ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IKS Information and Communication Systems Research Group

JOpera: Composing Service Oriented Architectures with Workflows

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

Gio Wiederhold

Peter Wegner

Stefano Ceri

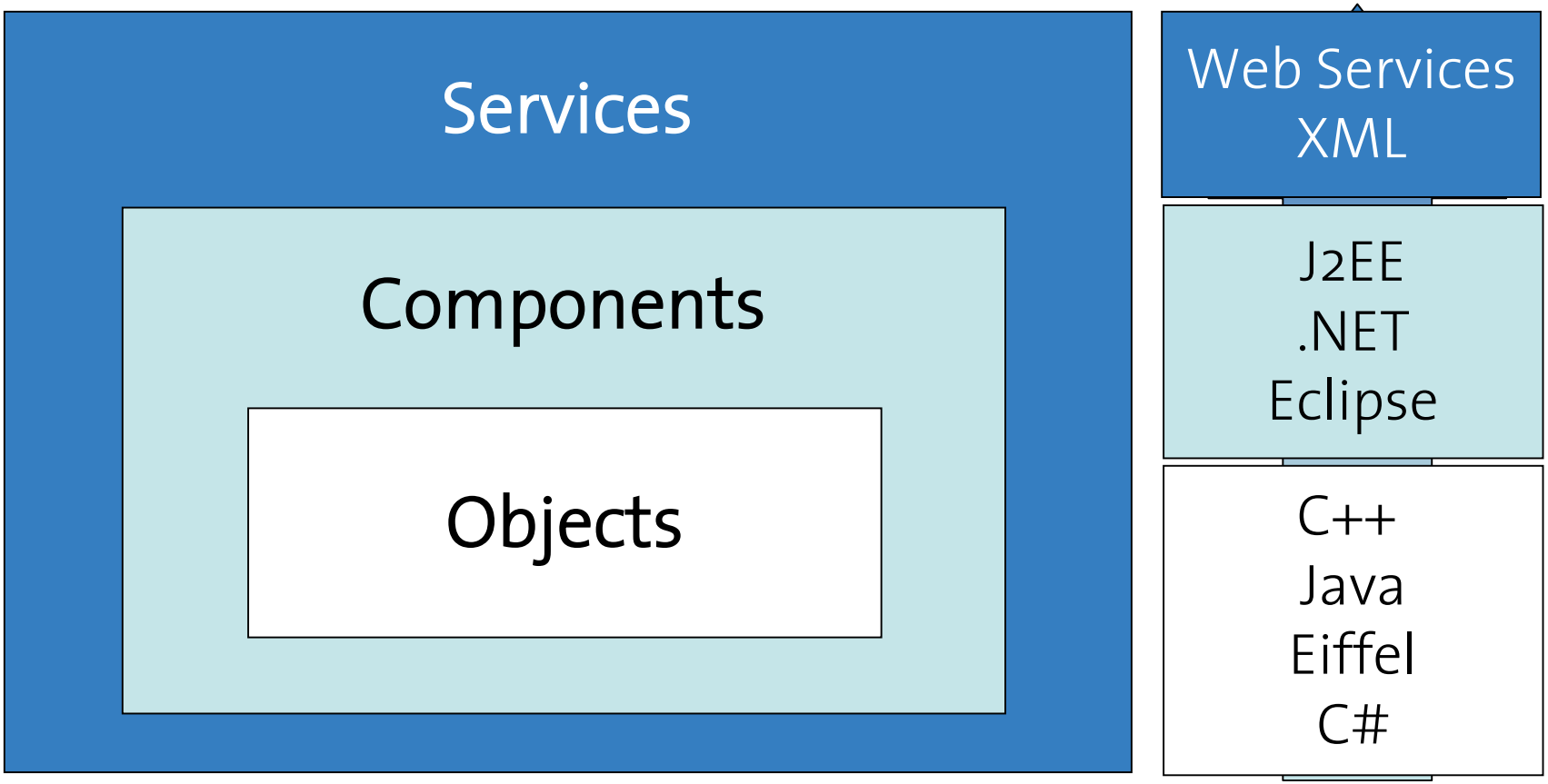
Toward Mega programming

***M**egaprogramming* is a technology for programming with large modules called *megamodules* that capture the functionality of services provided by large organizations like banks, airline reservation systems, and city transportation systems. Megamodules are internally homogeneous, independently maintained software systems managed by a community with its own terminology, goals, knowledge, and programming traditions. Each megamodule describes its externally accessible data structures and operations and has an internally consistent behavior. The concepts, terminology, and interpretation paradigm of a megamodule is called its *ontology*.

