ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik
Computer Scienc

IKS Information and Communication Systems Research Group

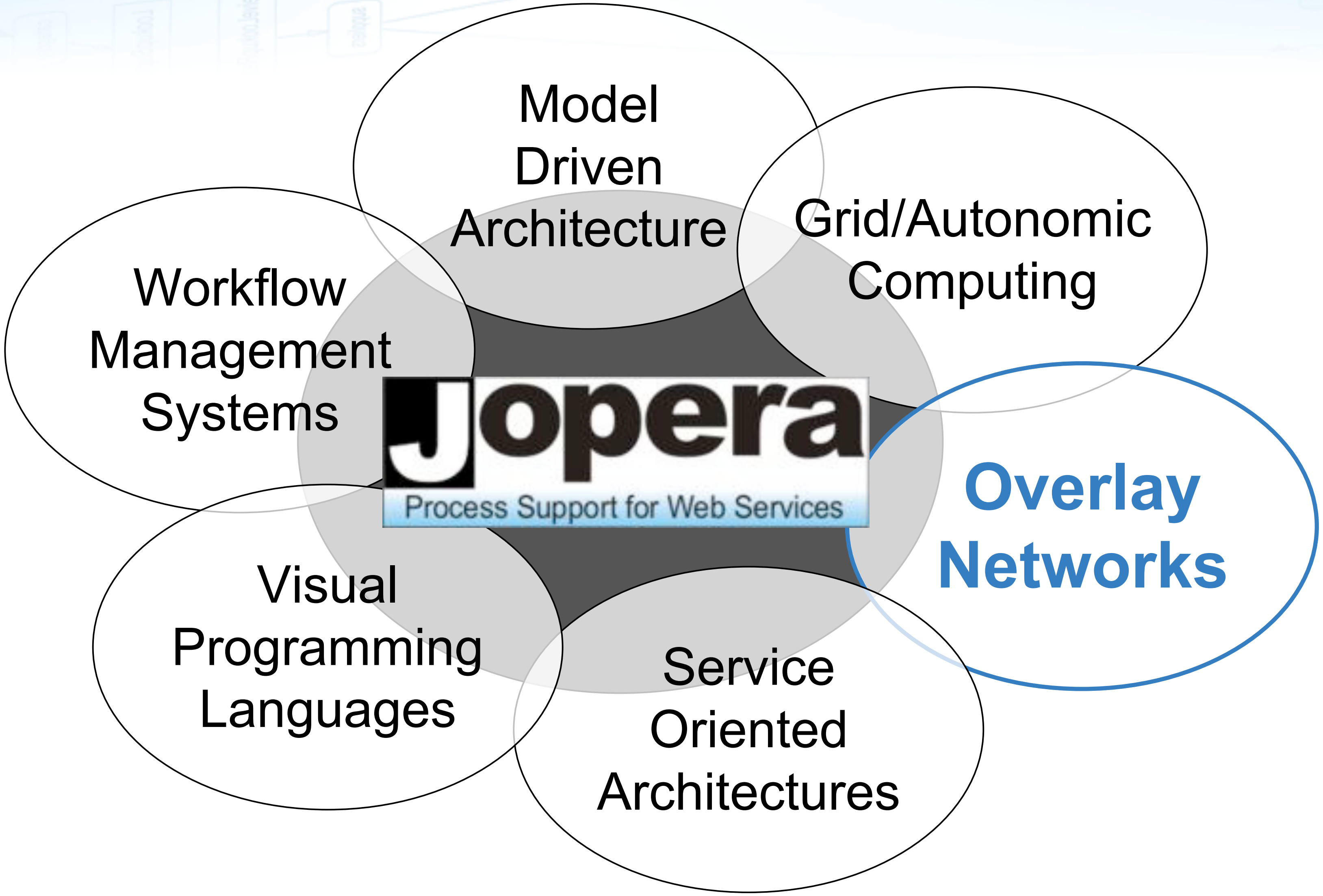
Towards Service Orchestration in Overlay Networks with JOpera

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

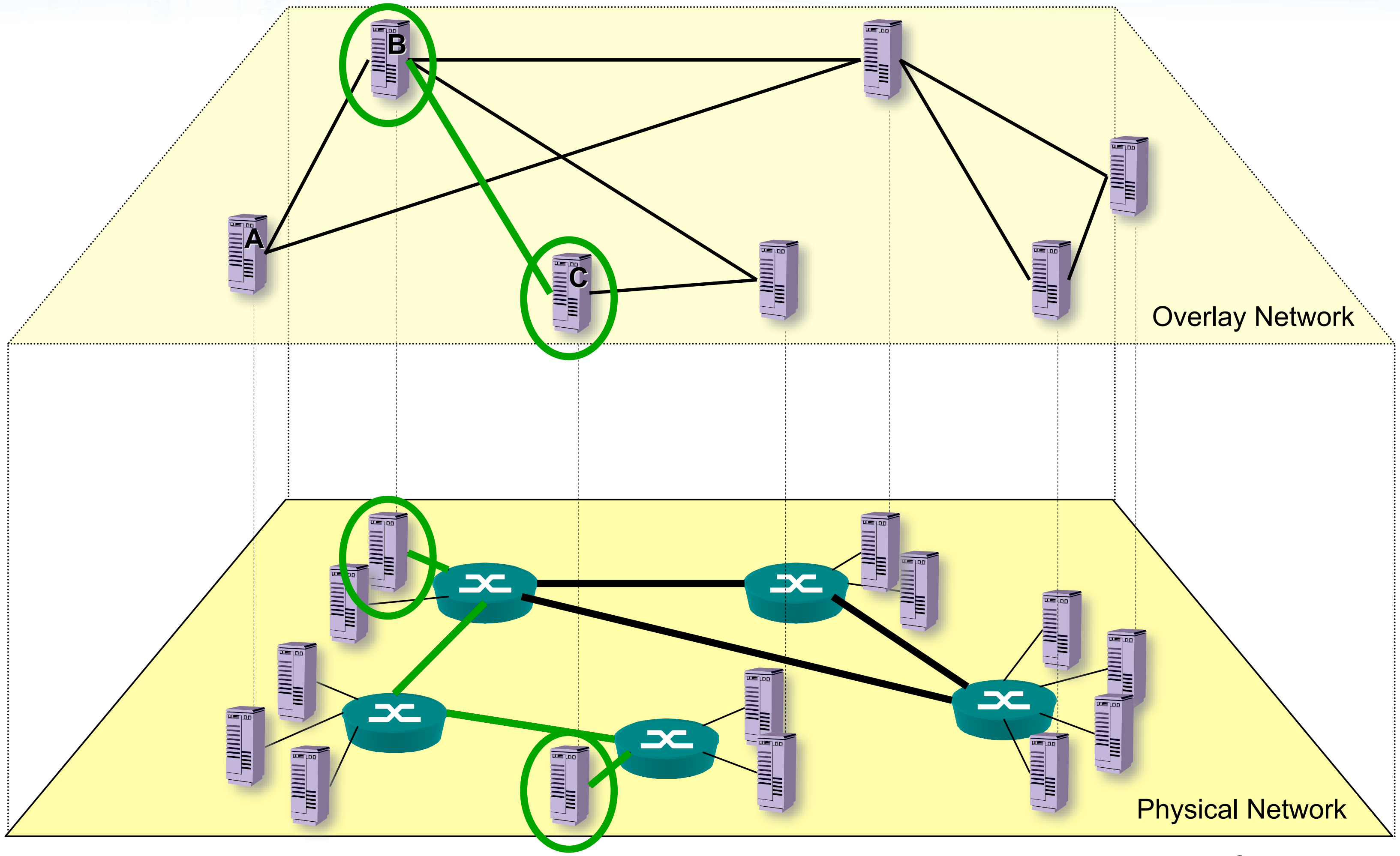
JOpera is kindly supported by:

- ETH Zurich
  - **IKS Group**, Prof. Gustavo Alonso (since 2000)
- European Union
  - **ADAPT** - Middleware Technologies for Adaptive and Composable Distributed Components (finished 2005)
  - **SODIUM** - Service Oriented Development in a Unified Framework (until 2007)
  - **AEOLUS** FET Project - Algorithmic Principles for Building Efficient Overlay Computers (until 2009)
- Hasler Stiftung
  - DICS Project: **Dependable Computing in Virtual Laboratories** (finished 2005)

My Research Interests

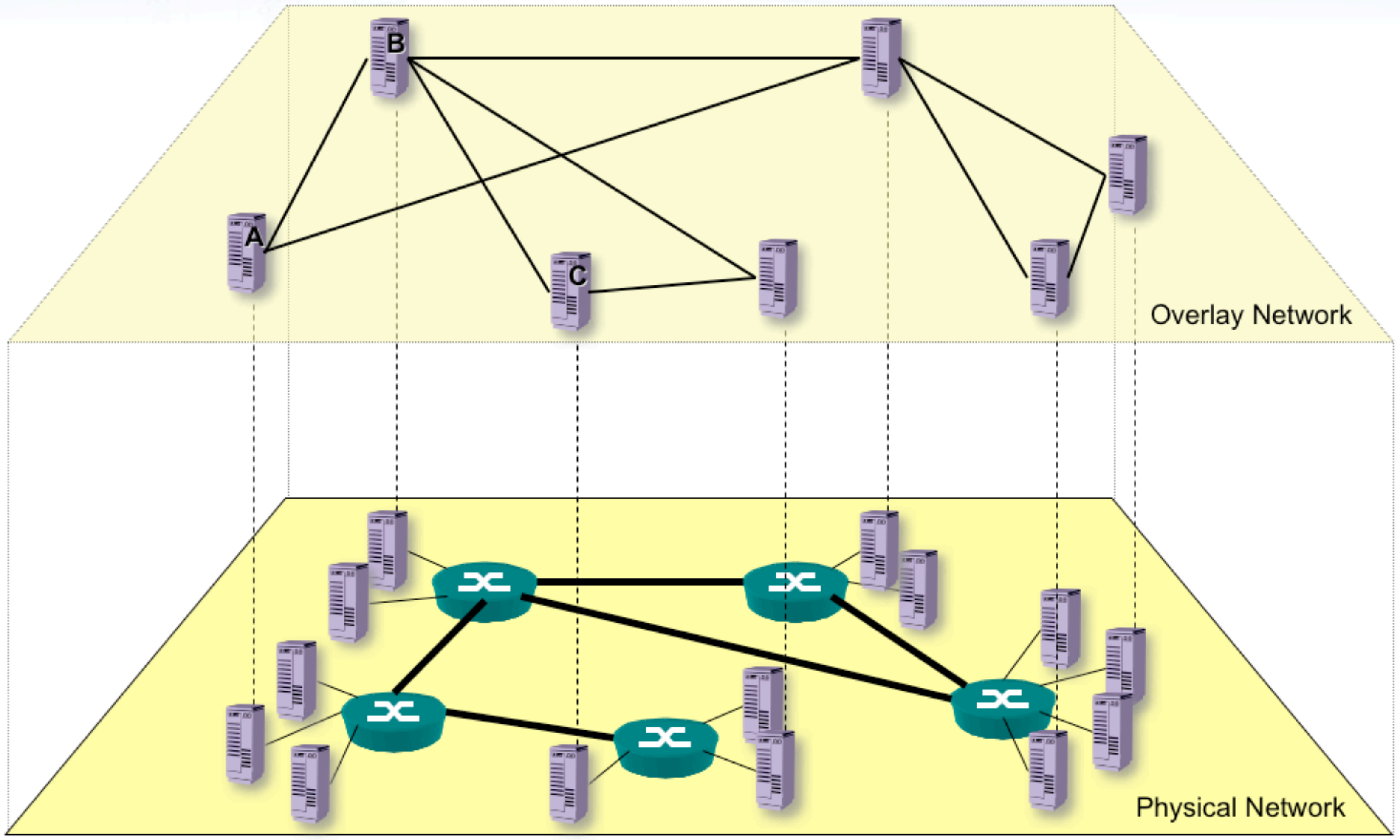

Model Driven Architecture
Grid/Autonomic Computing
Workflow Management Systems
JOpera
Process Support for Web Services
Overlay Networks
Visual Programming Languages
Service Oriented Architectures

