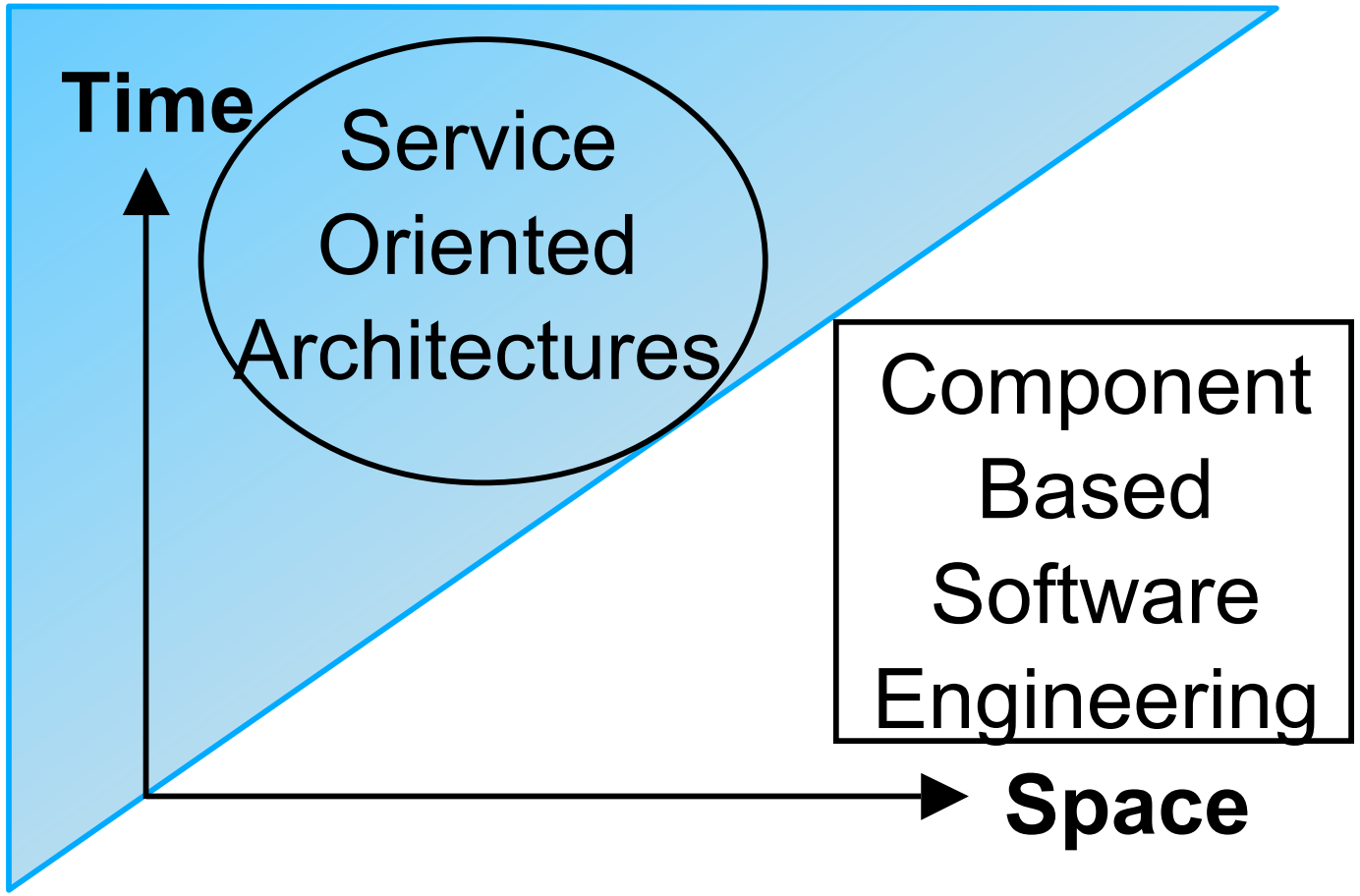
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IKS Information and Communication Systems Research Group

Executable Modeling of Generic Service Compositions with JOpera

Cesare Pautasso
Department of Computer Science, ETH Zurich, Switzerland
pautasso@inf.ethz.ch – www.jopera.org

How to model composition



Goal: Executable Service Composition

1. Design a **simple** workflow **language** for rapid composition of generic **services**
2. Build a user-friendly, efficient and **autonomic system** to execute it

3. Ensure their independence from the actual mechanisms and protocols involved

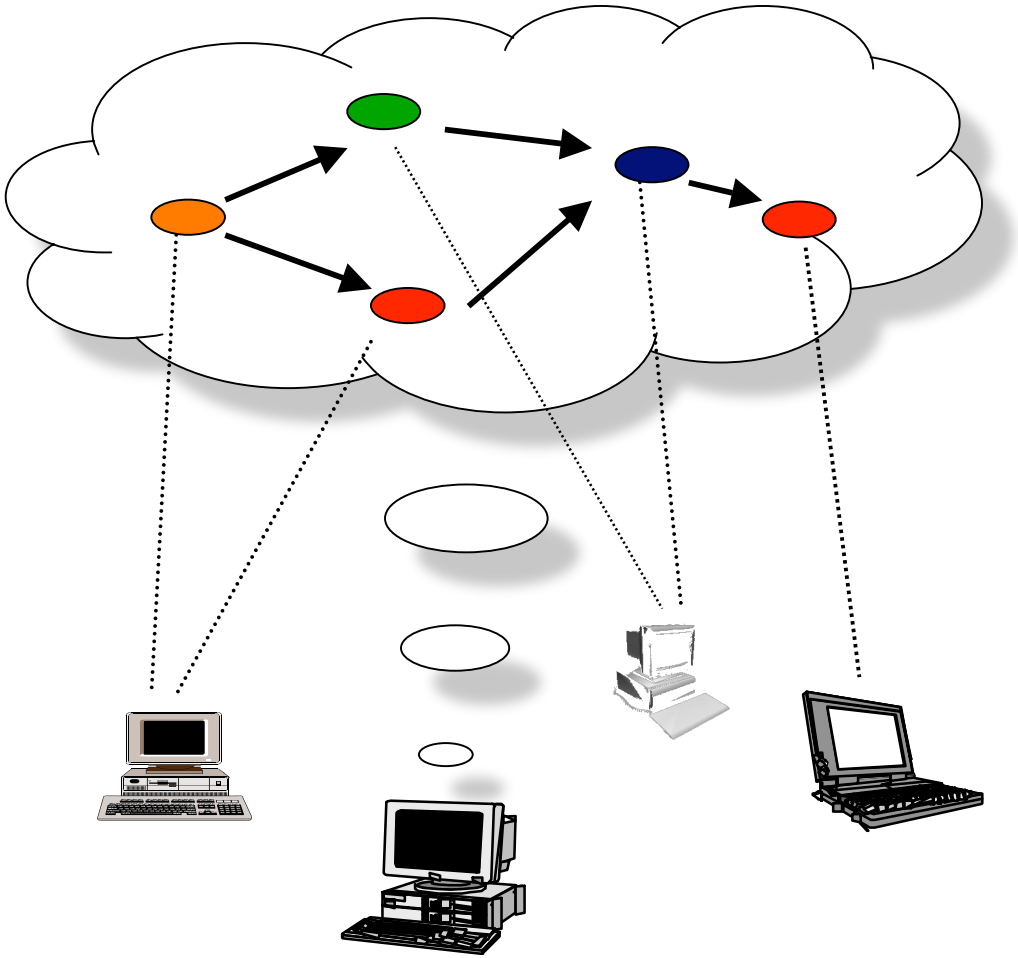
   (there are lots of standards and they change all the time)