Gio Wiederhold
Peter Wegner
Stefano Ceri

Communications of the ACM
Volume 35
Issue 11
November 1992
Pages: 89 - 99

New Abstractions for Application Integration



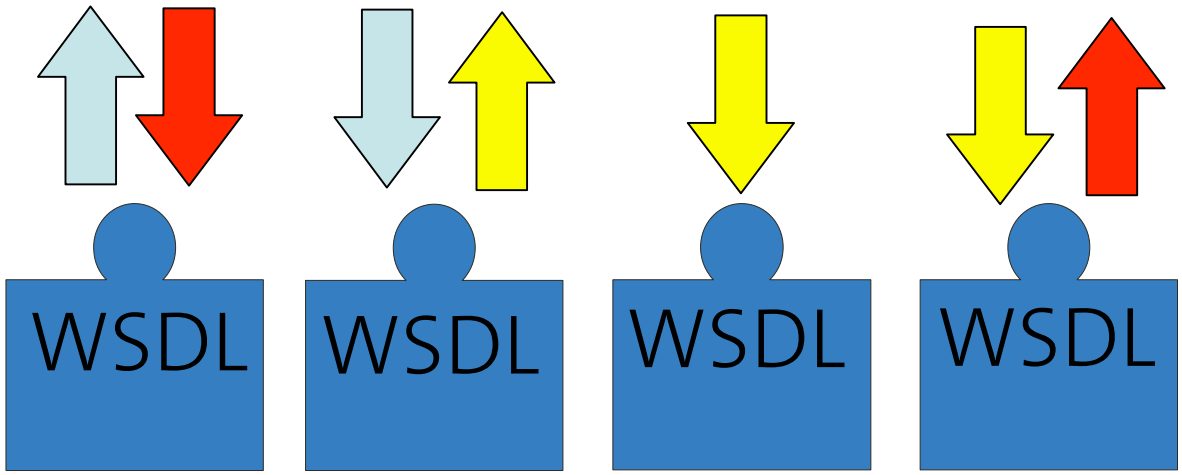
Service Interaction Patterns

Remote Procedure Call

Asynchronous Messaging

The Problem of Service Composition

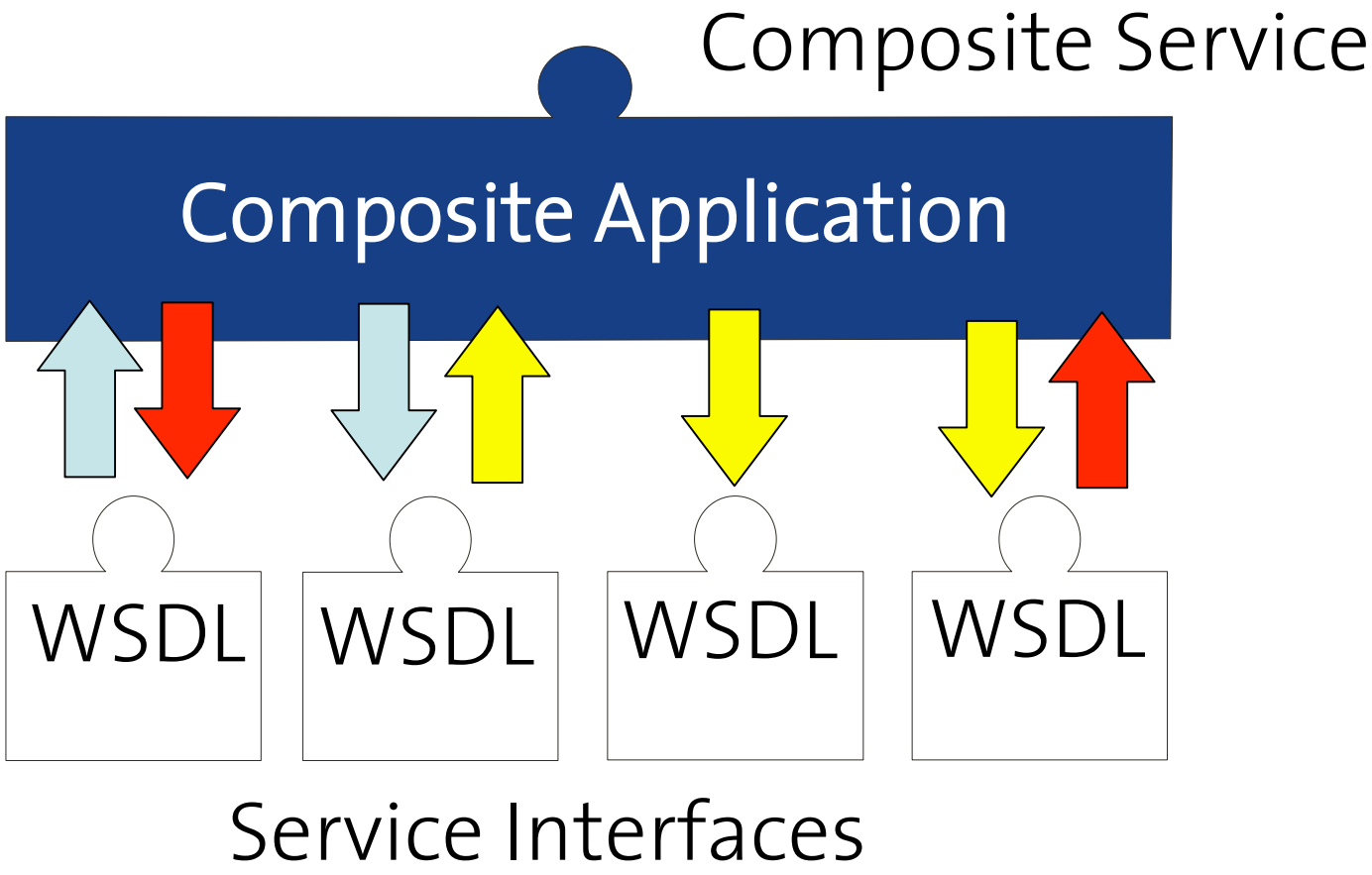
- How to build an application by reusing existing components delivered as a service?
- How to script the exchange of asynchronous messages between a set of services?



Service Interfaces

The Problem of Service Composition

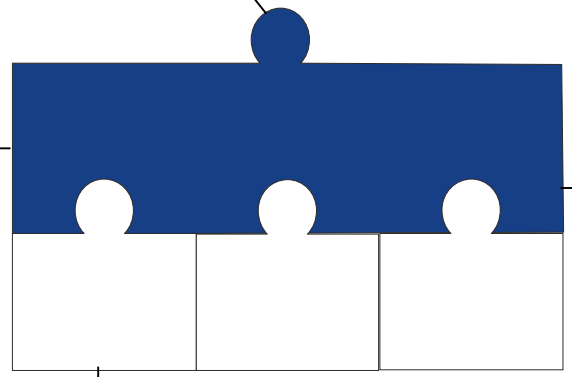
- Where to put the composition logic?



Service Composition Problems

How to publish a composition?

How to model a composition?



How to execute a composition?

What kind of services can be composed?

Service Composition with JOpera

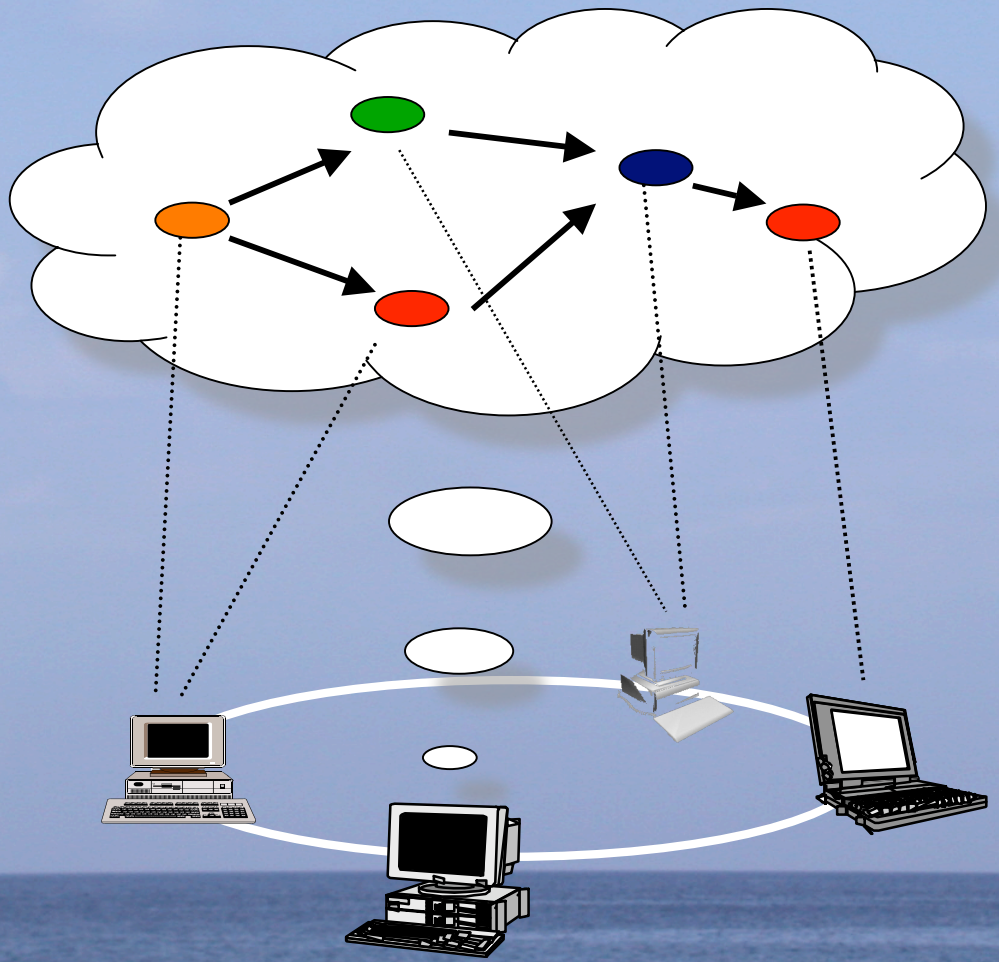
How to publish a composition?
Automatically

How to model a composition?
(Work)Flow Language

How to execute a composition?
Autonomic Engine

What kind of services can be composed?
Web Services and many more...

Modeling SOA Workflows with JOpera



Modeling Service Compositions

- What are good abstractions for modeling a service composition?
- Business Process Modeling Languages
  - Service invocation treated as *task*
  - *Control flow* (branches, loops, synchronization)
  - *Data flow* (and data *transformations*)
  - *Exception Handling*
  - *Dynamic Late Binding*
- Syntax
  - Textual, Visual, XML, UML