Context for this talk: Overlay Networks

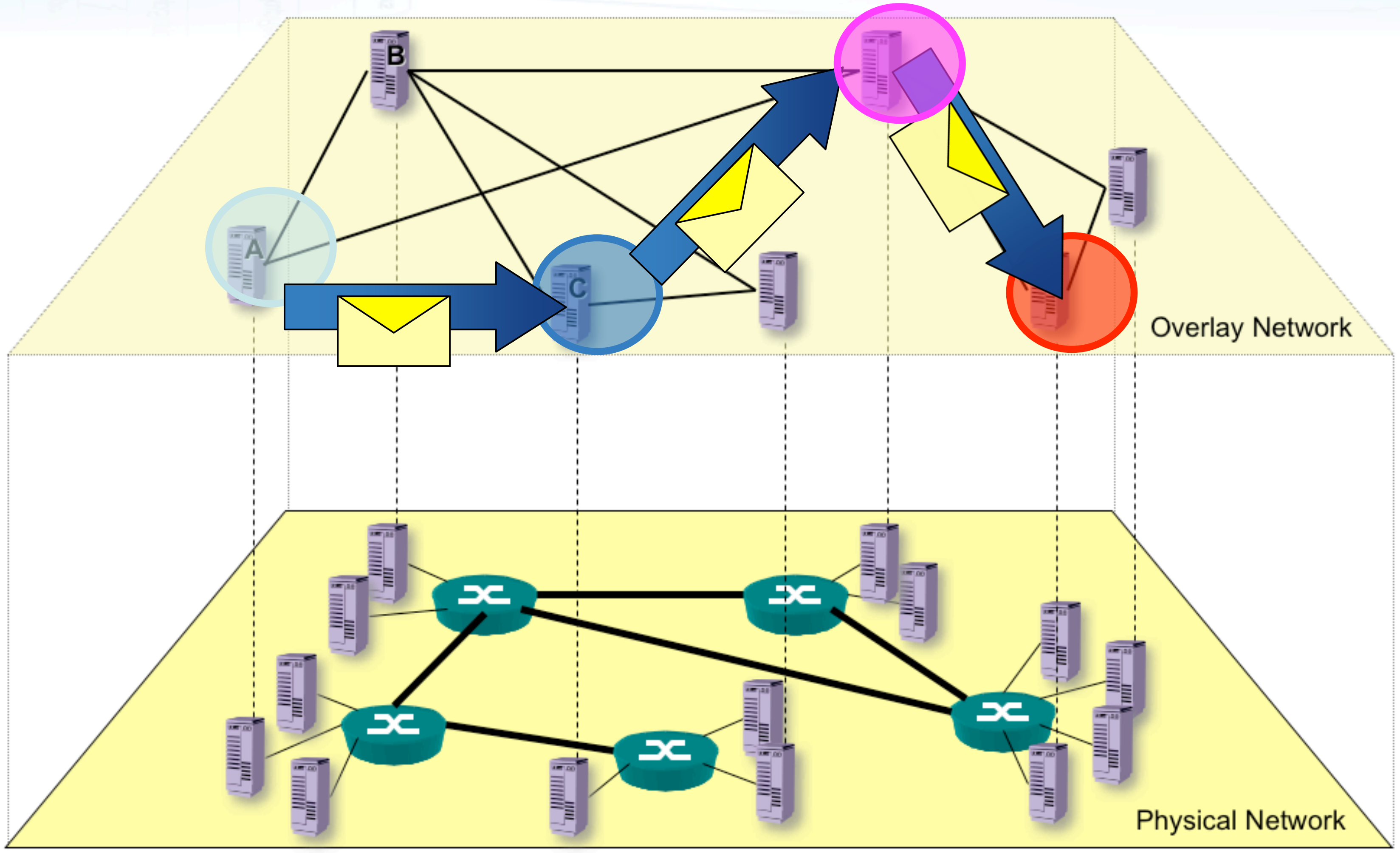


Courtesy of MASCOTTE

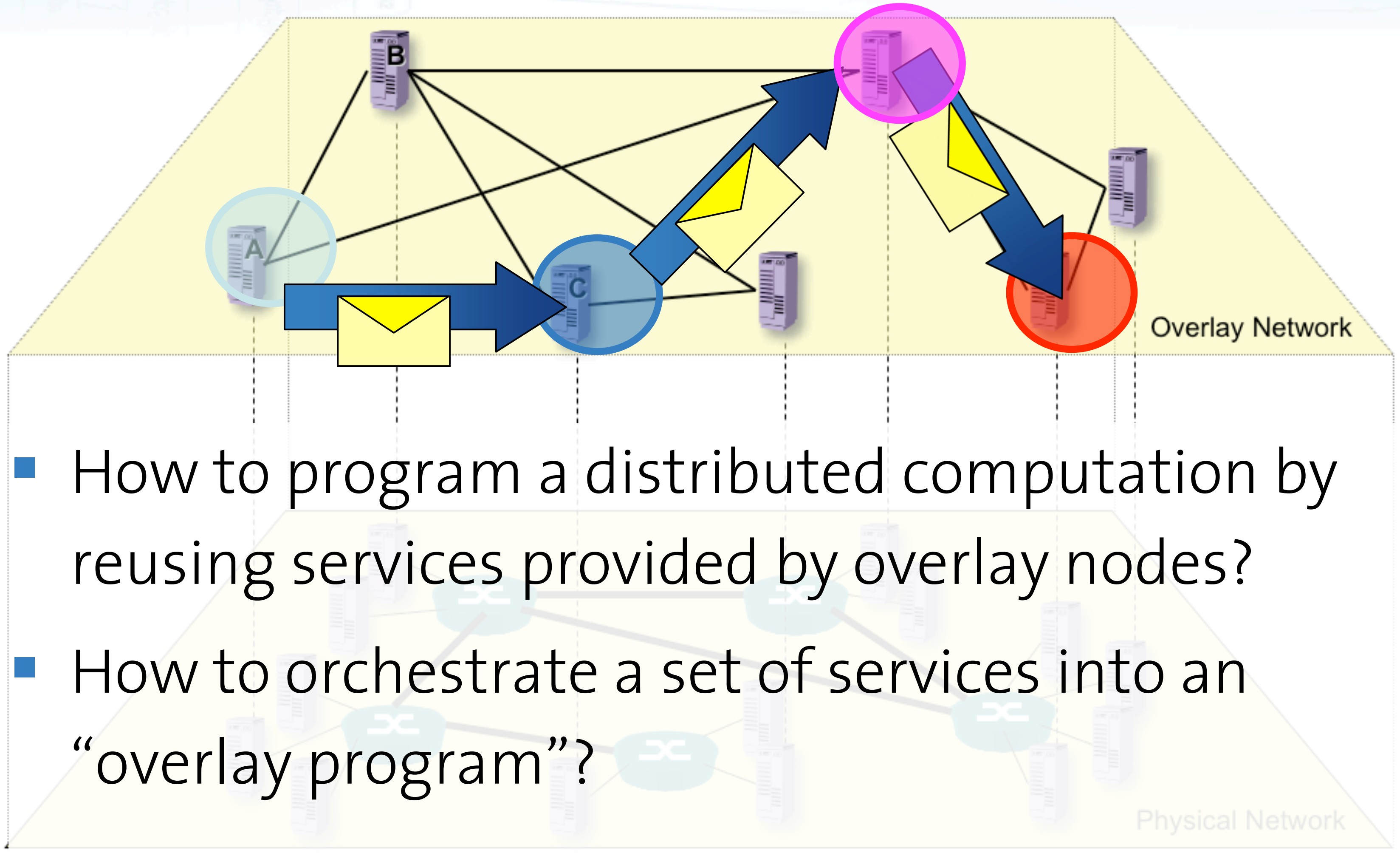
Challenge: How to Program an Overlay Network?



Challenge: How to Program an Overlay Network?



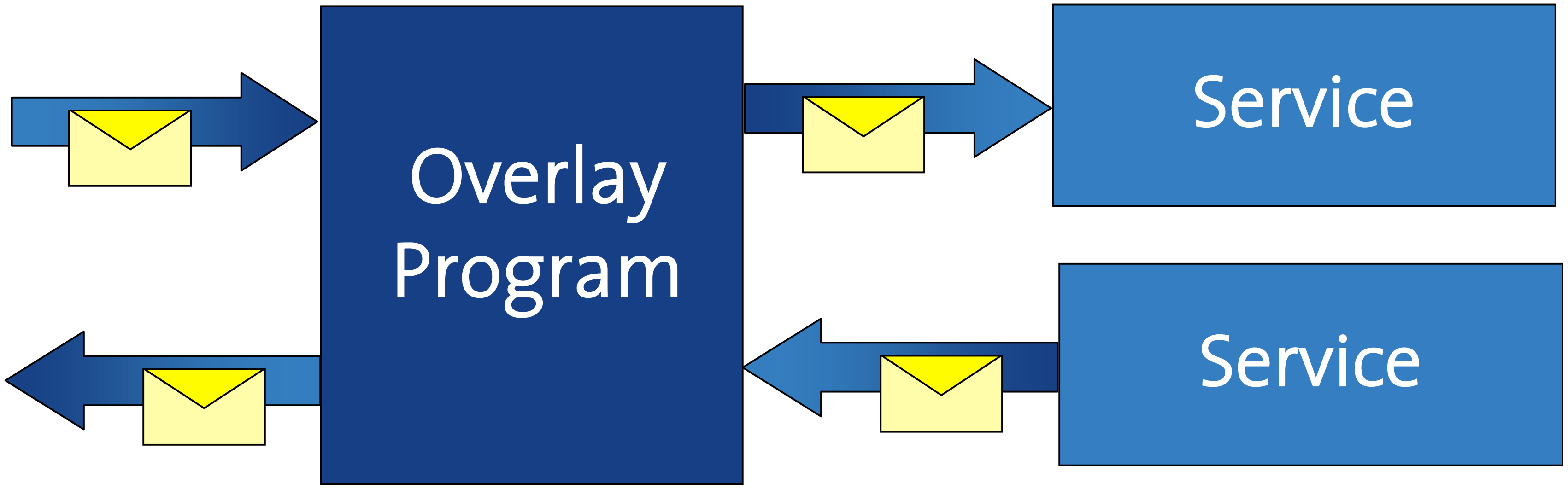
Challenge: How to Program an Overlay Network?



- How to program a distributed computation by reusing services provided by overlay nodes?
- How to orchestrate a set of services into an “overlay program”?