About JOpera for Eclipse

1. **Modeling** service composition as workflow
   - Graph-based, functional workflow modeling language (Visual syntax, XML under the hood)
   - Workflows not limited to Web/Grid services
2. **Execution** of the workflow models
   - Extensibility (Eclipse plug-ins to provide custom adapters for service invocation & publishing)
   - Distributed engine (on a cluster of computers)
   - Autonomic engine (self-healing, self-tuning)
   - Efficiency (optimizing compiler to Java bytecode)

[ICAC, ICWS 2005]

Modeling Service Compositions with JOpera for Eclipse



Web Services should be composed using a **visual** composition language

print “hello world!”

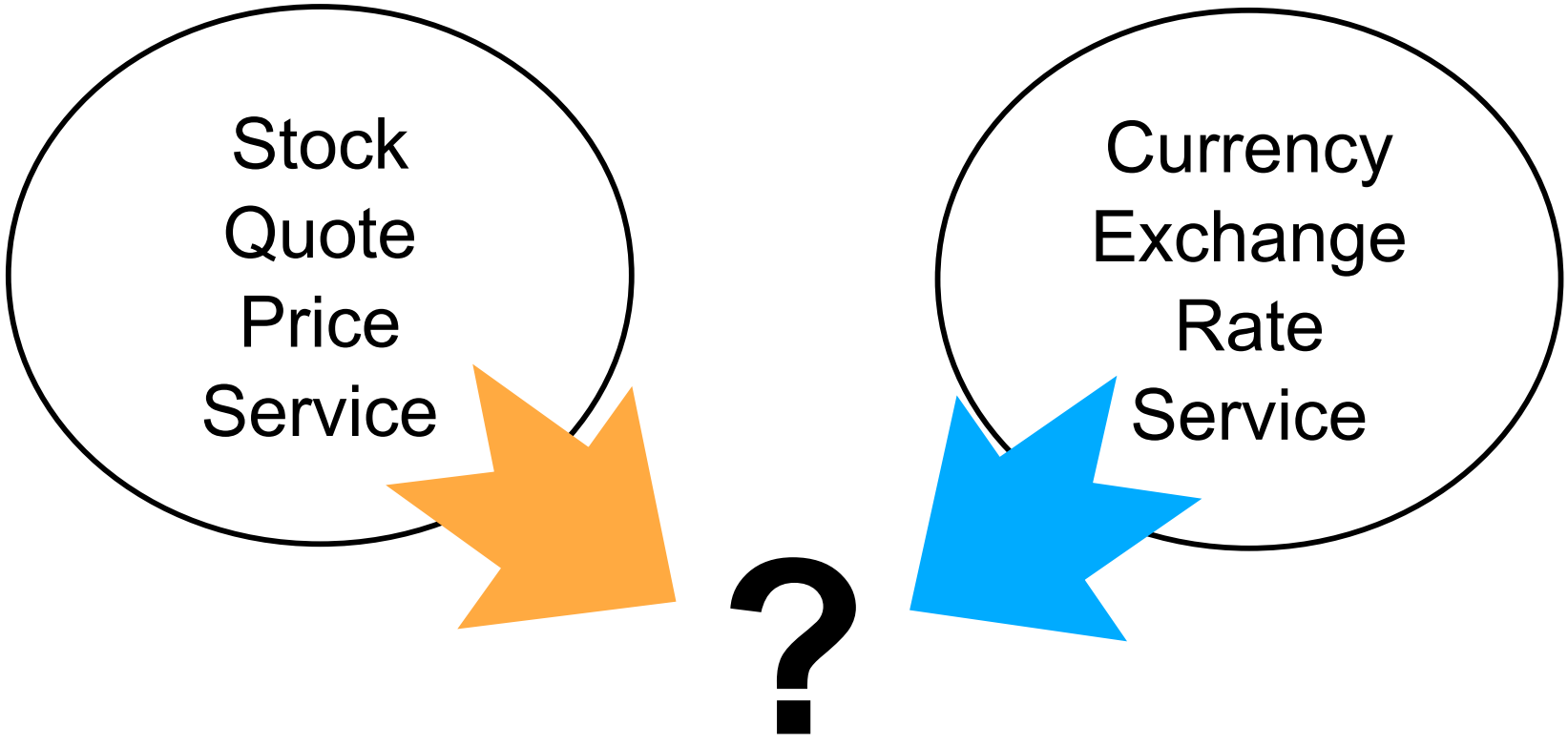
```
<!-- HelloWorld BPEL Process -->
<process name="HelloWorld"
         targetNamespace="http://samples.cxdn.com"  suppressJoinFailure="yes"
xmlns:tns="http://samples.cxdn.com"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
>

   <!-- List of services participating in this BPEL process -->
   <partnerLinks>
      <!--
      The 'client' role represents the requester of this service. It is
      used for callback. The location and correlation information associated
      with the client role are automatically set using WS-Addressing.
      -->
      <partnerLink name="client"
                   partnerLinkType="tns:HelloWorld" myRole="HelloWorldService"
partnerRole="HelloWorldRequester"
                   />
   </partnerLinks>

   <!-- List of messages and XML documents used as part of this
       BPEL process
       -->
   <variables>
      <!-- Reference to the message passed as input during initiation -->
      <variable name="input"
                messageType="tns:initiateHelloWorldSoapRequest"/>

      <!-- Reference to the message that will be sent back to the
          requestor during callback
          -->
```