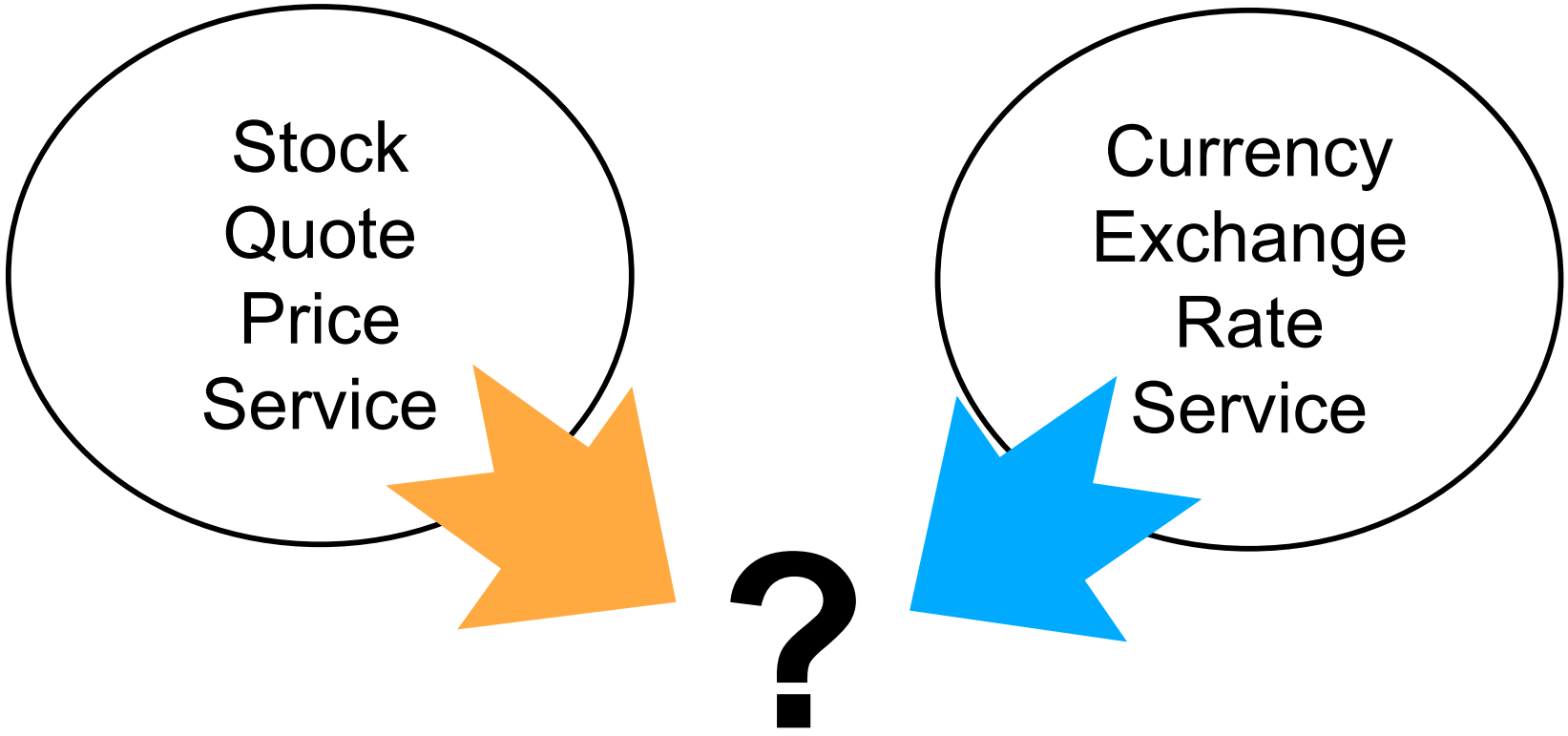
[HCC2003]

Workflow Lifecycle in JOpera

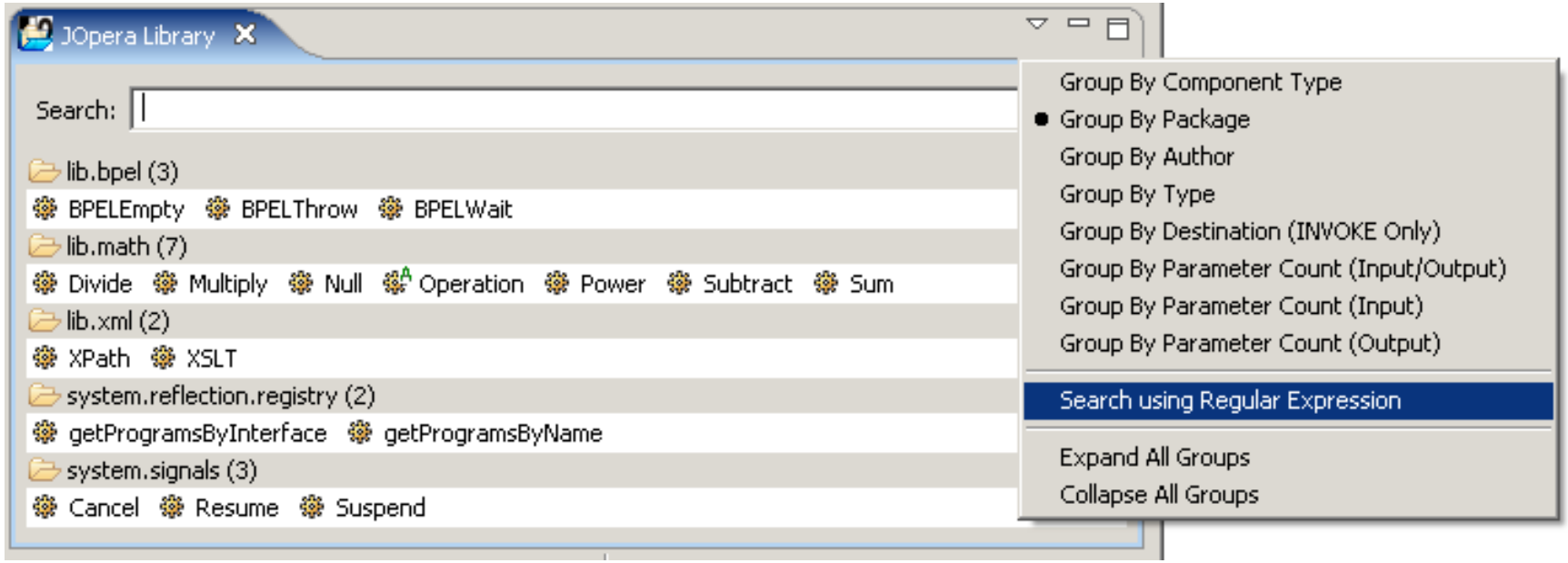
1. Select component services from a **library**
2. Build a process using a drag, drop and connect **visual** environment
3. Run, Test, and Debug the execution **within the same visual environment**
4. Deploy, Manage, Monitor, and Steer the execution of workflows in production
5. Publish the workflow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