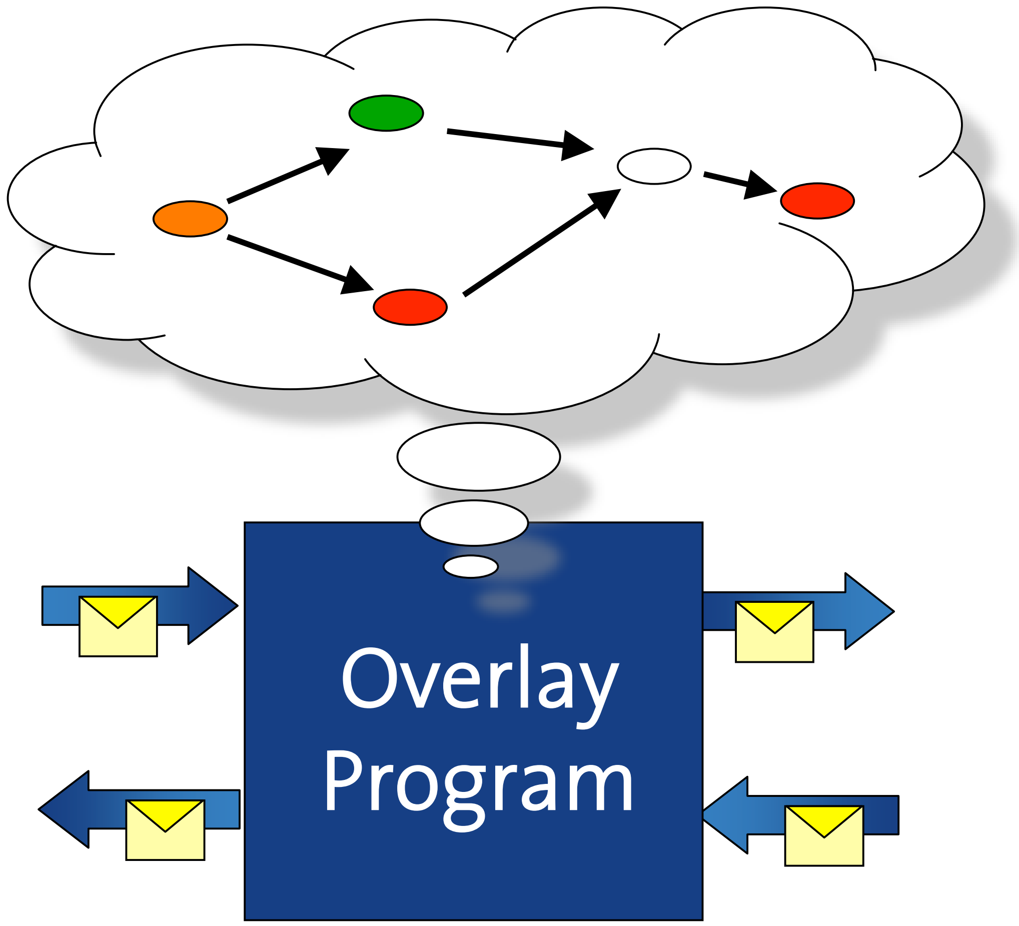
Orchestration on the Overlay Computer: What's New?

- The overlay is fully decentralized (e.g., no central registry for service discovery)
- The overlay is very dynamic (node churn)
- The overlay environment is heterogeneous

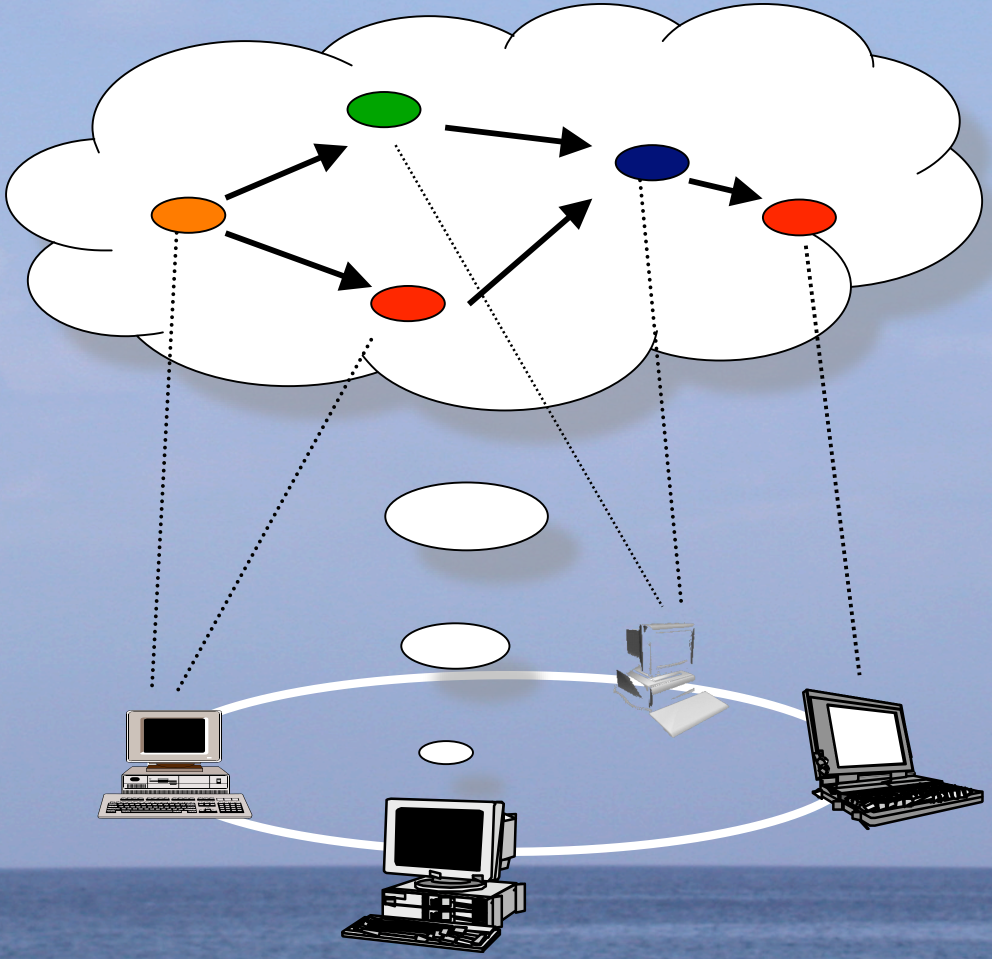


Service Orchestration on the Overlay Computer

- How to model an overlay program?
- How to execute such a model on the overlay network?
- How to deal with heterogeneous services?



How to Program the Overlay with JOpera?



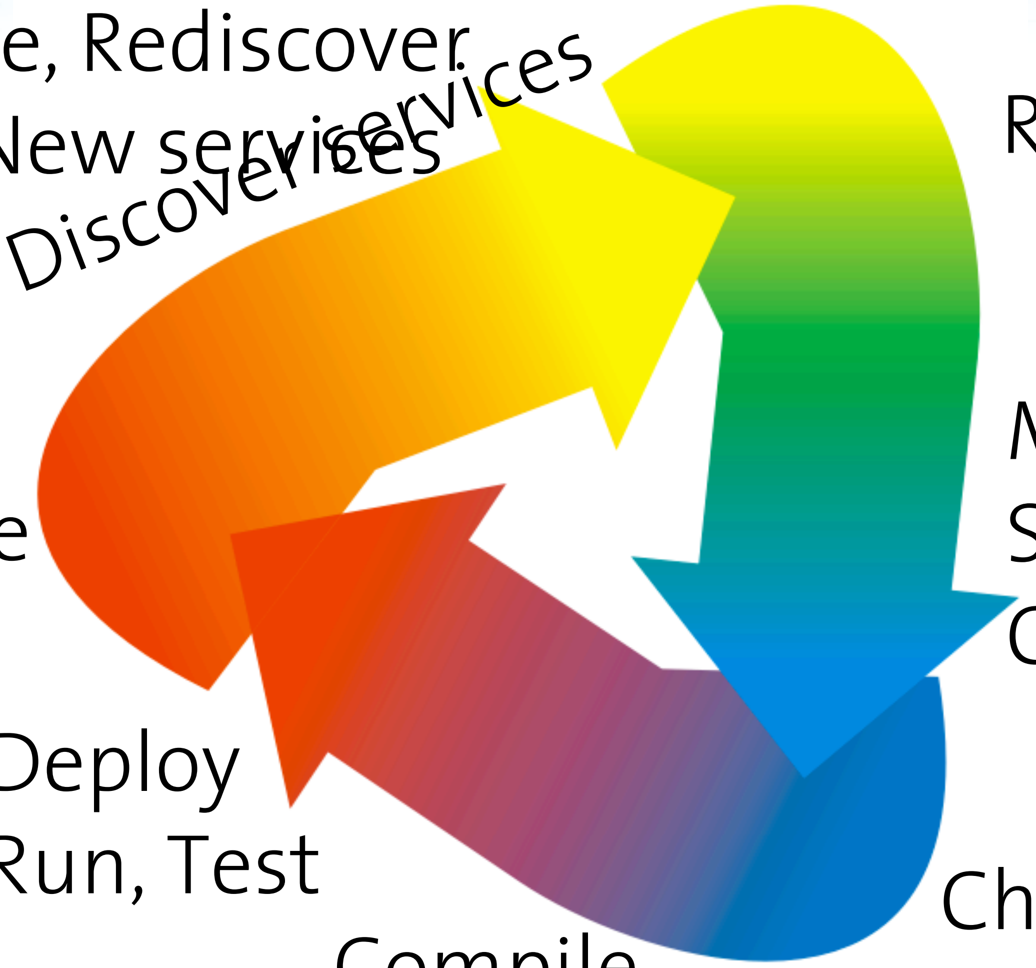
Top-down Composition

1. Define a **goal** and Draw a *skeleton of the overlay program* that satisfies it
2. Refine it and **Bind** services into it:
   - Search for existing matching services
   - Build missing services (if necessary)
   - Add required data transformations
3. Run, Test, and Debug the execution **within the same modeling environment**
4. Share and Publish it as Web Service