Workflow Lifecycle in JOpera for Eclipse

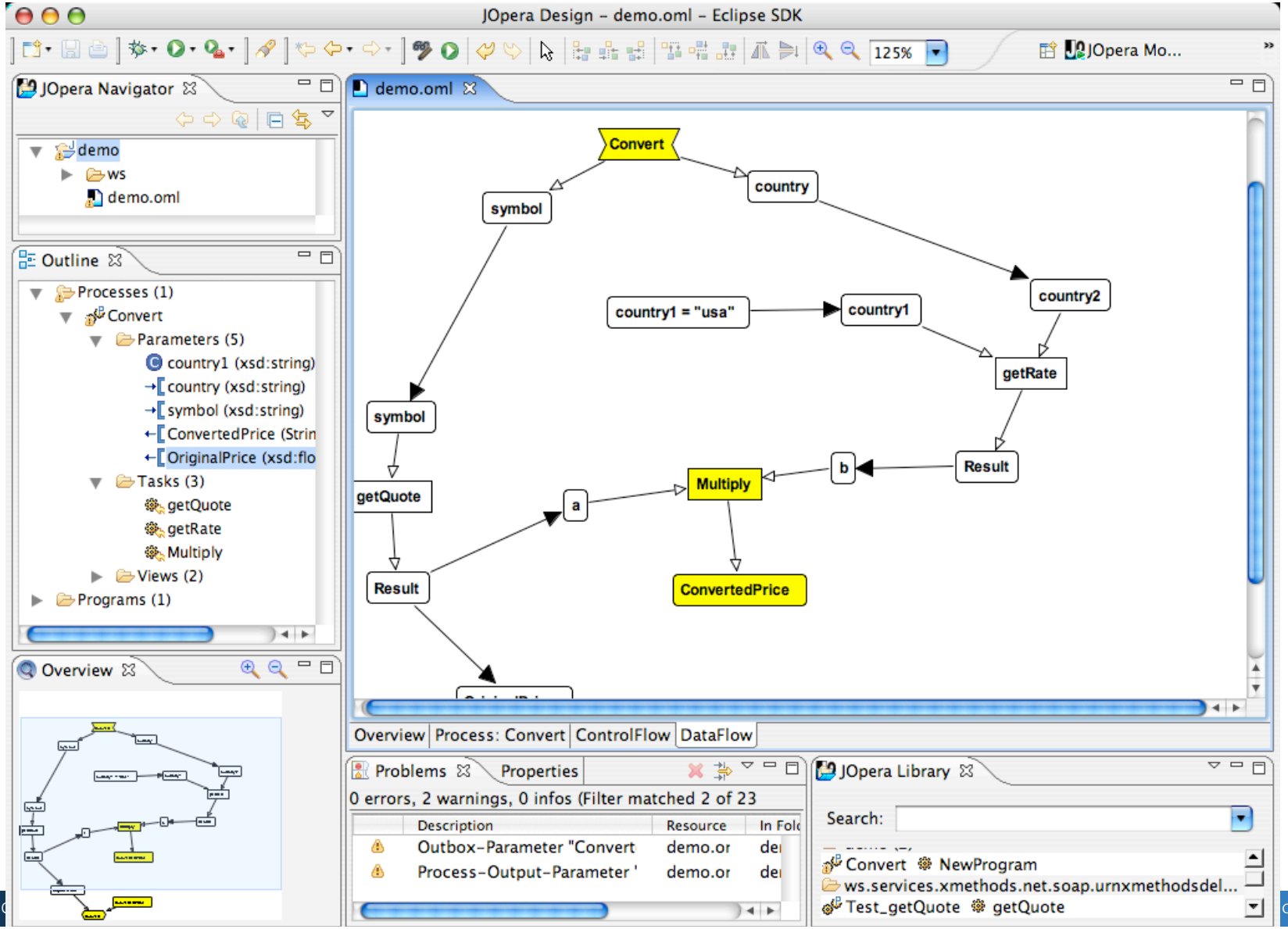
1. Select component services from a **library**
2. Build a process using a drag, drop and connect **visual** environment
3. Run, Test, and Debug the process execution **within the same visual environment**
4. Deploy, Manage, Monitor, and Steer the execution of processes in production
5. Publish the process as Web Service