Demo 1

- Stock Quote Currency Conversion

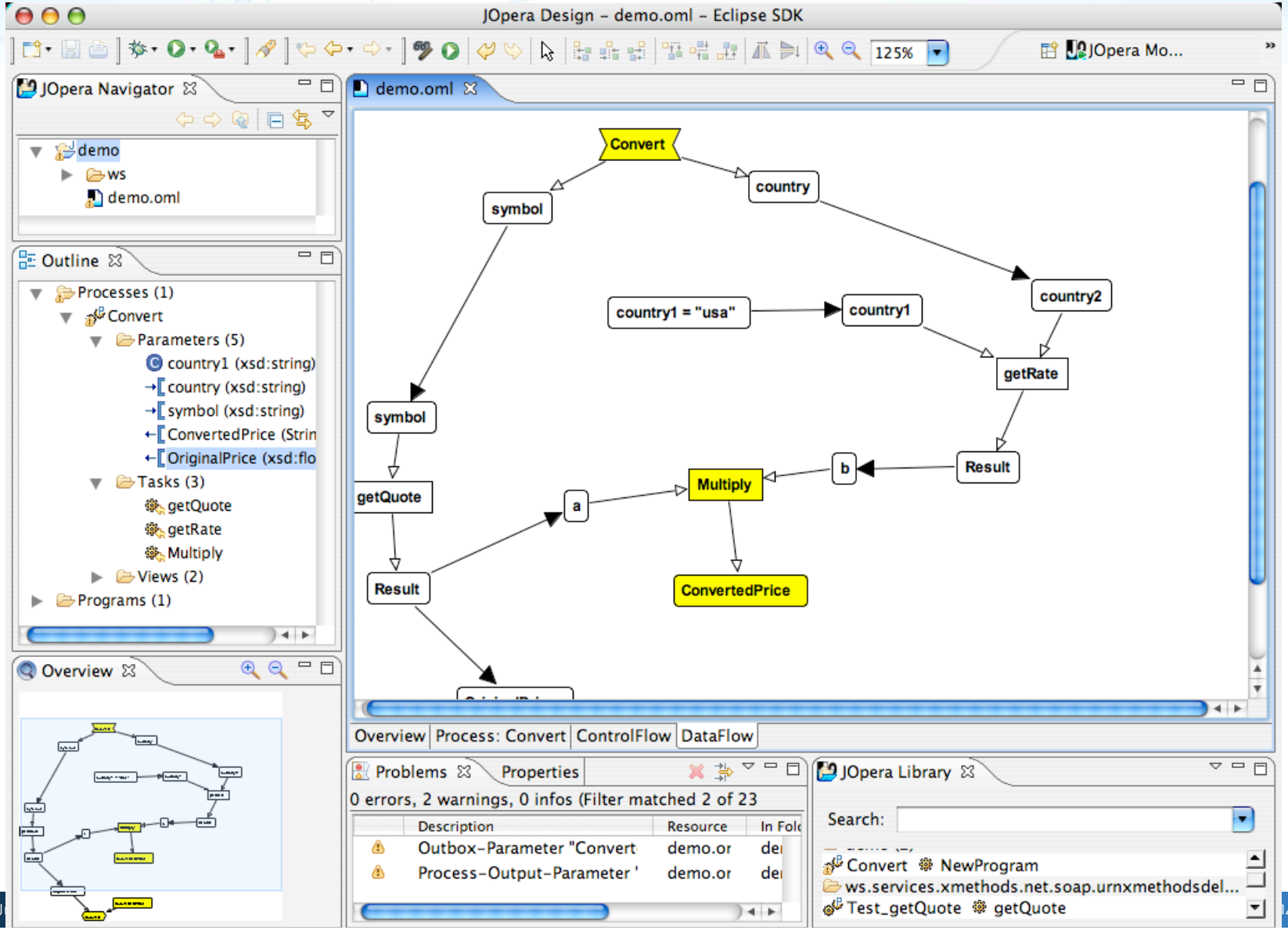


Service Library

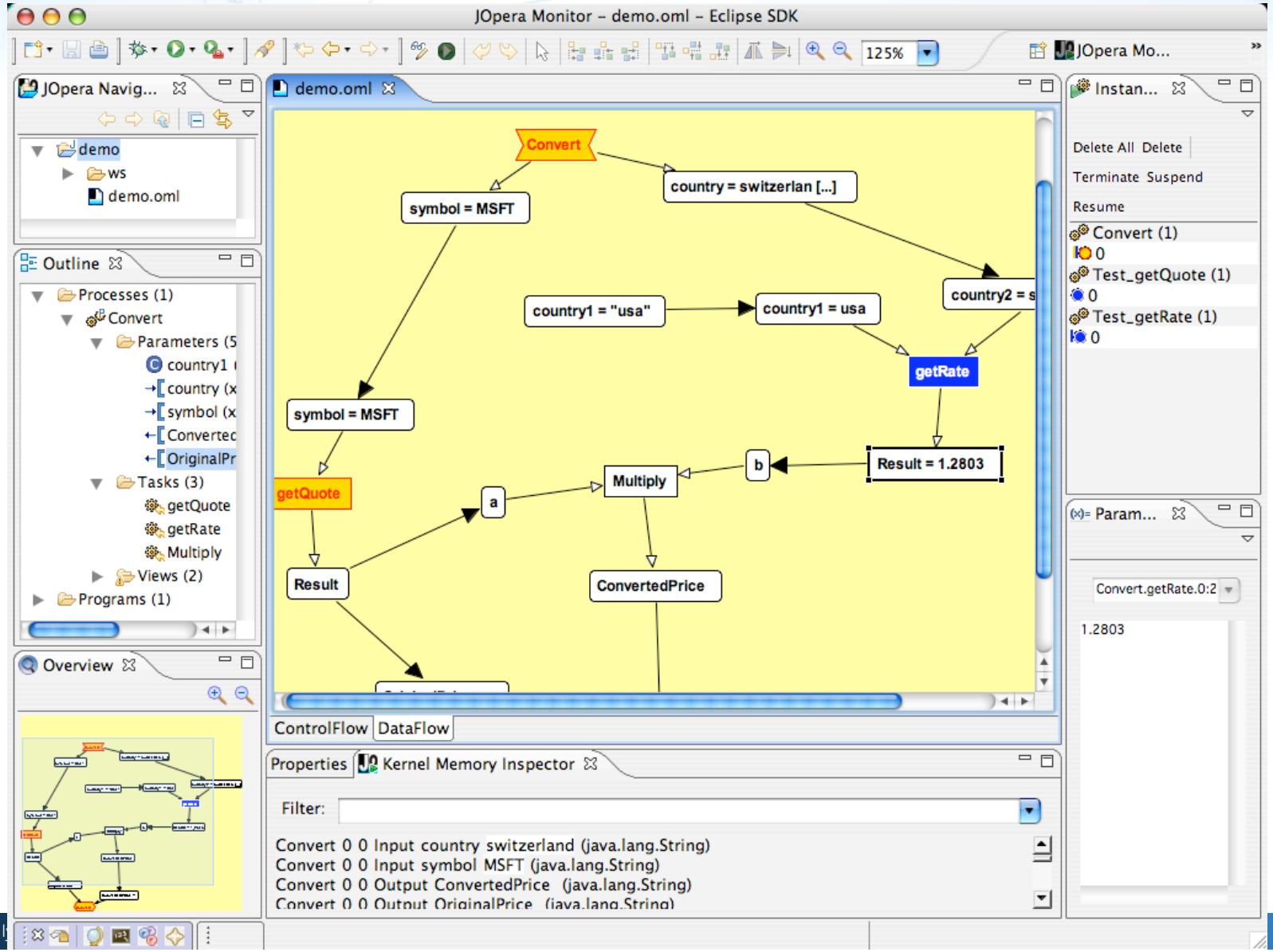


1. Search services as you type (also with regex)
2. Group services by different (orthogonal) criteria

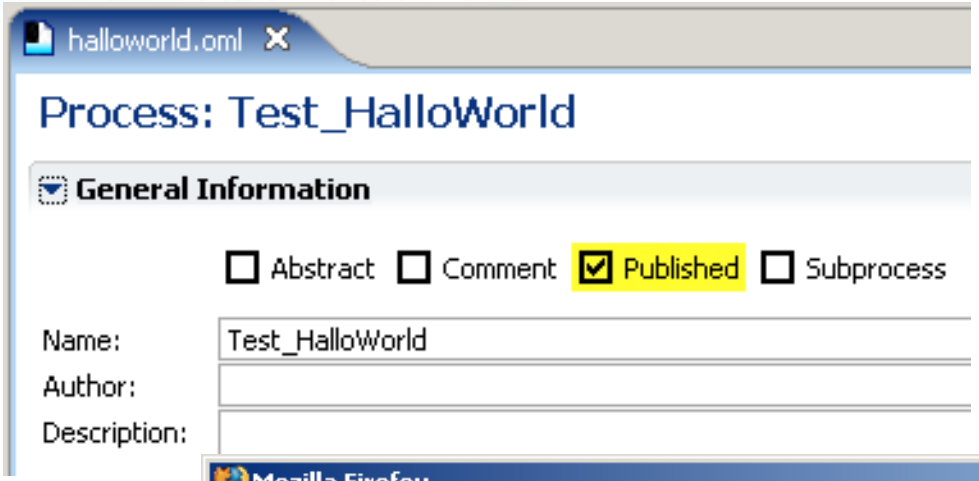