Bottom-up Composition

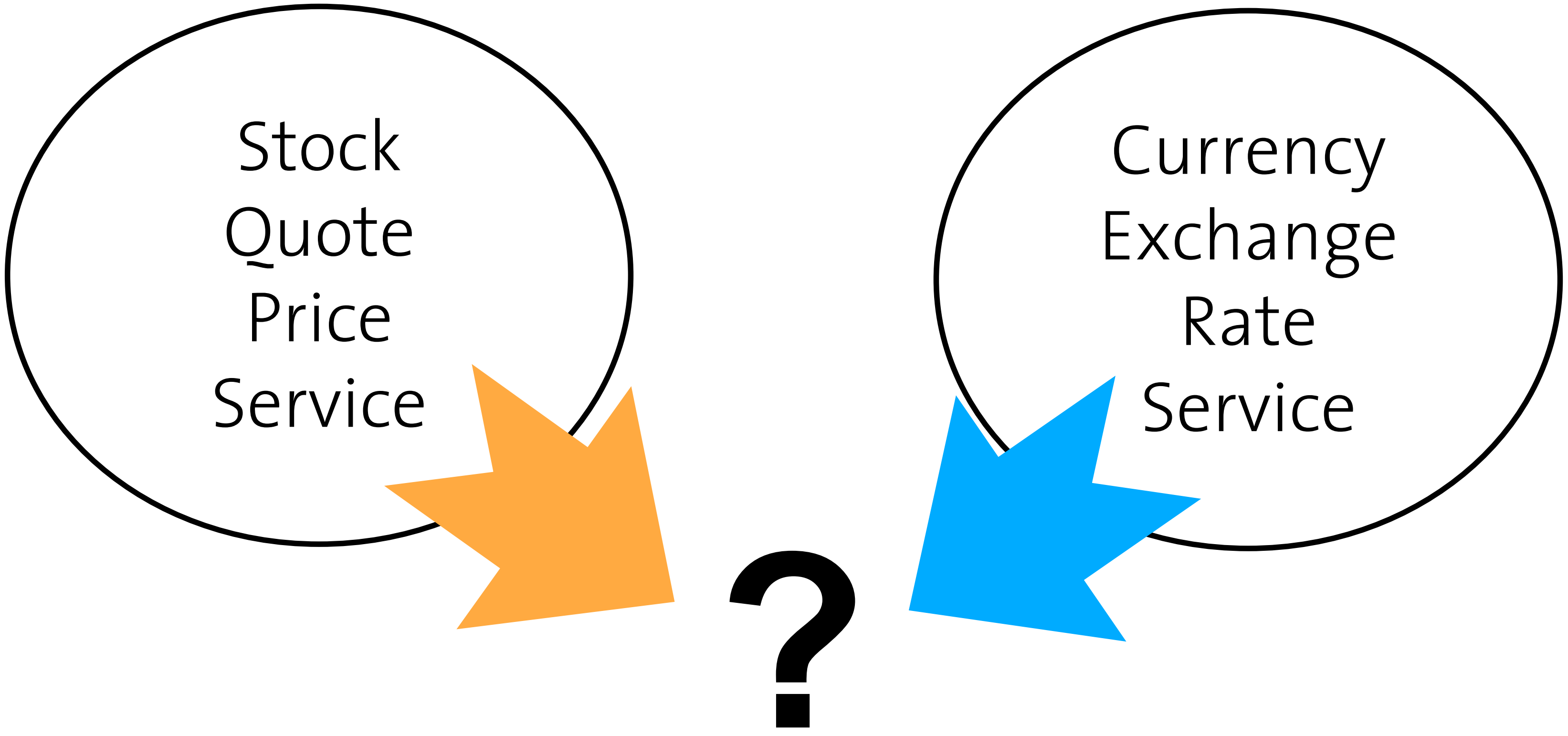
4. Share and Publish it as Web Service
3. Run, Test, and Debug the execution **within the same modeling environment**
2. Build a composition using a drag, drop and connect **modeling** environment
1. Select available services from a **library**
   - Lookup in your own library
   - Import from external WSDL
   - Search the standard JOpera library

Iterative Composition

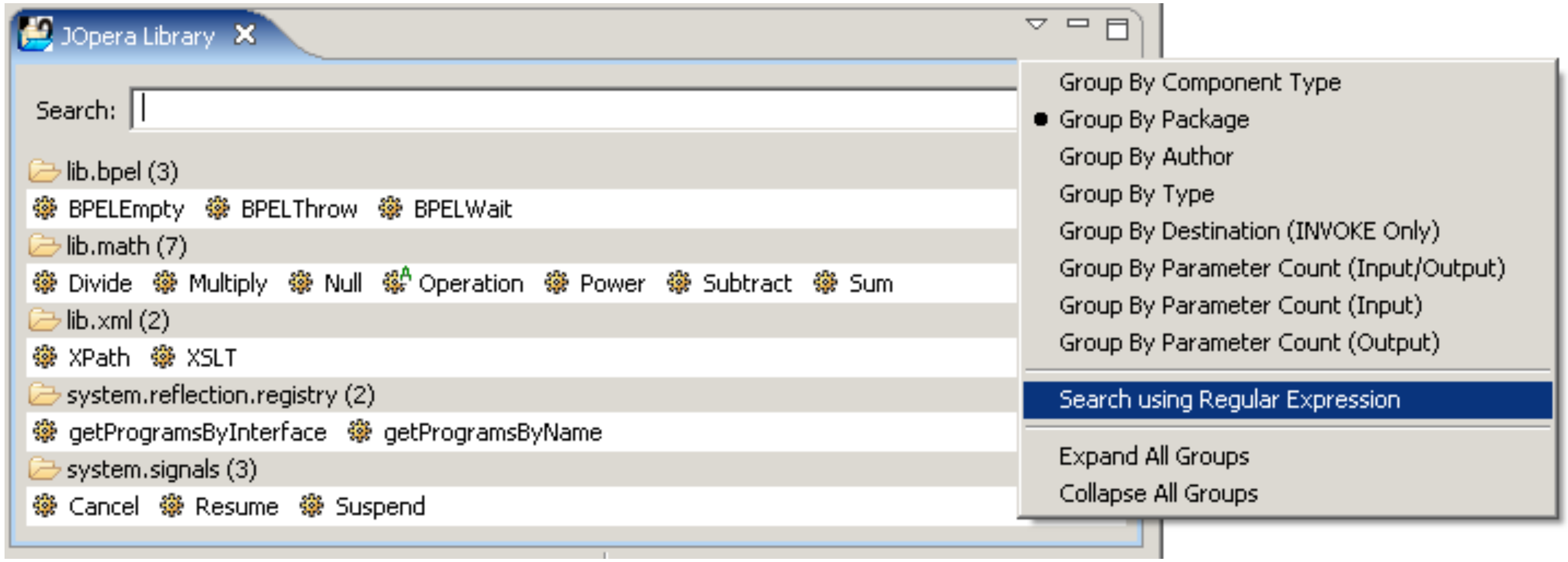


Quick Demo

- Stock Quote Currency Conversion



Service Library



JOpera Library

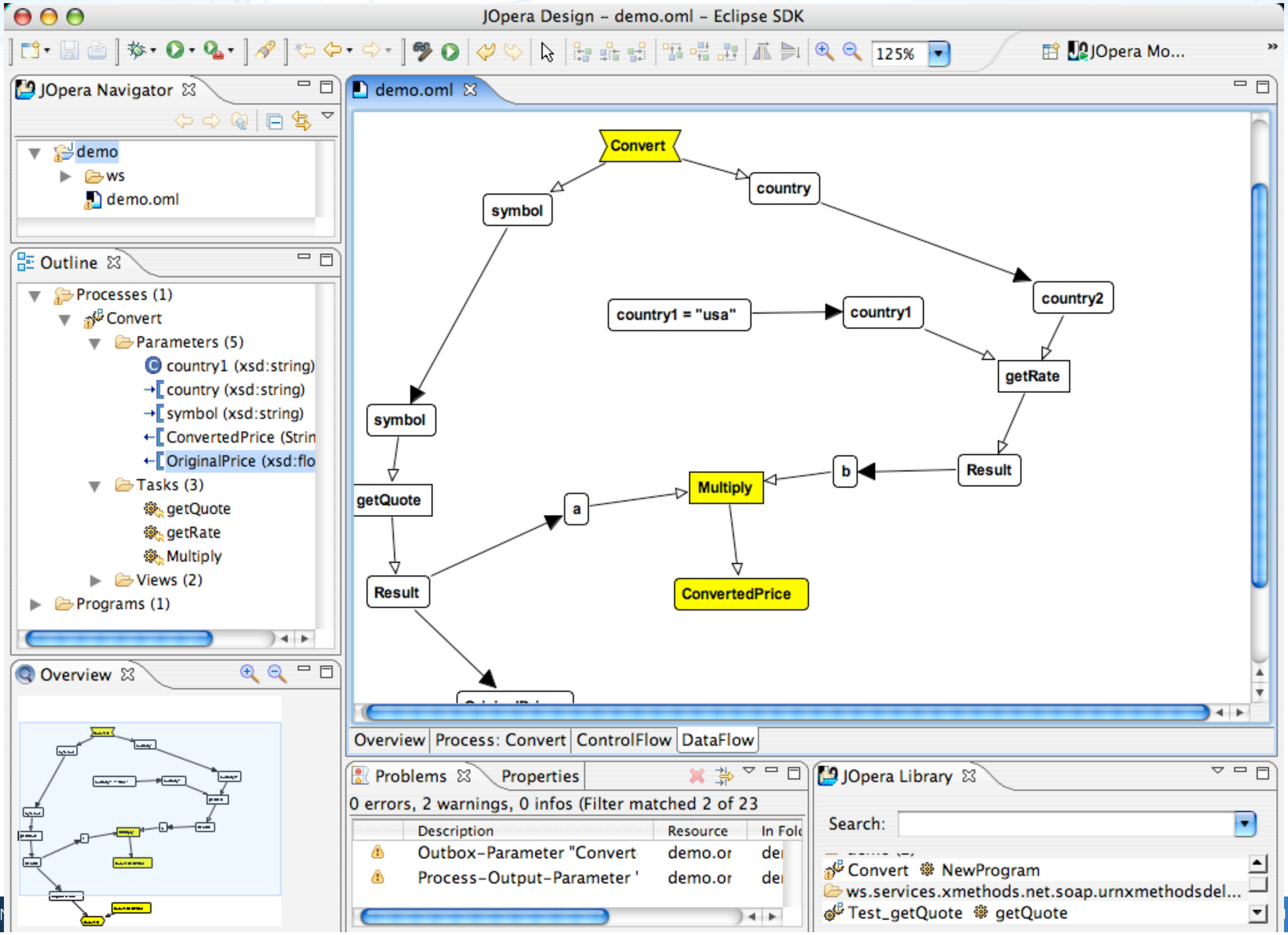
Search:

lib.bpel (3)
BPELEmpty BPELThrow BPELWait
lib.math (7)
Divide Multiply Null Operation Power Subtract Sum
lib.xml (2)
XPath XSLT
system.reflection.registry (2)
getProgramsByInterface getProgramsByName
system.signals (3)
Cancel Resume Suspend

Group By Component Type
● Group By Package
Group By Author
Group By Type
Group By Destination (INVOKE Only)
Group By Parameter Count (Input/Output)
Group By Parameter Count (Input)
Group By Parameter Count (Output)
Search using Regular Expression
Expand All Groups
Collapse All Groups