Quick Demo Example

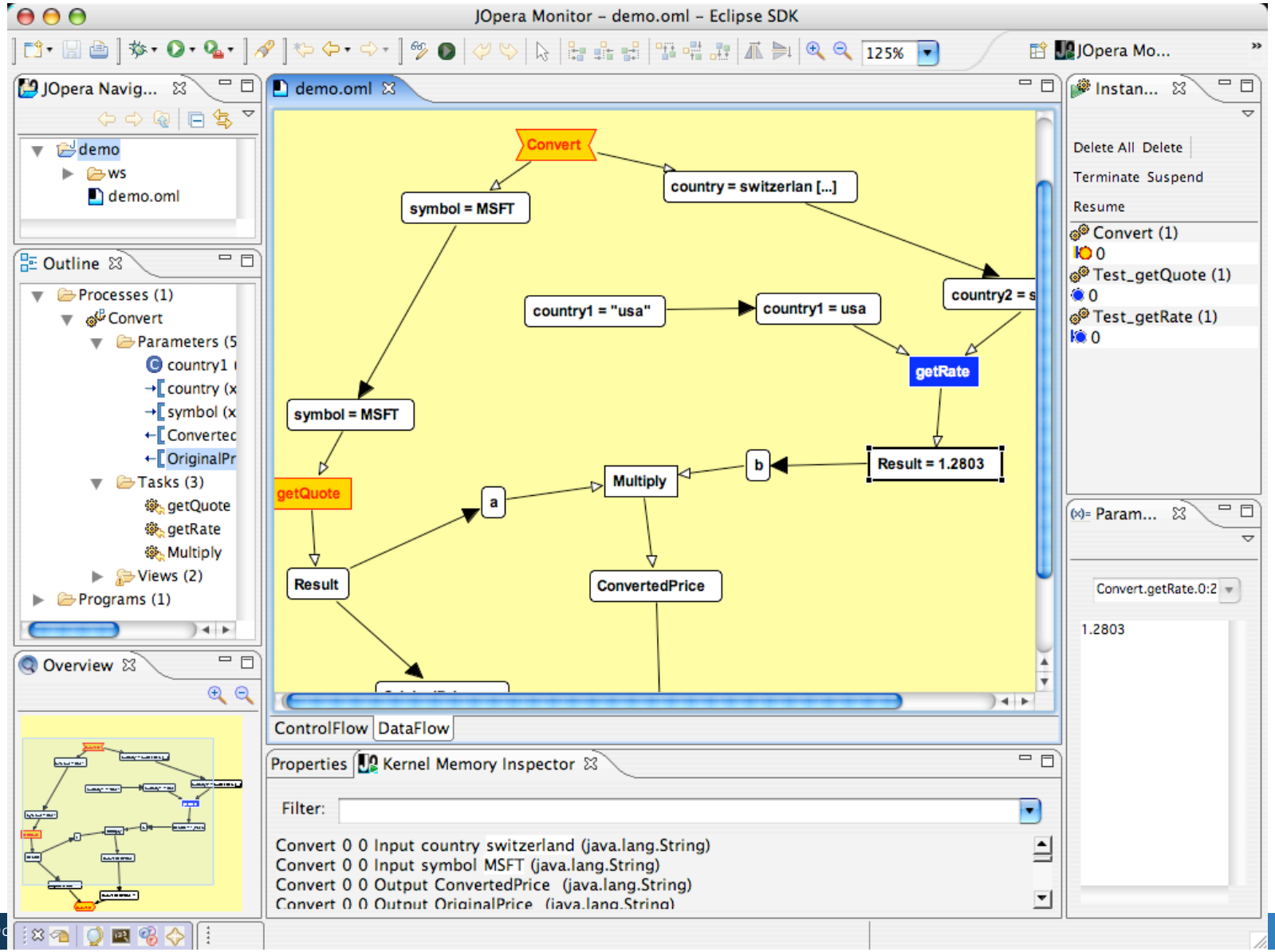
- Stock Quote Currency Conversion



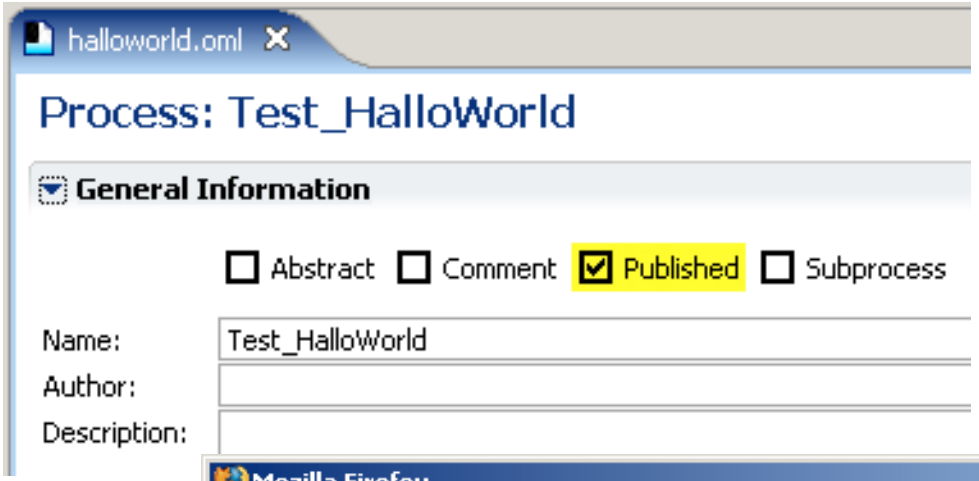
Drag, Drop and Connect



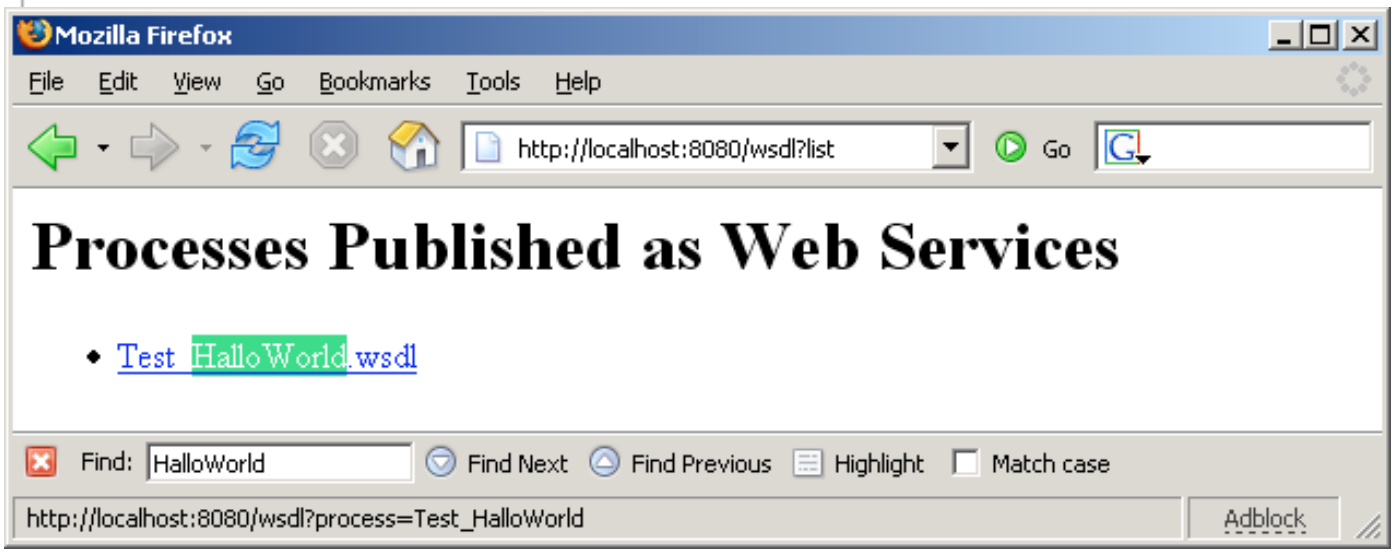
Run, Monitor, Steer and Debug



Publish as a Web/Grid service



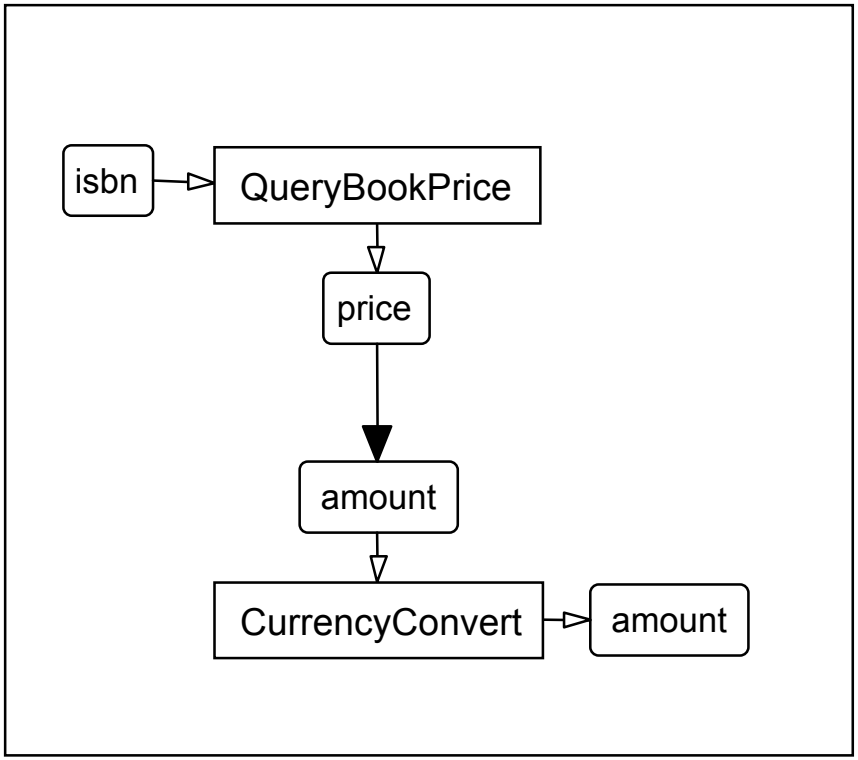
With one mouse click!

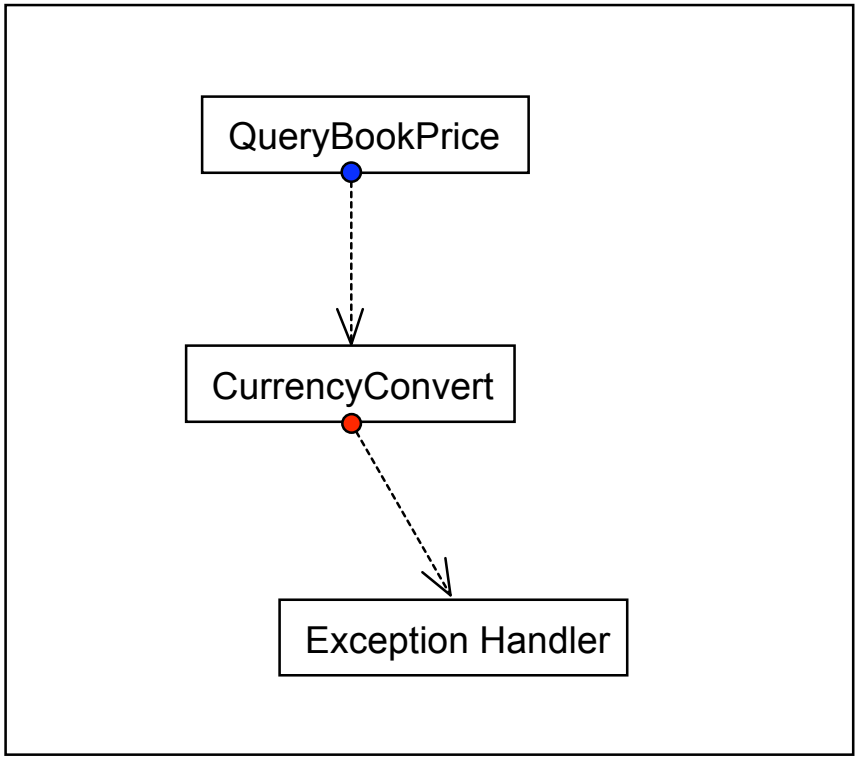


[e-Science2005]

JOpera Visual Composition Language Overview

- Services are composed using processes, which define their interactions using two graphs:
  - Data Flow
  - Control Flow





JOpera Visual Composition Language Features

- Processes model generic service composition
  - **Data flow** as the primary representation
  - Explicit **control flow** (branch, synchronization, exception handling, loops, pipeline, workflow patterns)
- **SubProcesses**: Modularity, Nesting and Recursion
- **First order functions**
  - Map (parallel/sequential/discriminator) and Reduce
- **Reflection** (introspection)
  - Dynamic late binding
  - Quality of Service monitoring