Drag, Drop and Connect



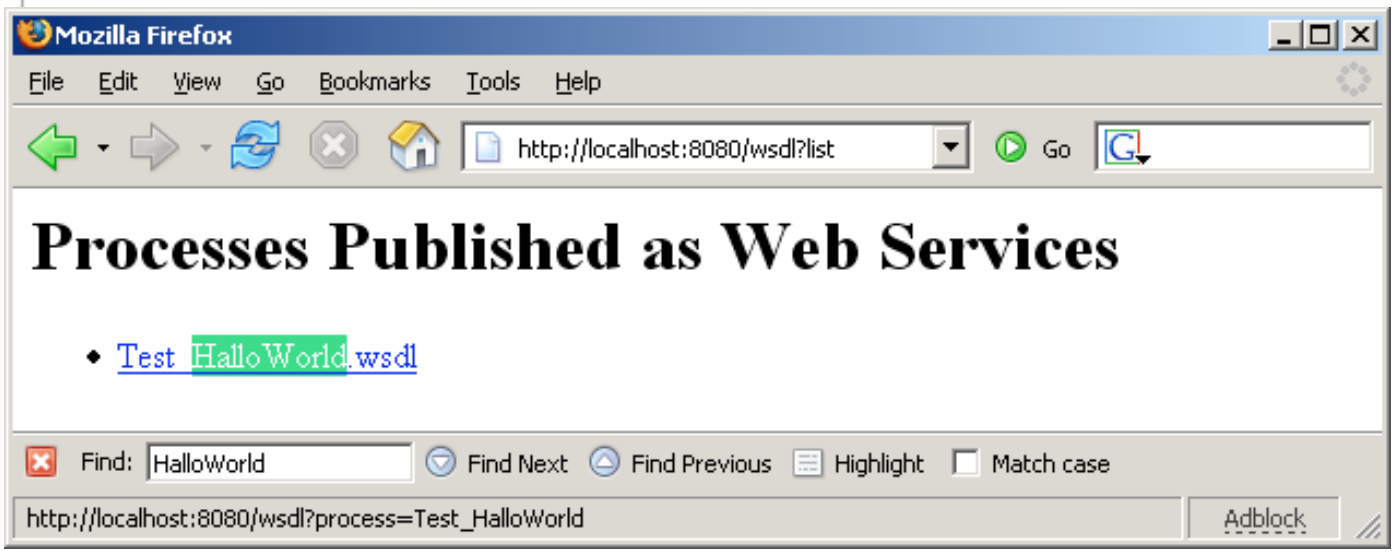
Run, Monitor, Steer and Debug



Publish as a Web/Grid service



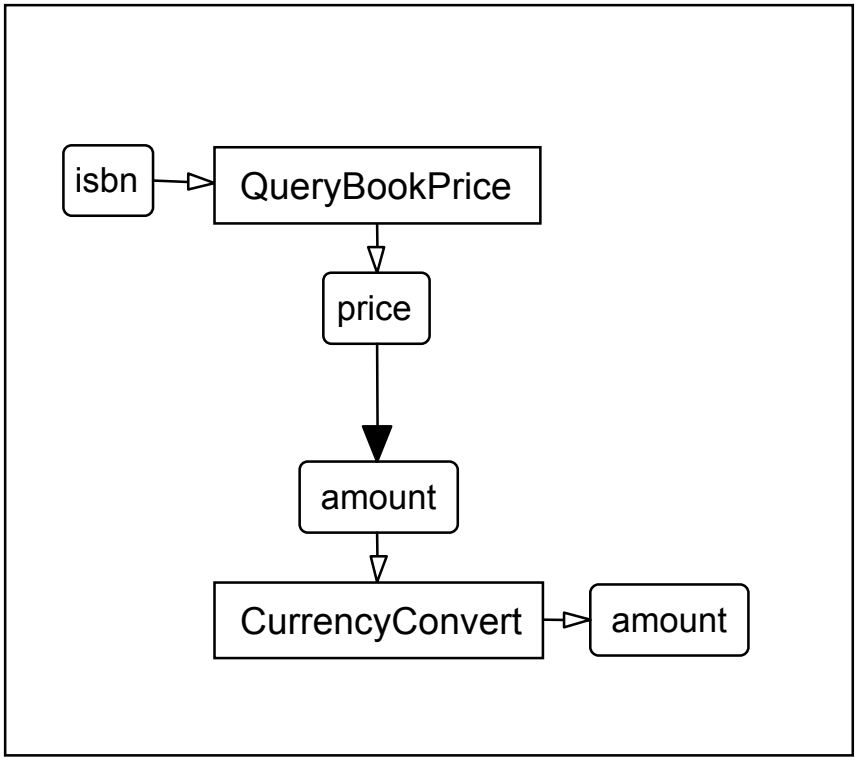
With one mouse click!

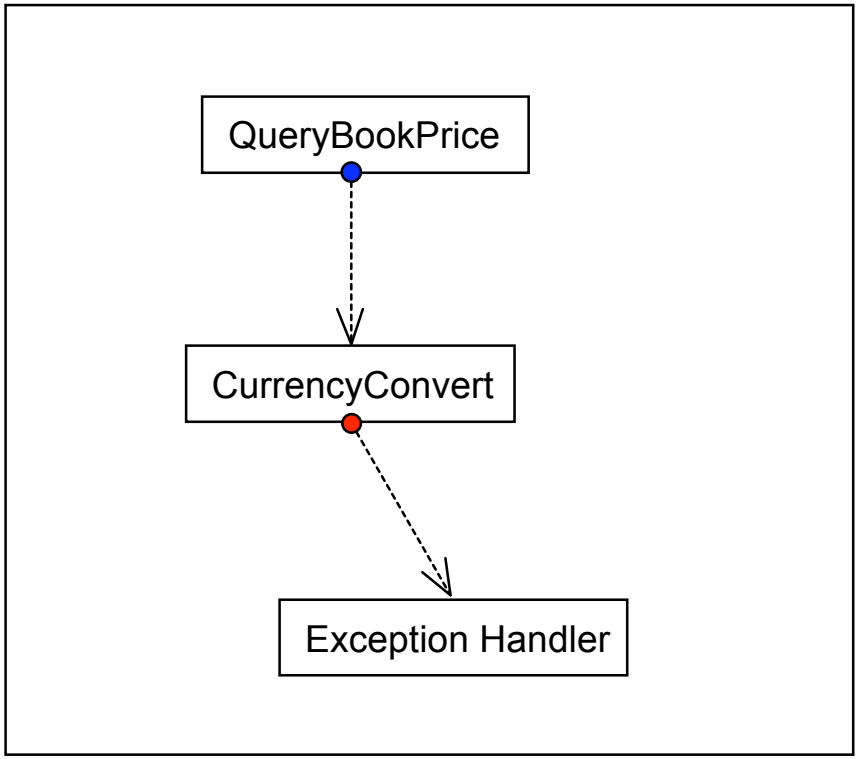


[e-Science2005]