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

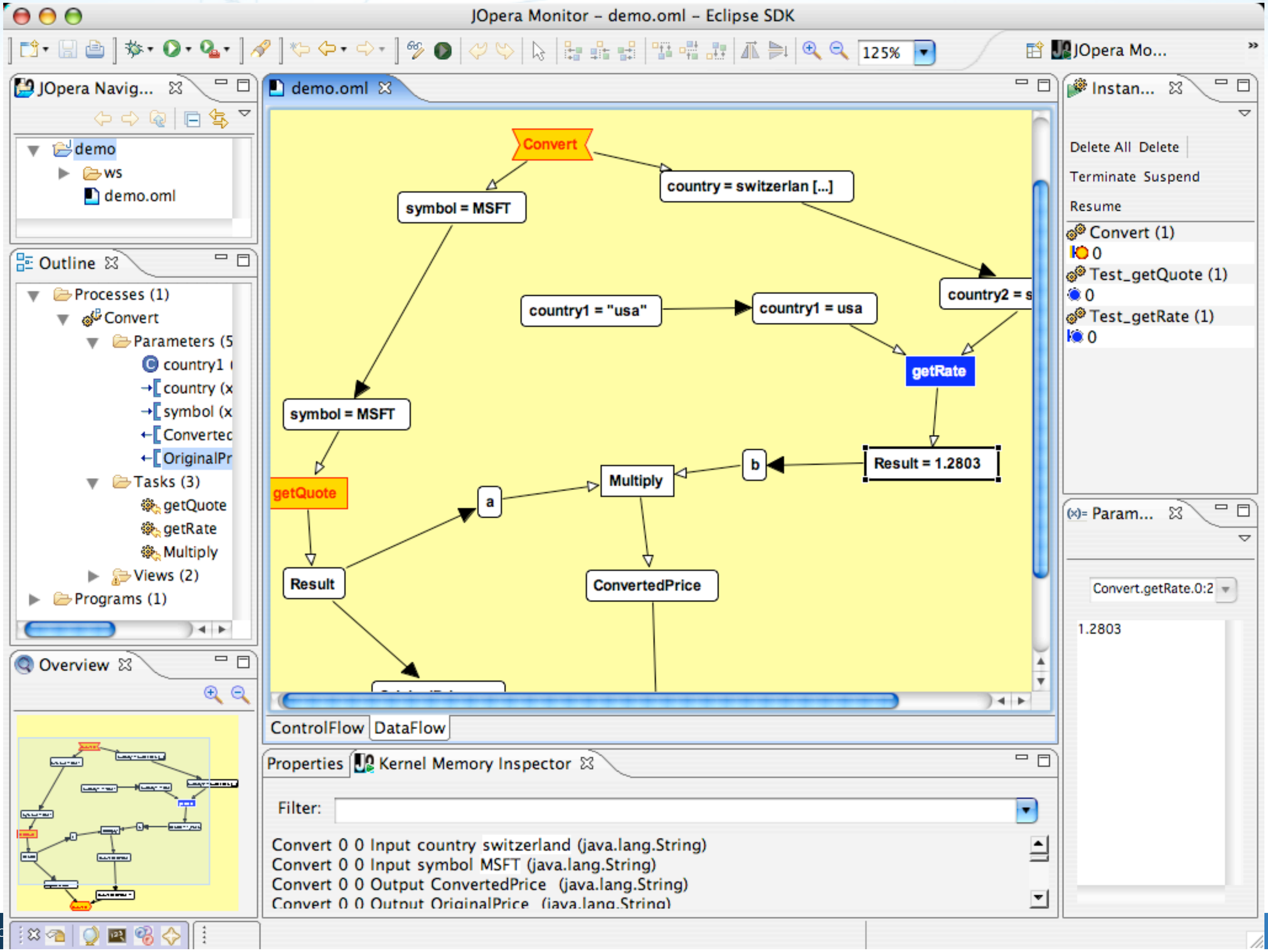
Drag, Drop and Connect



0 errors, 2 warnings, 0 infos (Filter matched 2 of 23

Description	Resource	In Fold
Outbox-Parameter "Convert	demo.or	de
Process-Output-Parameter '	demo.or	de

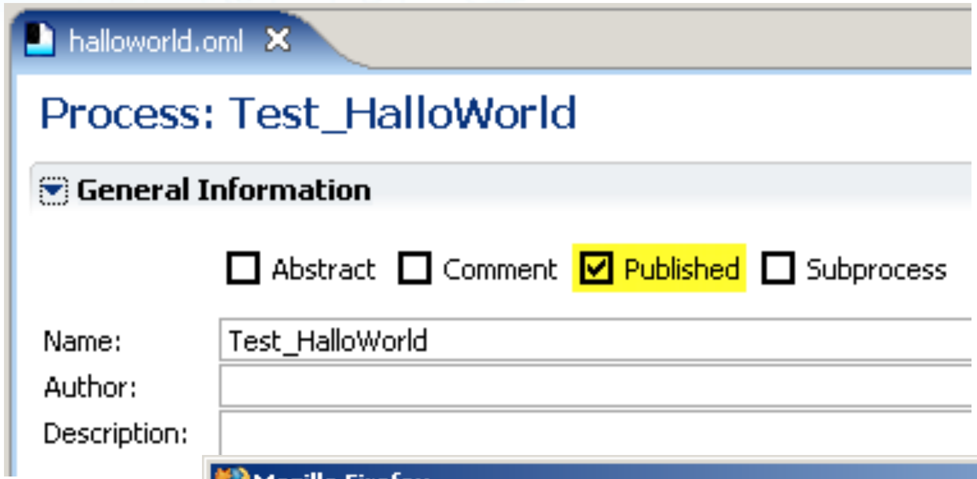
Run, Monitor, Steer and Debug



```
Convert 0 0 Input country switzerland (java.lang.String)
Convert 0 0 Input symbol MSFT (java.lang.String)
Convert 0 0 Output ConvertedPrice  (java.lang.String)
Convert 0 0 Output OriginalPrice  (java.lang.String)
```

Publish as a Web/Grid service

With one mouse click!

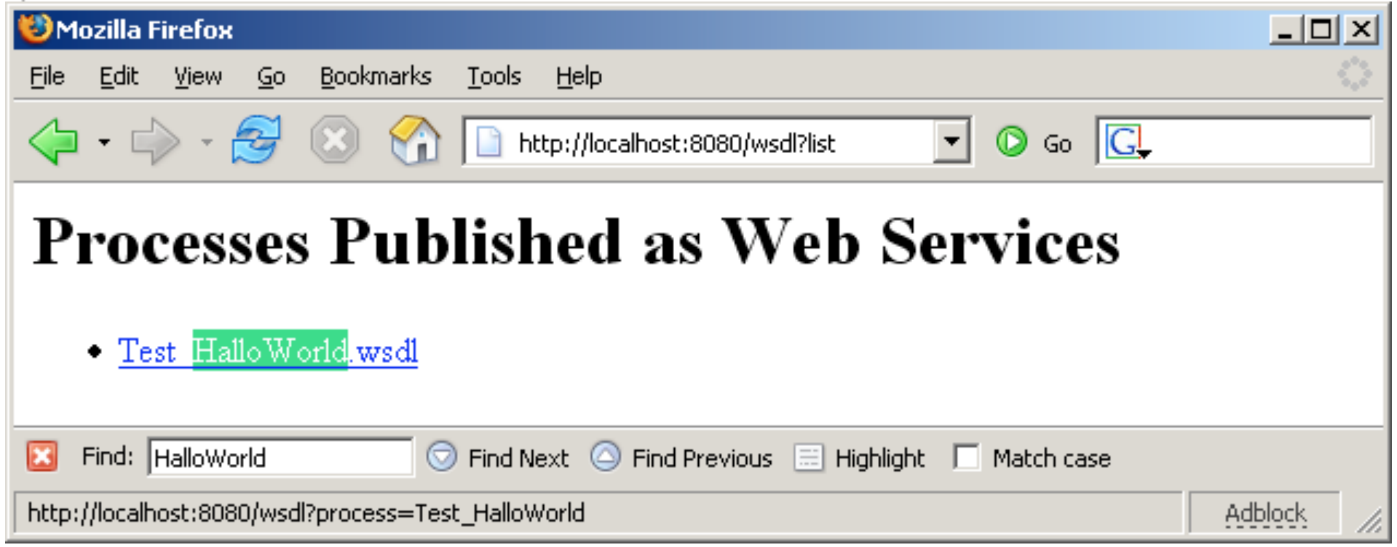

halloworld.oml

Process: Test_HalloWorld

General Information

☐ Abstract ☐ Comment ☑ Published ☐ Subprocess

Name:	Test_HalloWorld
Author:	
Description:	



http://localhost:8080/wsdl?list

Processes Published as Web Services

- Test_HalloWorld.wsdl

Find: HalloWorld ☐ Match case

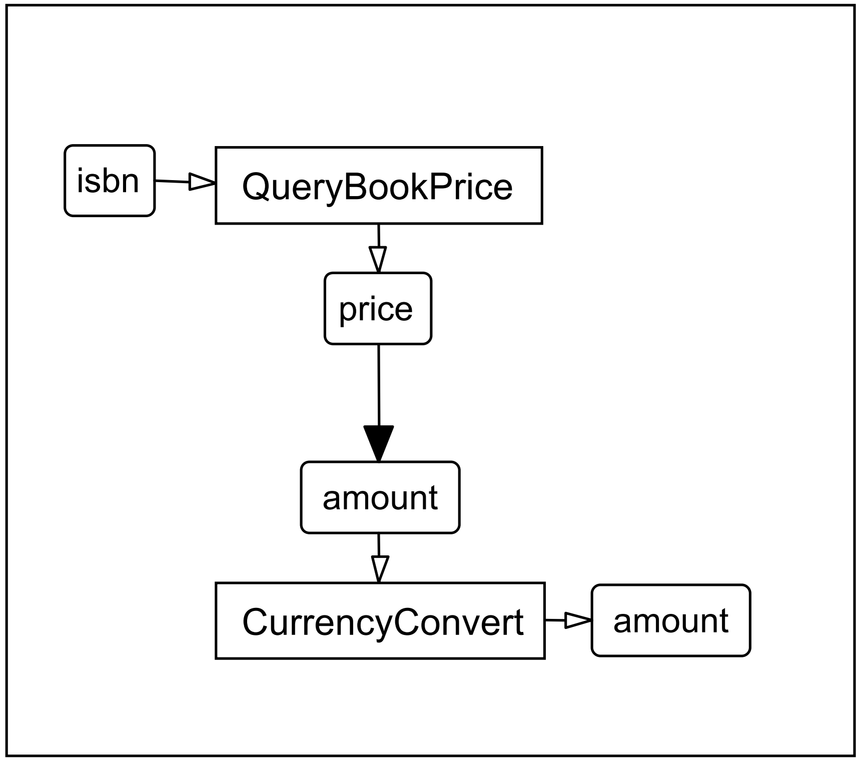
http://localhost:8080/wsdl?process=Test_HalloWorld

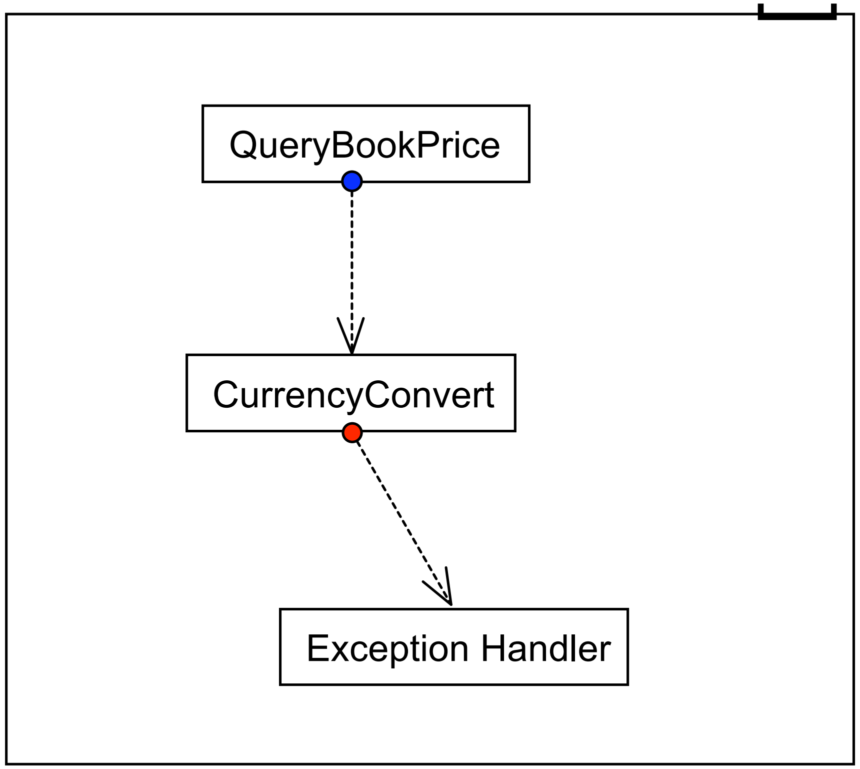
[e-Science2005]

JOpera Visual Composition Language Overview

[HCC2003]