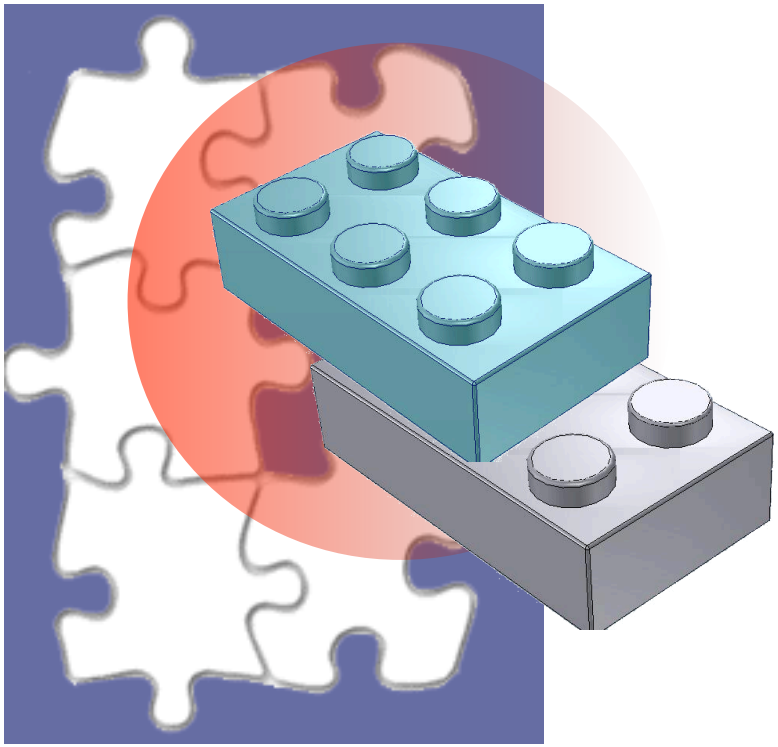
[JVLC2005]

JOpera Rapid Composition Environment

- Drag&Drop&Connect visual metaphor
- Immediate Feedback
  - Errors and Warnings are provided while editing
  - Execution is monitored within the same language
- Automatic Completion
  - Connect to “matching” parameters
  - Suggest “matching” services
- Visual Refactoring
  - Element Renaming
  - Sub-Process extraction and inlining

[VL/HCC2005]

Generic Service Composition with JOpera for Eclipse



How NOT to deal with heterogeneity

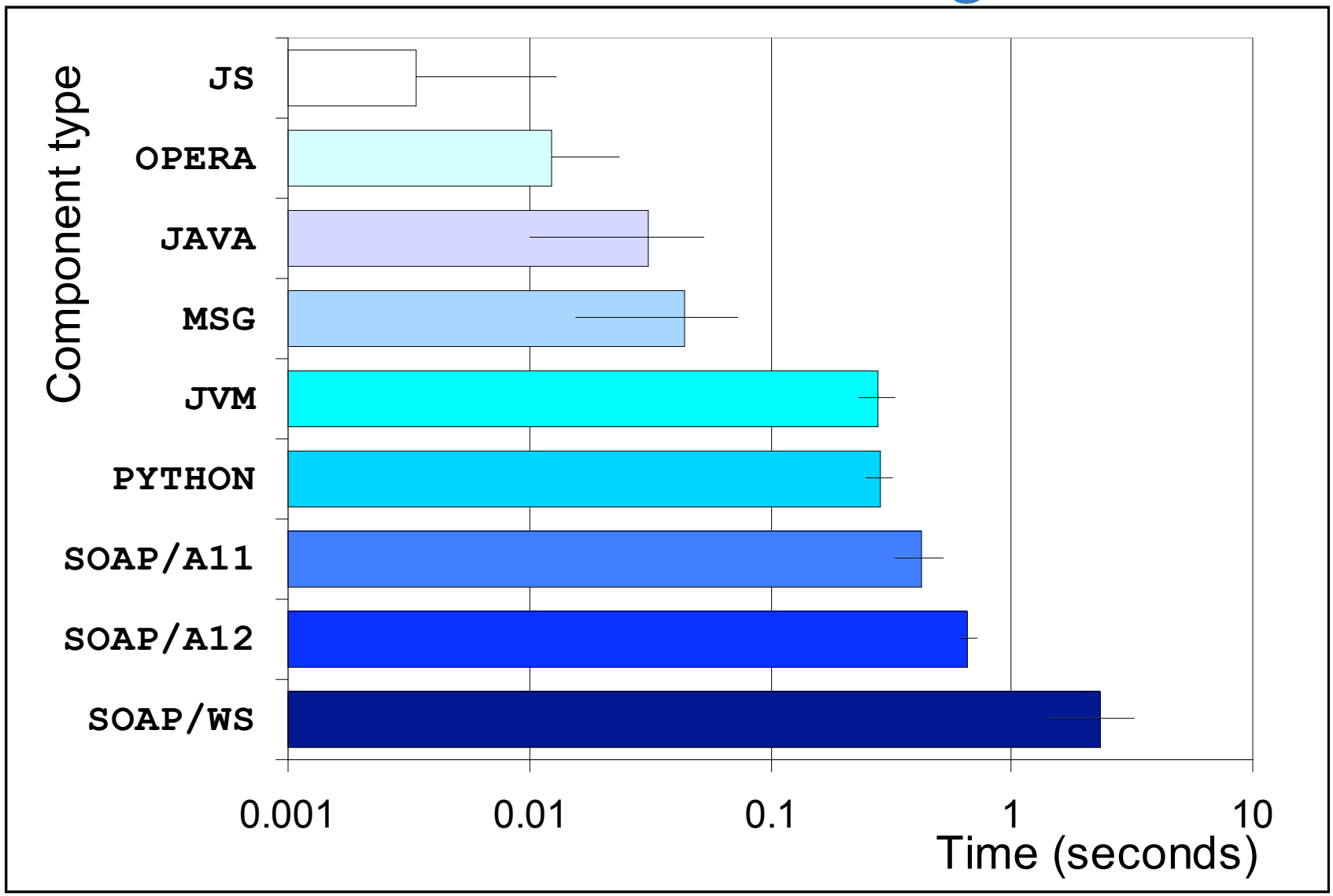
1. Assume that all services to be orchestrated will conform to one standard
2. Force all existing implementations to be wrapped to comply with that standard
3. Modify the workflow language to extend its support to other standards