JOpera Visual Composition Language Overview

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



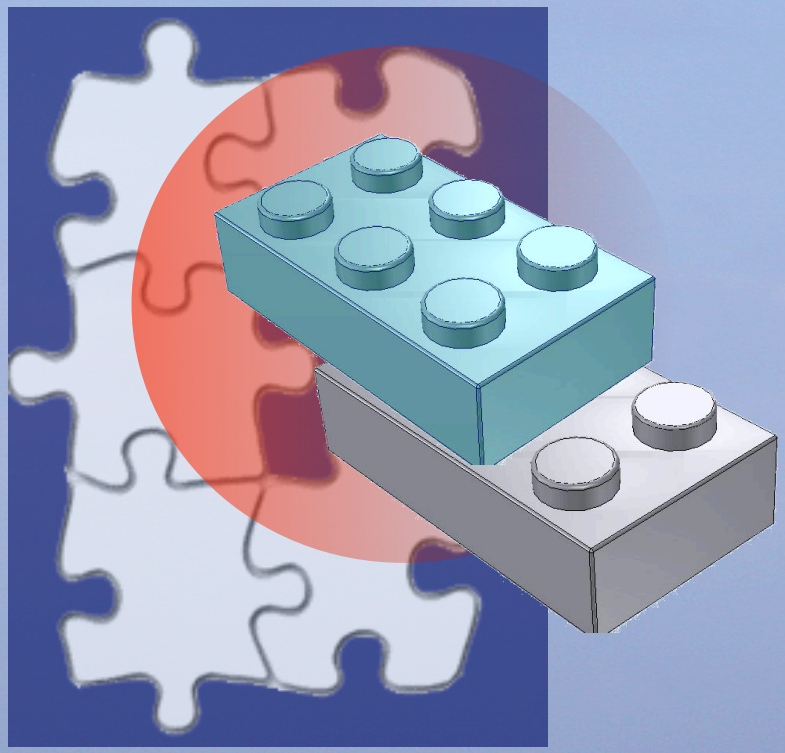


JOpera Visual Composition Language Features

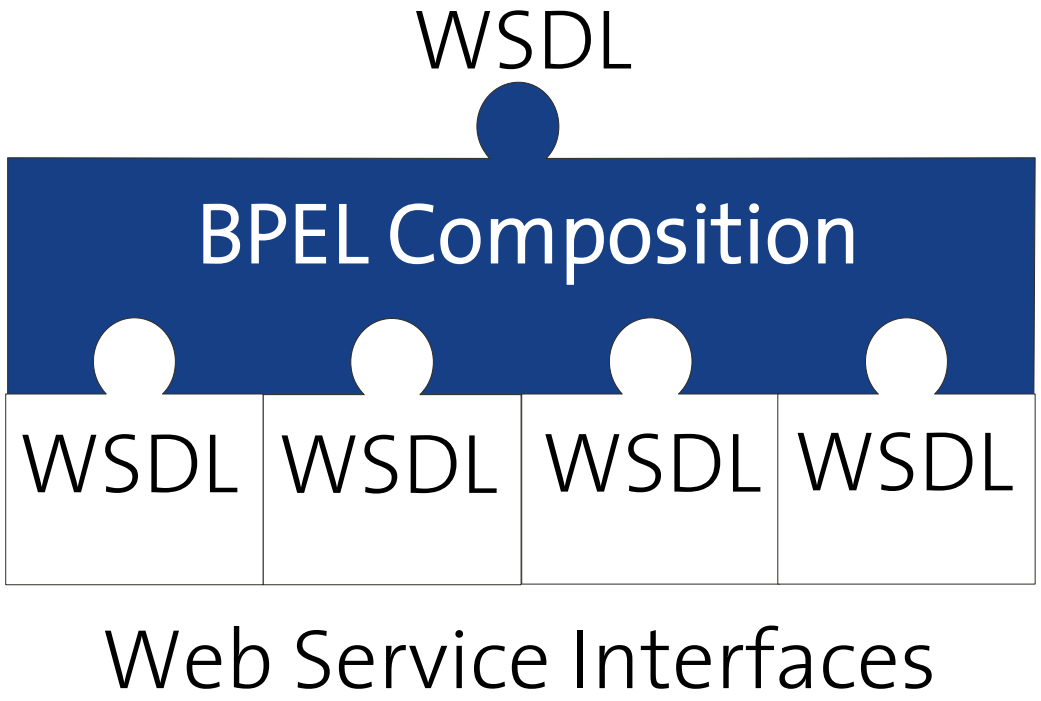
- Processes model generic service composition
  - **Data flow** as the primary representation
  - Explicit **control flow** (branch, synchronization, exception handling, loops, pipeline, workflow patterns)
- **SubProcesses**: Modularity, Nesting and Recursion
- **First order functions**
  - Map (parallel/sequential/discriminator)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

[JVLC2005]

What kinds of Services can you compose with JOpera?



What kind of services can you compose with WS-BPEL?



Assumption:
Web Services (SOAP/WSDL) are the only kind of services to be composed

Problem:
extensions to the BPEL standard are needed to support code snippets (BPELJ) and human tasks (BPEL4PEOPLE)

How NOT to deal with heterogeneity

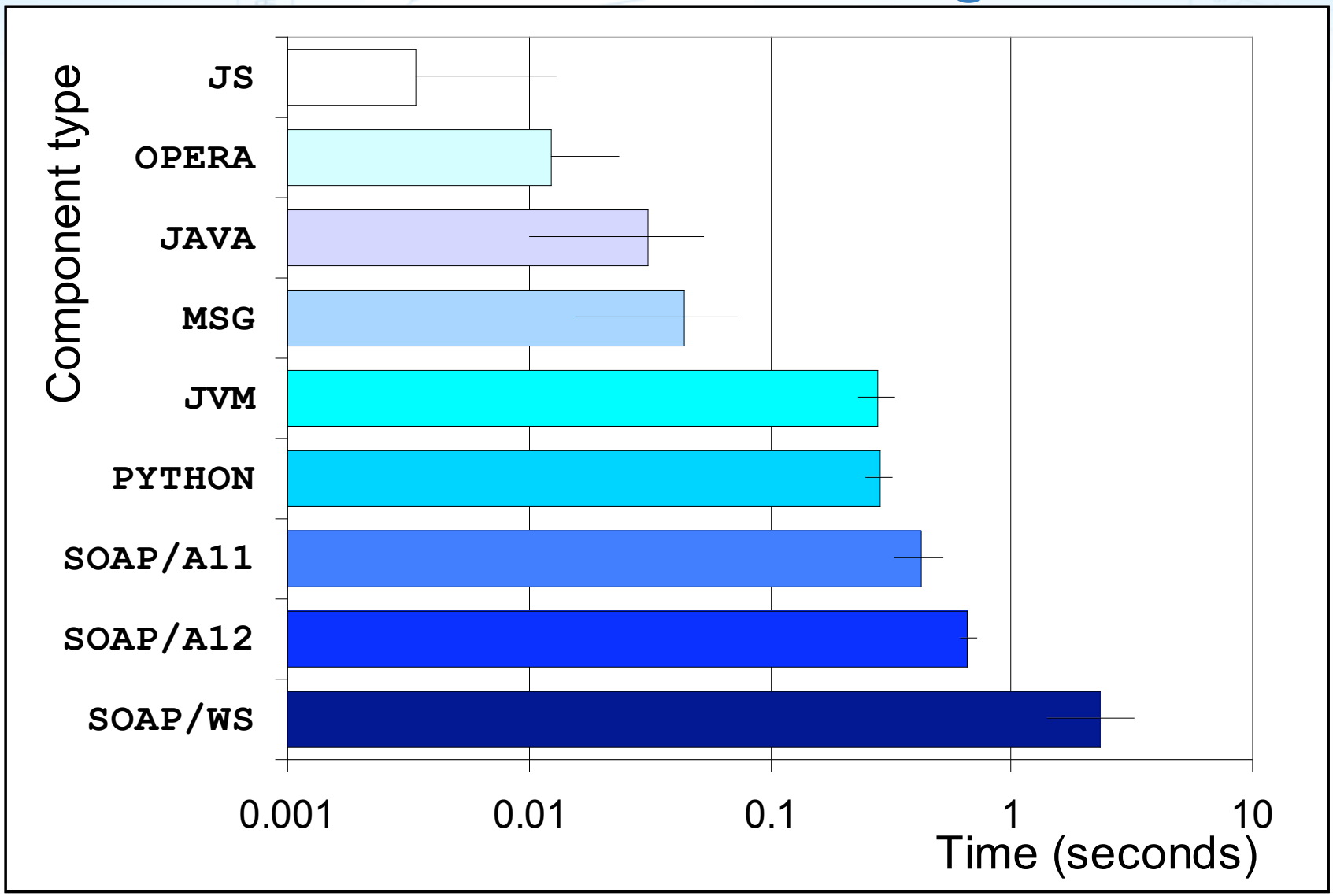
1. Assume that all services to be orchestrated will conform to one standard
2. Force all existing implementations to be wrapped to comply with that standard
3. Modify the workflow language to extend its support to other standards

(Example: BPEL4WS, BPELJ, BPEL#, BPEL4PEOPLE)