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





JOpera Visual Composition Language Features

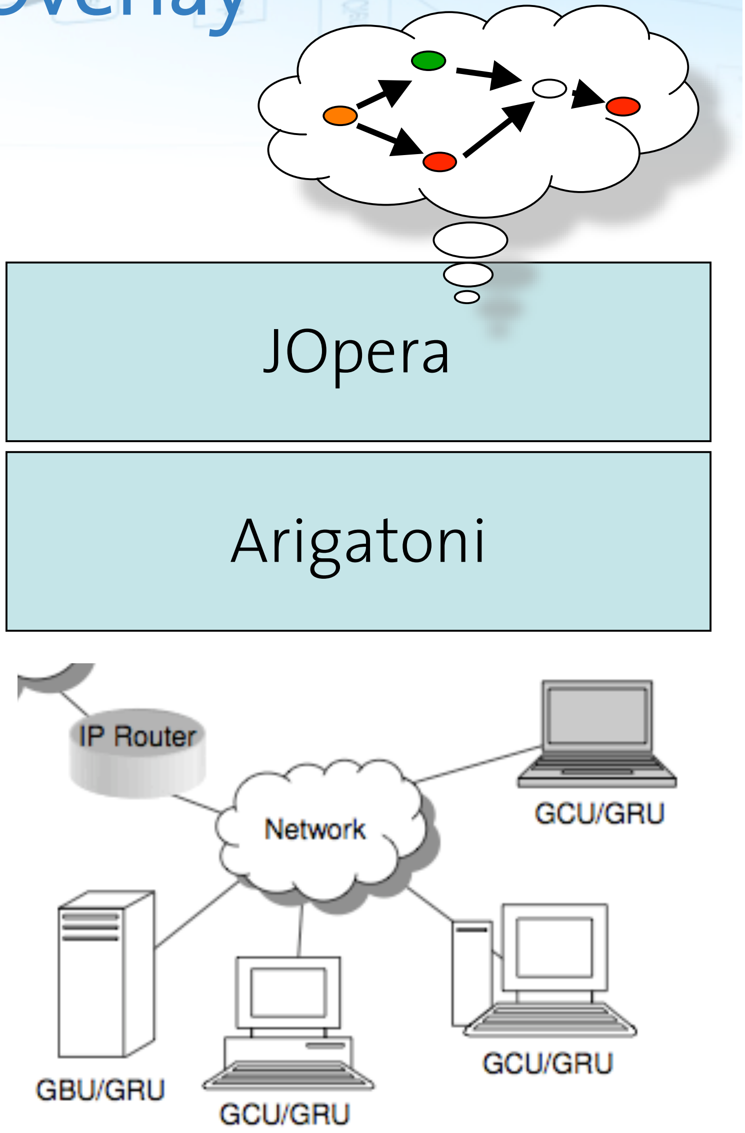
- Processes model overlay programs
  - **Data flow** as the primary representation
  - Explicit **control flow** (branch, synchronization, exception handling, arbitrary loops, *pipelines*, workflow patterns)
- **SubProcesses**: Modularity, Nesting and Recursion
- First order functions
  - **Map** (parallel/sequential/discriminator)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

[JVLC2005]

Running Overlay Programs with JOpera for Eclipse

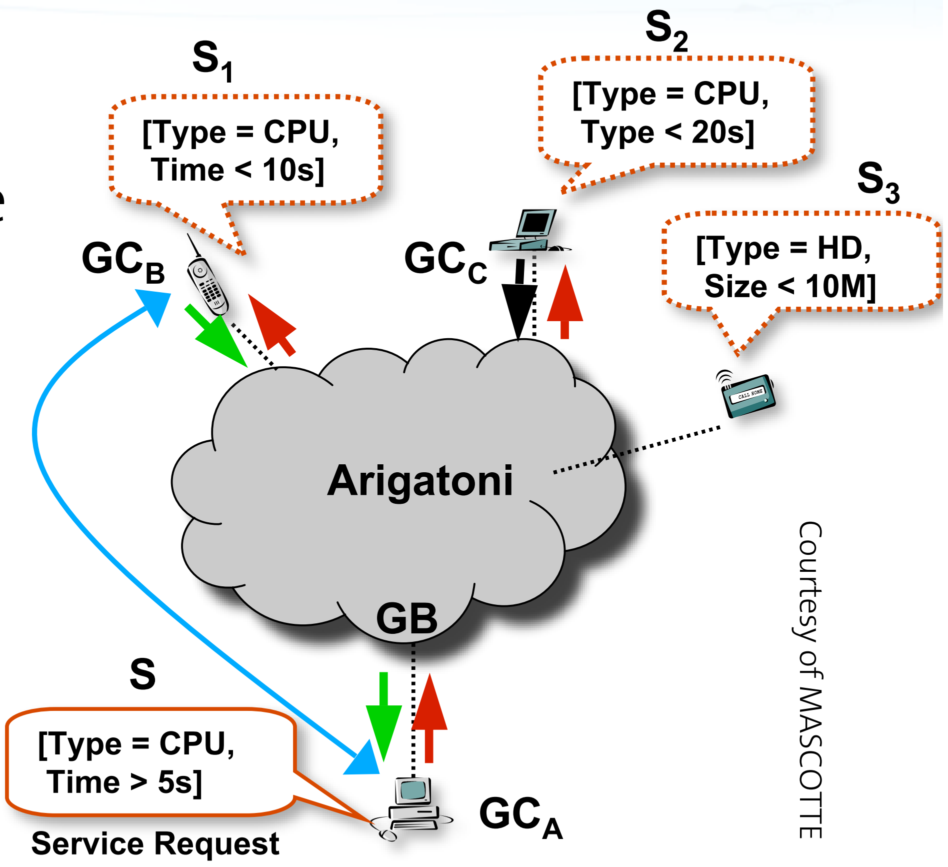
Running Workflows on the Overlay

- **Discovery**
  - Map overlay program to actually available services provided by the overlay nodes
- **Orchestration**
  - Execute the “workflow” by interpreting the overlay program
  - Control how the nodes exchange data and use each other’s computational services
- **Adaptation**
  - Survive churn on the overlay



About the Arigatoni Overlay Network

- Resource Discovery Protocol (RDP)
  - Once resources and service are discovered, orchestration can begin
- Virtual Intermittence Protocol (VIP)
  - Deal with churn, by updating routing tables
- **Arigatoni Orchestration Language (AOL) by JOpera**



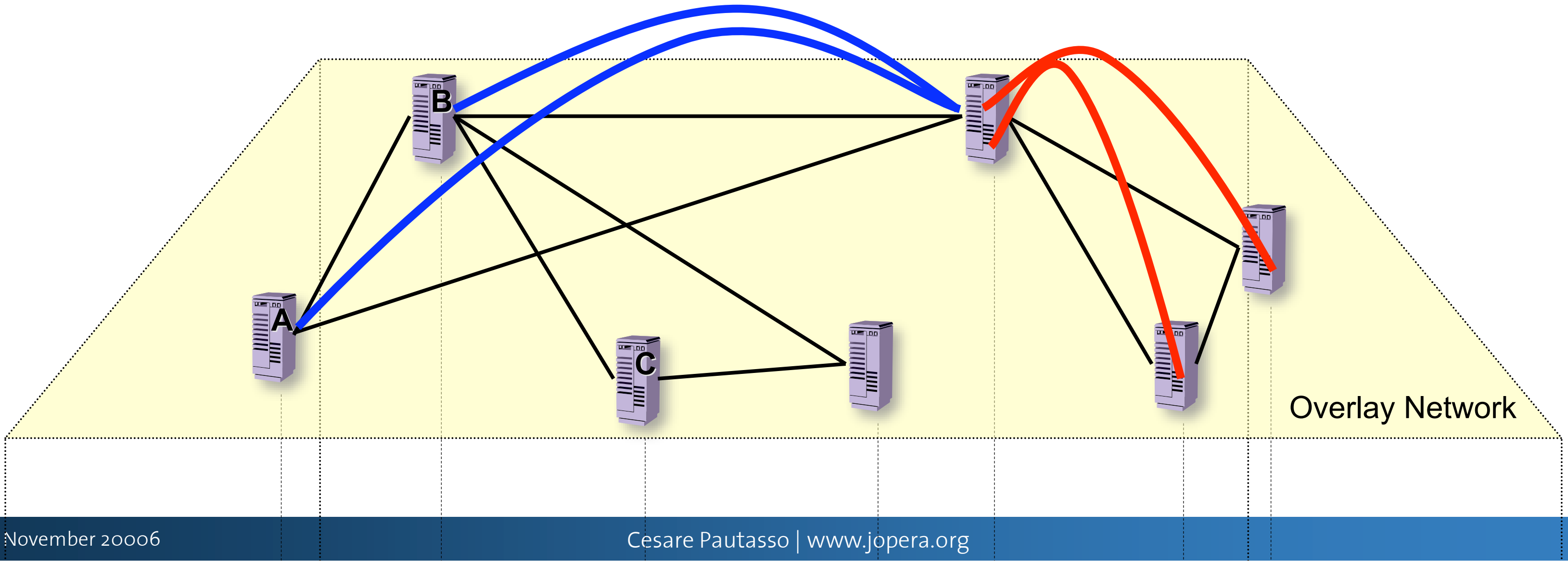
Courtesy of MASCOTTE

Discovery before or during Orchestration?

- Late, Synchronous, Local
  - When each workflow task is about to start, run the discovery protocol to locate the most suitable node
- Early, Synchronous, Global
  - Before each workflow execution, run the discovery protocol over all tasks of the workflow
  - The query takes into account the workflow structure
  - Re-run the discovery protocol if tasks fail because of churn
- Asynchronous
  - Periodically discover nodes and update workflow binding information accordingly

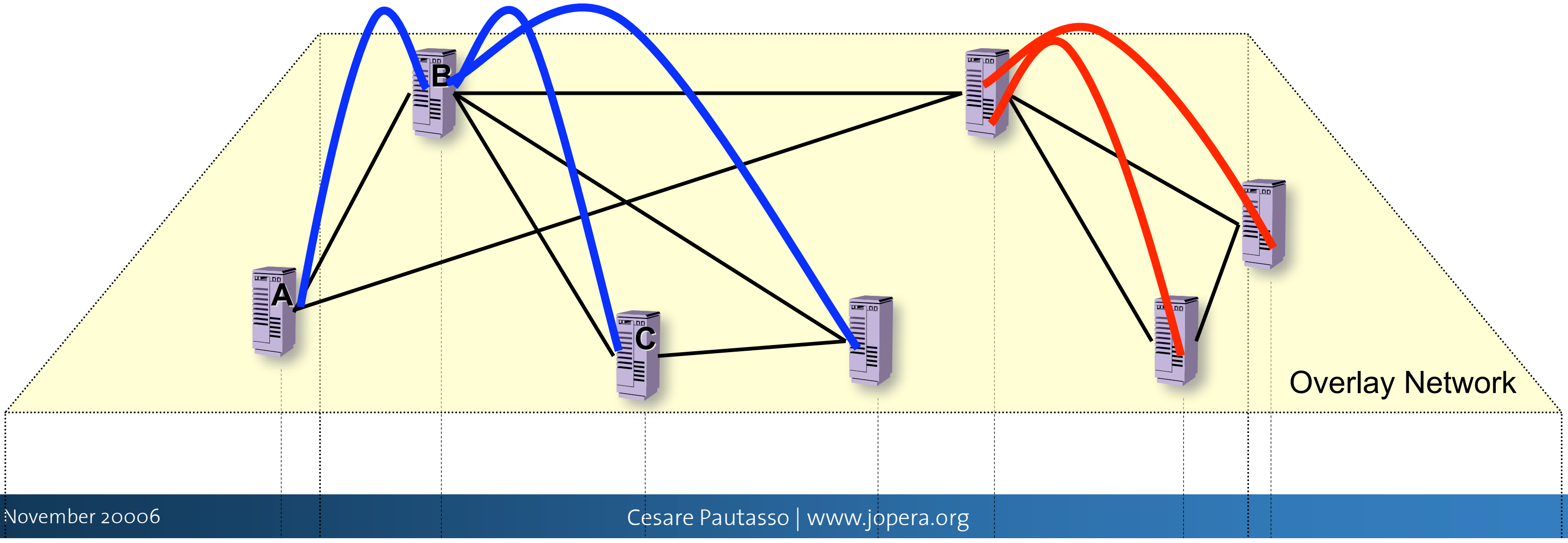
Orchestration and the Overlay (1)