(See BPEL, BPELJ, BPEL# controversy for an example)

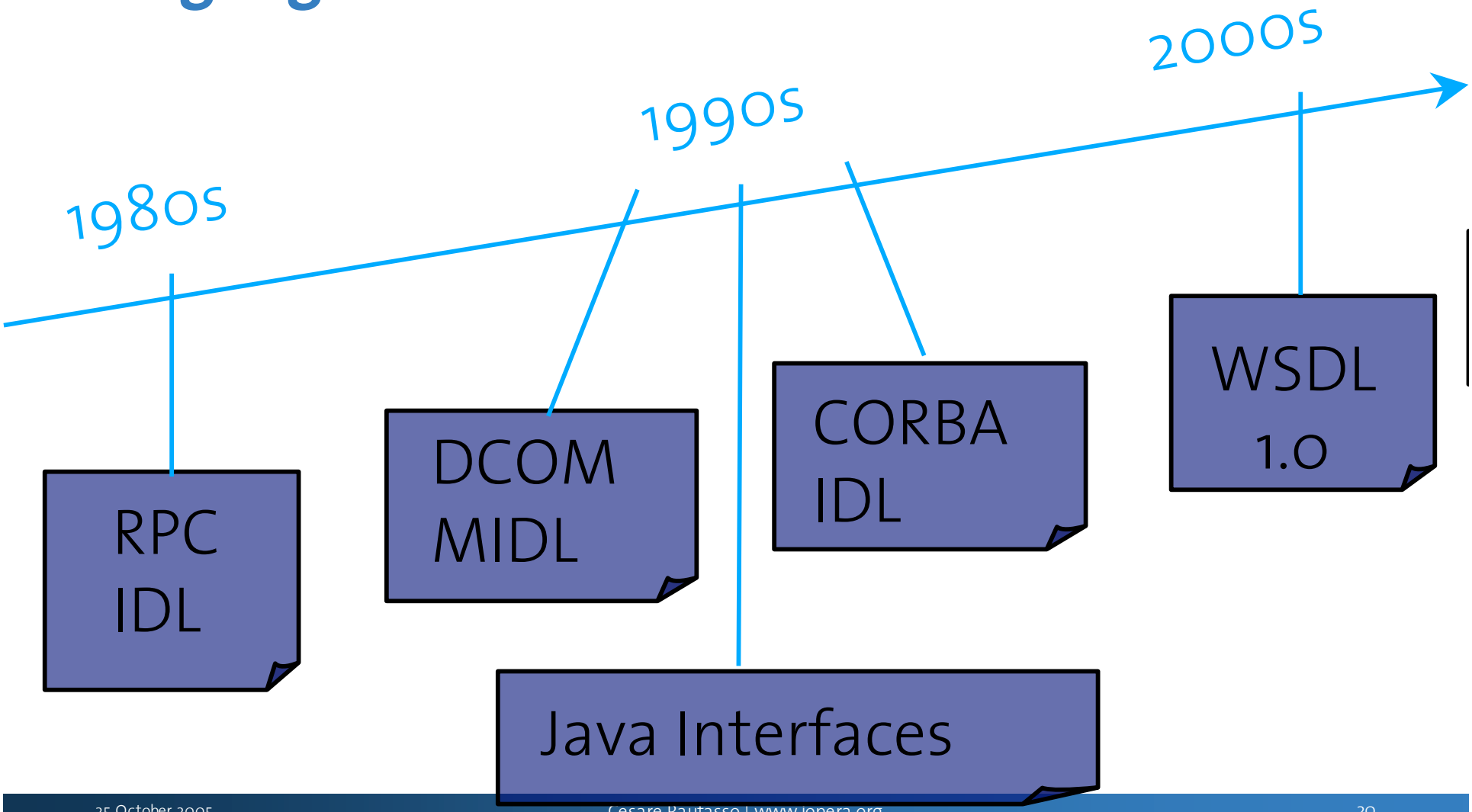
Problems of composing *only* Web Services

- Web Services are **coarse-grained**
- All existing heterogeneous tools must be **wrapped** as a Web Service
  - Wrapping imposes both a performance penalty and additional development & maintenance costs
- The **adapter/mediator** between mismatching Web services must also be a Web service
- Web services standards are not stable

Service Invocation Overhead (Log-scale)



A Brief History of Interface Description Languages



How to design a composition language **independent** of the types of services to be composed?

Generalizing service composition

- How to design a workflow language independent of the kinds of services to be orchestrated?

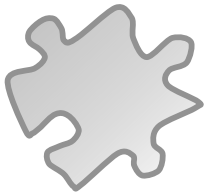
1. Separate the description of the process from the description of how to invoke each of its tasks
2. A process should make minimal assumptions about its tasks (i.e., data flow signature)
3. Bind tasks to different invocation mechanisms without affecting the process definition

[VLDB/TES2004]

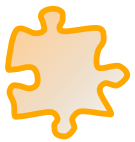
Main advantages

- Freedom of choice for developers:
  - Use the most appropriate kind of service in terms of Access Protocols and Mechanisms, Functionality, Performance, Reliability, Security, Convenience, Ease of use
- The workflow language is simpler
  - Many constructs (e.g., data transformation, synch vs. asynch invocations, timeouts) can be shifted from the language definition to the standard library of service types
- The composition language does not change...
  - ...when the system is extended to support future standards and new kinds of services

Service Types Supported by JOpera

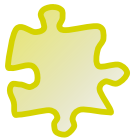


JOpera provides an *extension-point* for custom **service invocation plugins**

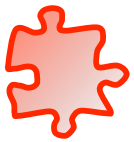


Web servers (HTTP/HTML)

Web Services (SOAP, WSIF)



Grid Services (WSRF)