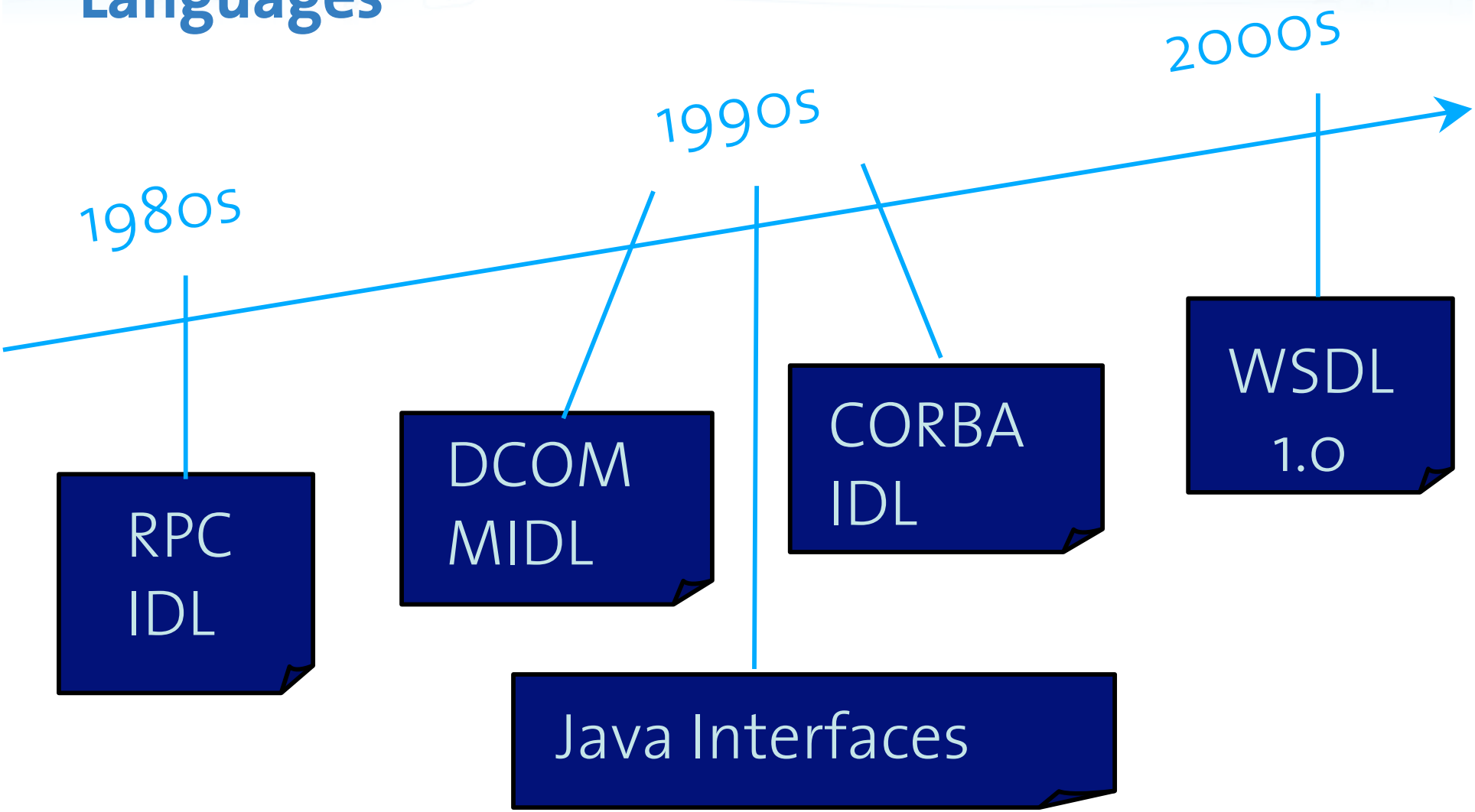
Problems of composing *only* Web Services

- 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)



A Brief History of Interface Description Languages



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

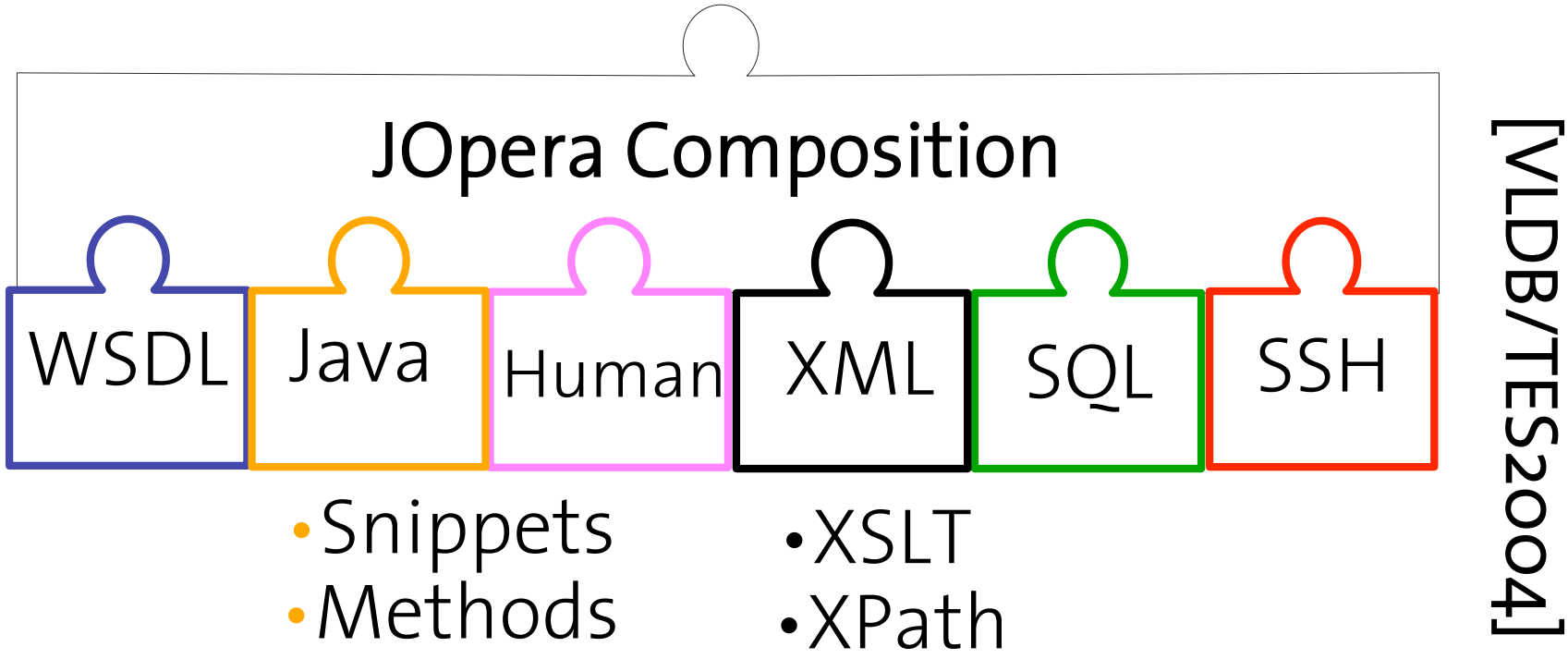
[VLDB/TES2004]