- Global Centralized Workflow Engine
  - All workflows are executed on the same overlay node (the "CPU" of the overlay computer)
  - All discovery queries go through the same node
  - All workflow execution messages go thro the same node



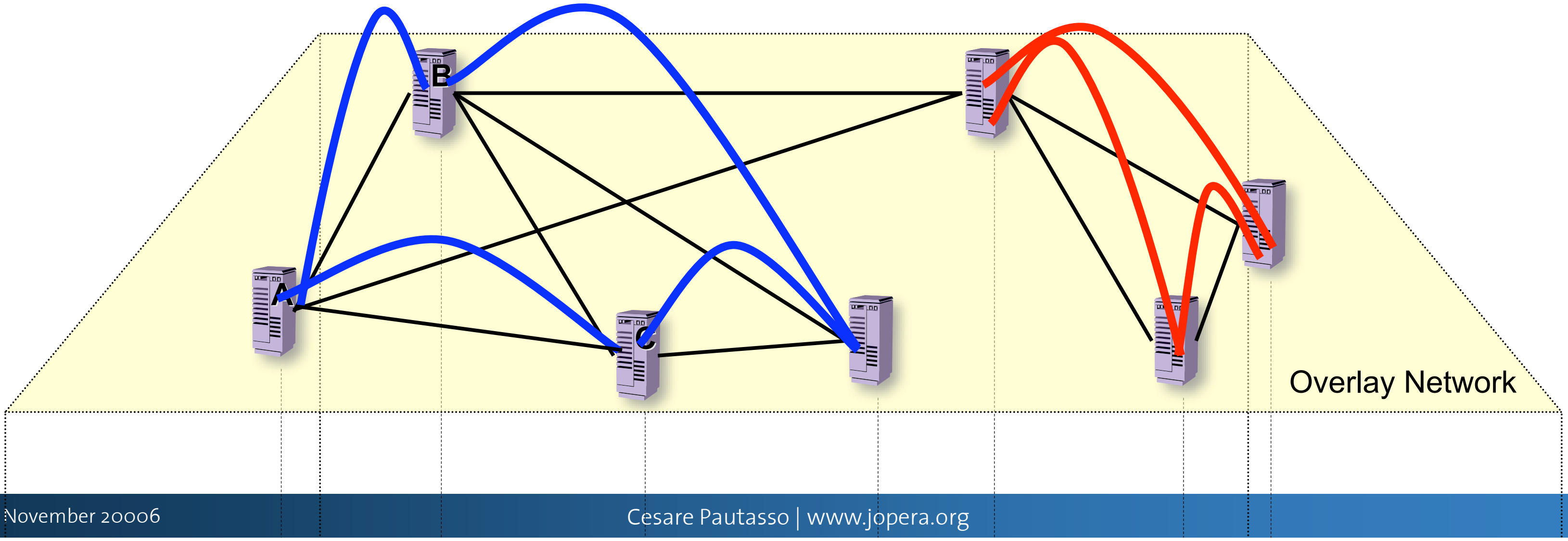
Orchestration and the Overlay (2)

- **Local** Centralized Workflow Engine
  - Each workflow runs on a different node, which orchestrates the services of a collection of nodes (the “Tracker” for the workflow instance)
  - Workflow execution node can be discovered
  - Centralized Data flow

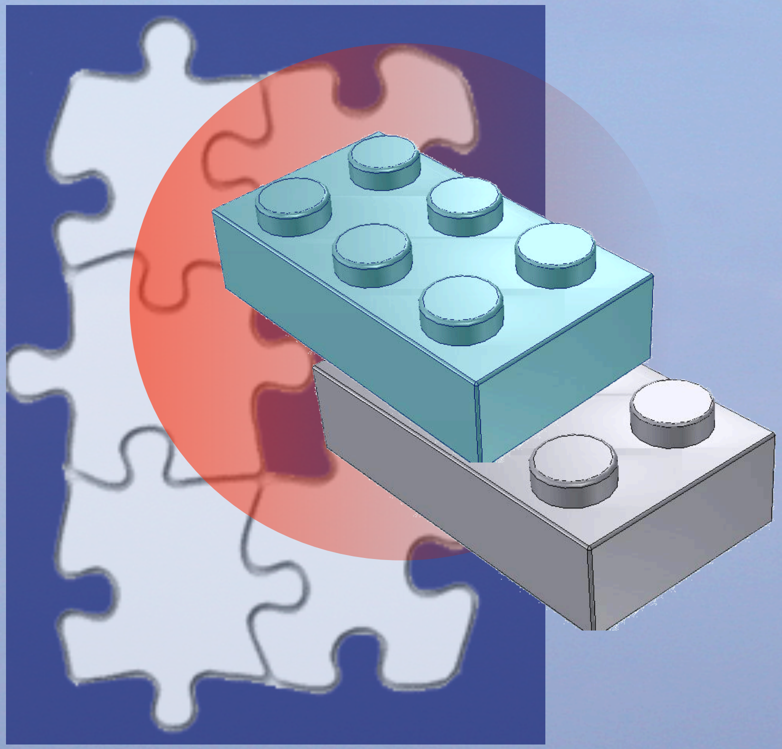


Orchestration and the Overlay (3)

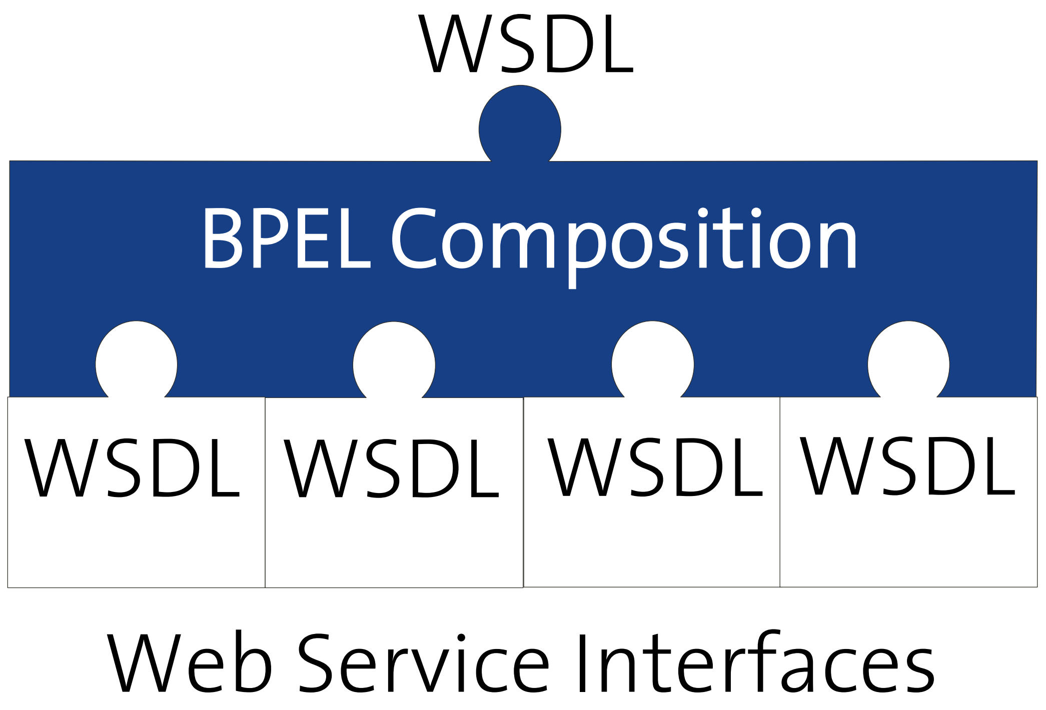
- Partitioned Engine
  - The nodes providing the services also take care of orchestrating themselves according to the workflow
  - Nodes directly connect to each other (discovery more complex)
  - Workflow needs to be partitioned, and each partition sent to nodes



What kinds of Services can you compose with JOpera?



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

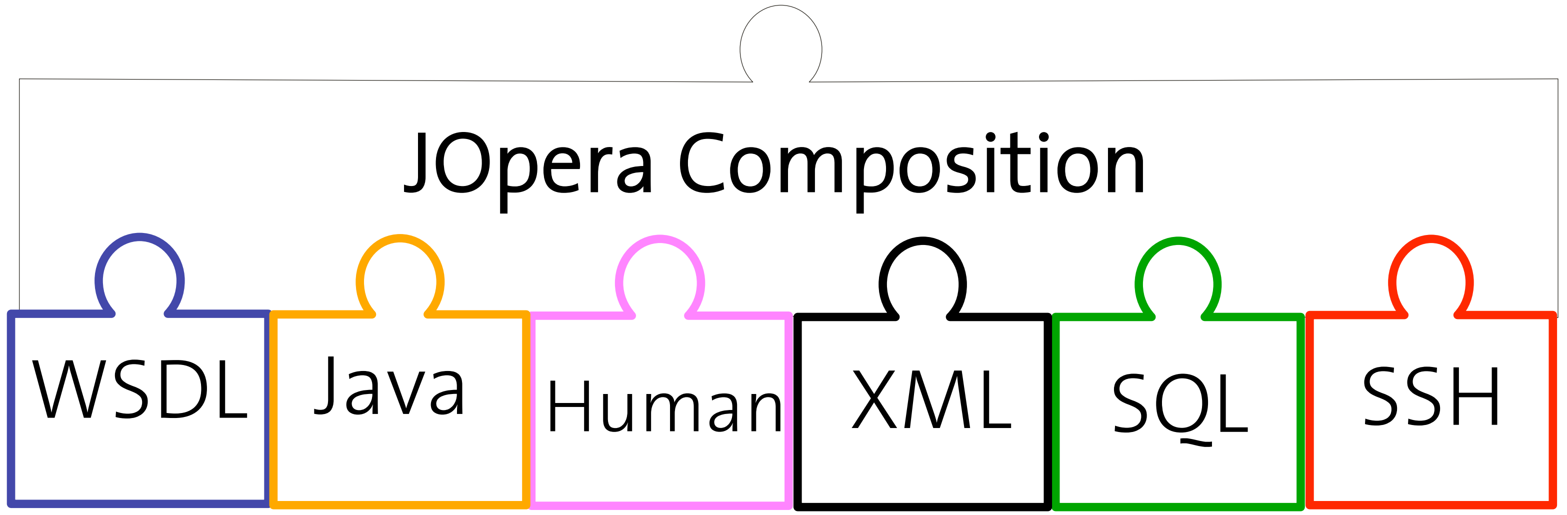


Assumption:
Web Services (SOAP/WSDL) are the only kind of services to be composed