Human activities

SQL Queries (JDBC)

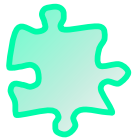
XML Transformations (XSLT, X-Path)

UNIX Commands

Windows

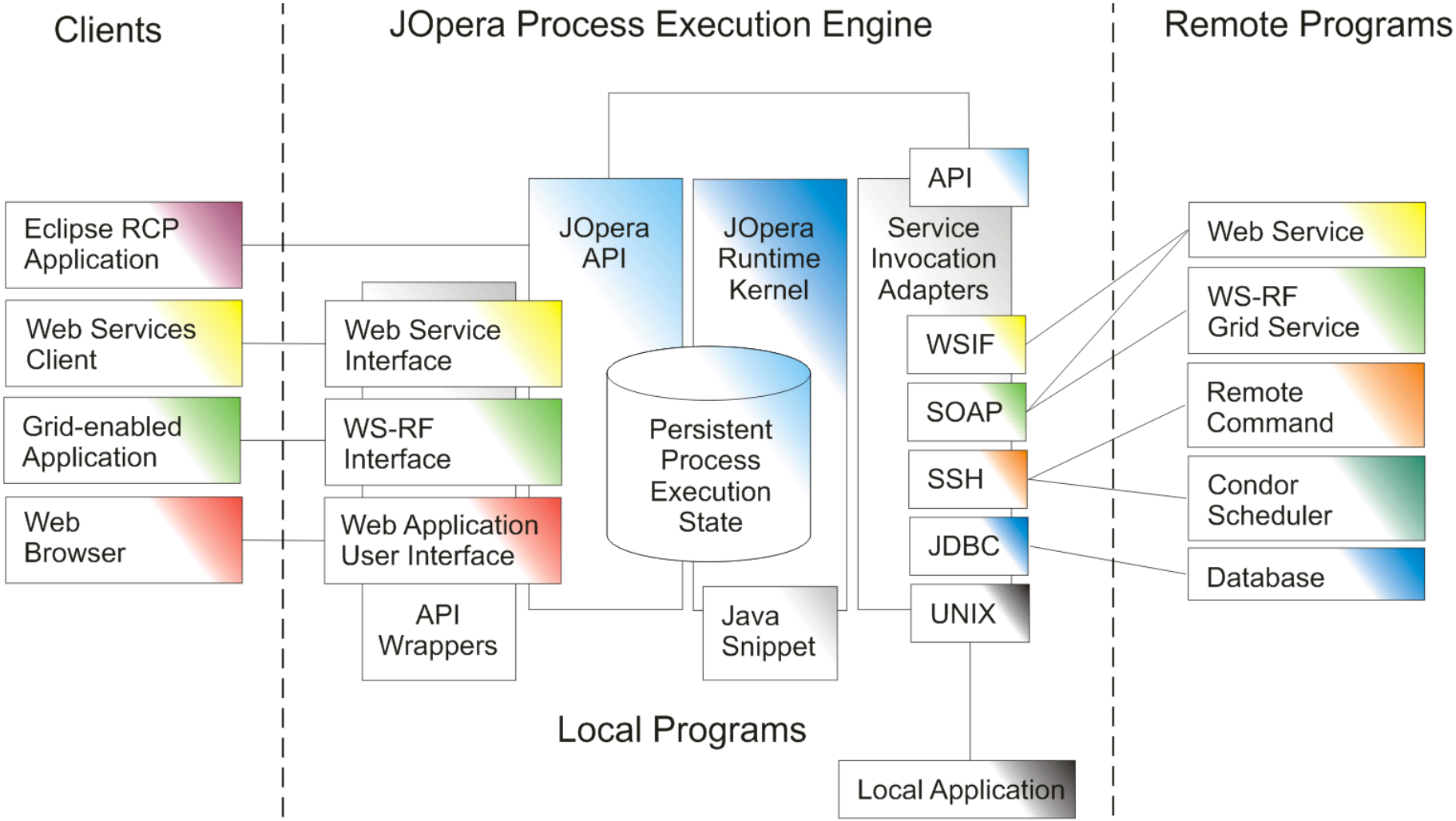
SSH

Java snippets



Java methods

Architecture of JOpera for Eclipse



Conclusions

- JOpera is a workflow tool for building distributed applications made out of heterogeneous parts
- Processes provide high level abstractions for specifying the behavior of such applications
- JOpera offers a completely **open**, flexible and extensible service composition platform
- JOpera currently focuses on manual composition with some syntax-based automatic tools

Outlook & Potential Interactions

- Model checking based on Syntax and Semantics (more advanced error detection)
- Using syntax and semantics for auto-completion (make smarter suggestions)
- A service invocation adapter to use WSMX for Dynamic Service Selection and Binding
- Grounding the planning results to run on the JOpera engine

References

[e-SCIENCE2005] Thomas Heinis, Cesare Pautasso, Oliver Deak, Gustavo Alonso, **Publishing Persistent Grid Computations as WS Resources**, accepted to the 1st IEEE International Conference on e-Science and Grid Computing (e-Science 2005), Melbourne, Australia, December 2005.

[ICWS2005] Cesare Pautasso, Thomas Heinis, Gustavo Alonso: **Autonomic Execution of Service Compositions**, In: Proc. of the 3rd International Conference on Web Services (ICWS 2005), Orlando, Florida, July 2005.

[ICAC2005] Thomas Heinis, Cesare Pautasso, Gustavo Alonso: **Design and Evaluation of an Autonomic Workflow Engine**, In: Proc of the 2nd International Conference on Autonomic Computing (ICAC-05), Seattle, Washington, June 2005.

[JVLC2005] Cesare Pautasso, Gustavo Alonso **The JOpera Visual Composition Language** Journal of Visual Languages and Computing (JVLC), 16(1-2):119-152, 2005

[VLDB/TES2004] Cesare Pautasso, Gustavo Alonso: **From Web Service Composition to Megaprogramming** In: Proceedings of the 5th VLDB Workshop on Technologies for E-Services (TES-04), Toronto, Canada, August 29-30, 2004.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

More information & download:
www.jopera.org

Available Today