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

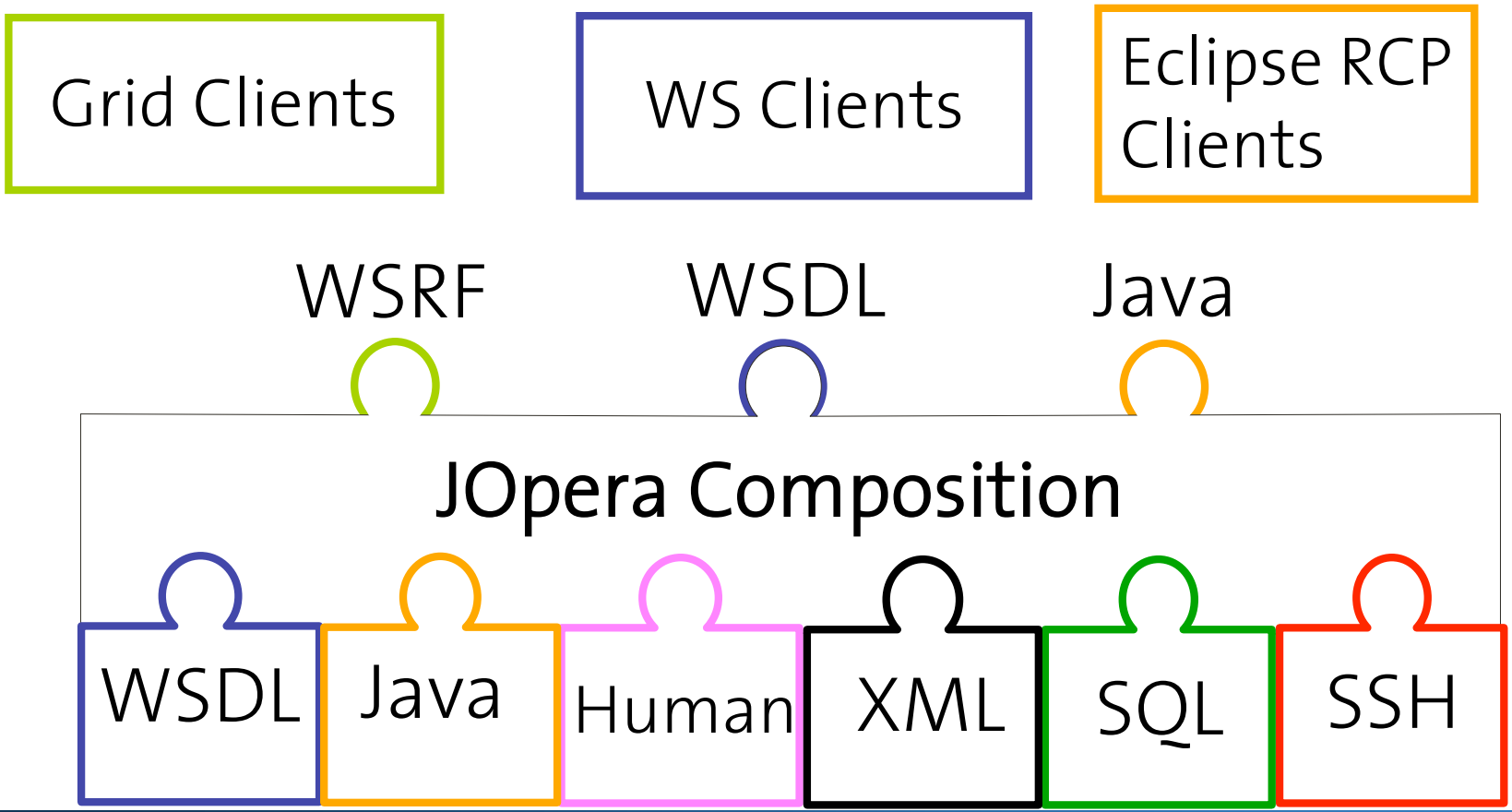
Dealing with heterogeneity in JOpera

- The JOpera composition language does not have to be changed when adding a new kind of service



Publishing a composition with JOpera

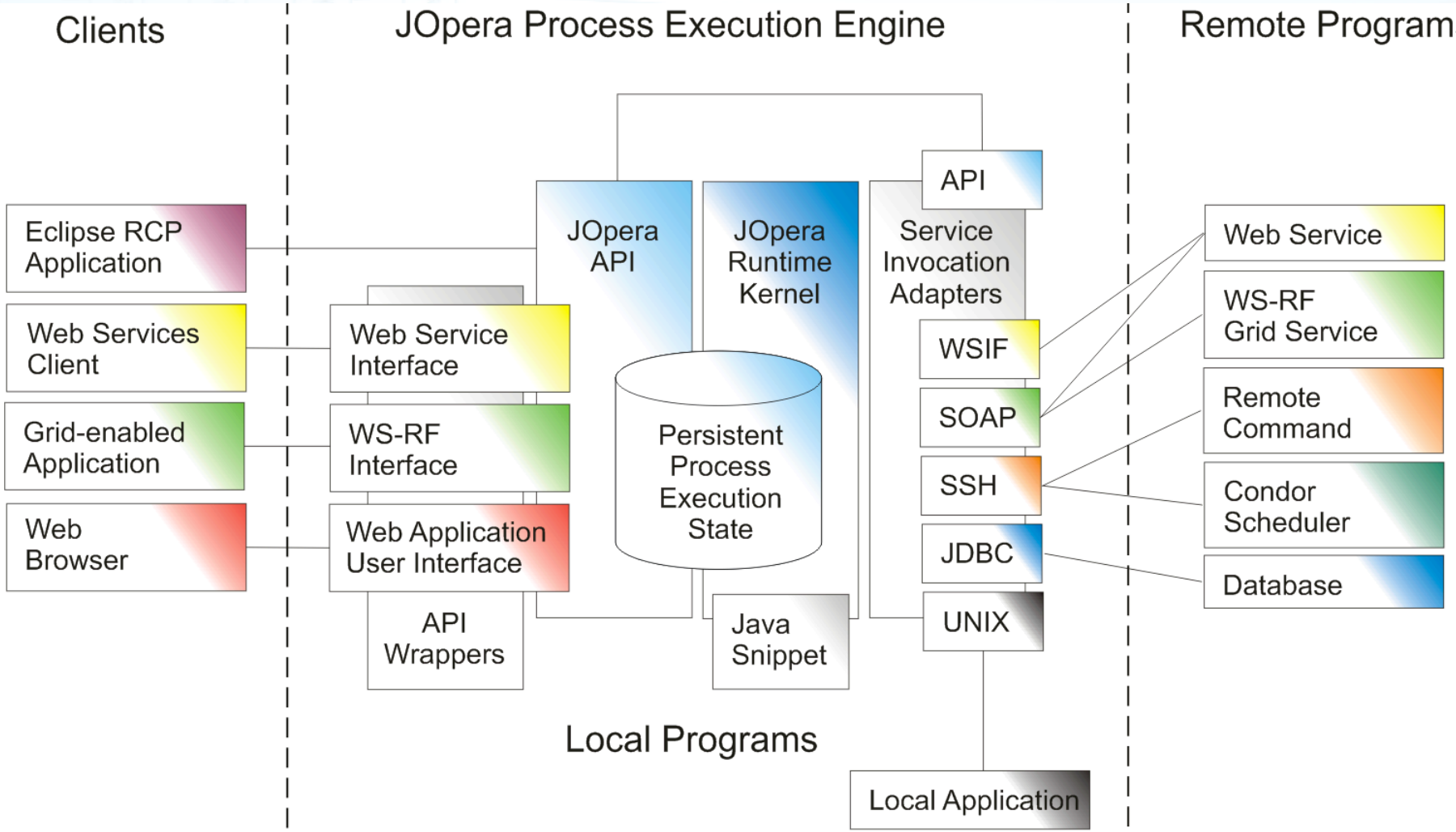
- JOpera processes are automatically published to clients using a variety of access protocols



[eScience2005, CCGrid2006]


Running Workflows
with JOpera for Eclipse

Architecture of JOpera for Eclipse



Conclusion

- **Modeling** service composition behavior
  - Workflow-based **composition language** (Visual & XML)
  - Development and Debugging tools for Eclipse
  - Composition not limited to Web services
- **Execution** of the composition models
  - Efficiency (compiled to Java bytecode)
  - Distributed engine (on a cluster of computers)
  - Autonomic platform (self-healing, self-tuning)
  - Extensibility (Eclipse plug-ins to provide custom service publishing and invocation adapters)

References on the language

[VL/HCC2005] Cesare Pautasso, **JOpera: an Agile Environment for Web Service Composition with Visual Unit Testing and Refactoring**, In Proceedings of the 2005 IEEE Symposium on Visual Languages and Human Centric Computing (VL/HCC'05), Dallas, TX, September 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.

[HCC2003] Cesare Pautasso, Gustavo Alonso: **Visual Composition of Web Services** In: Proc of the 2003 Symposia on Human Centric Computing Languages and Environments (HCC 2003), Auckland, New Zealand, Oct 2003.

References on the system

[CCGrid2006] Thomas Heinis, Cesare Pautasso, Gustavo Alonso, **Mirroring Resources or Mapping Requests: implementing WS-RF for Grid workflows**, accepted to the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid2006), Singapore, May 2006.

[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.

[IJET'04] C. Pautasso, G. Alonso **JOpera: a Toolkit for Efficient Visual Composition of Web Services** International Journal of Electronic Commerce (IJEC), 9(2):107-141, Winter 2004/2005

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

More information & download:
www.jopera.org

Available Today