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

Dealing with heterogeneity in JOpera

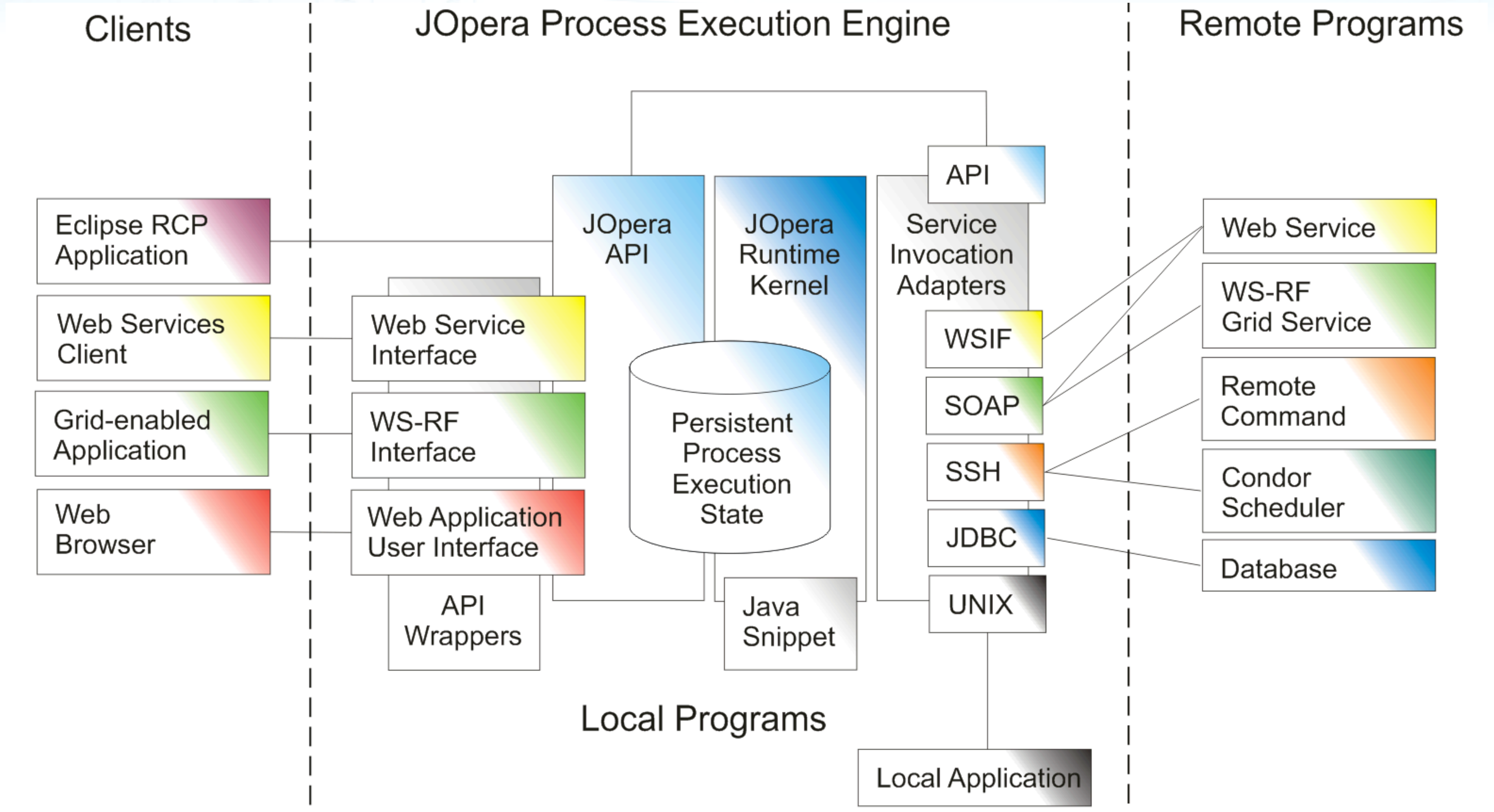
- The JOpera composition language does not have to be changed when adding a new kind of service

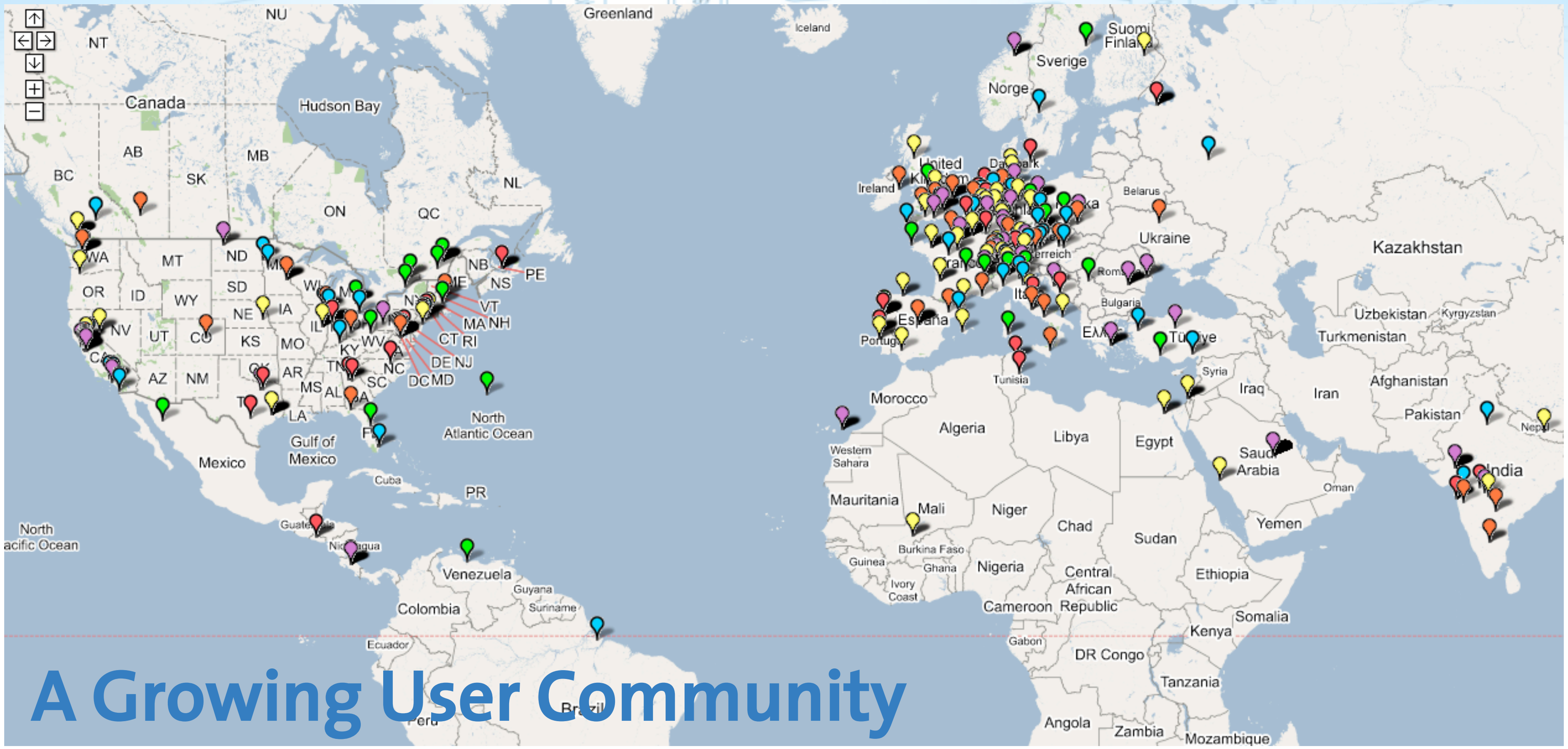


- Snippets
- Methods

- XSLT
- XPath

Architecture of JOpera for Eclipse



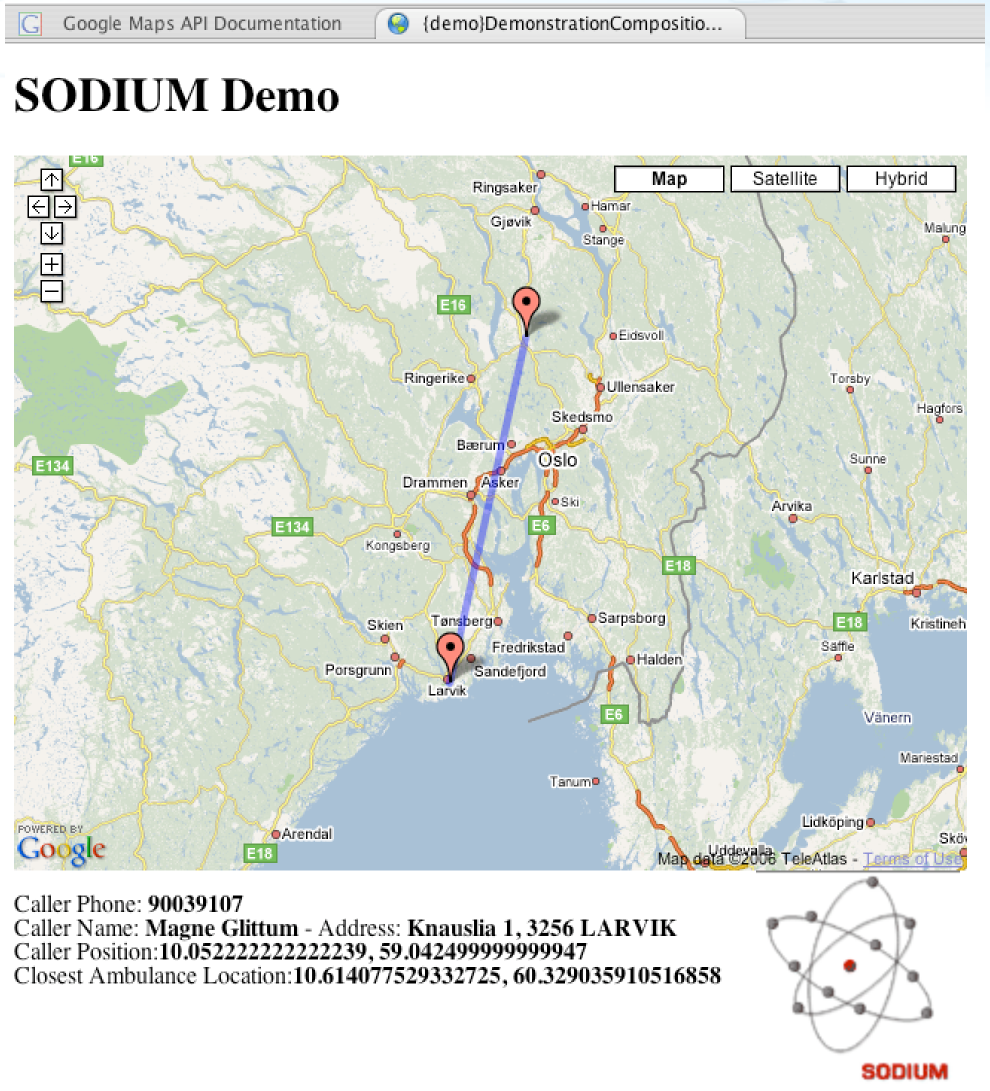


A Growing User Community

ETH Zurich, Swiss Bioinformatics Institute, Swiss National Supercomputing Center, European Synchrotron Radiation Facility, Purdue University, McGill University (Montreal), Singapore Mgmt University, National University of Defence Technology (China), Arjuna (UK), SINTEF (No), Locus (No), NCSA, …

LOCUS, Norway

- SODIUM EU Project
- Service Oriented Development in a Unified Framework
- Pilot application in GIS, e-Health and emergency rescue services



SODIUM Demo

Caller Phone: **90039107**
Caller Name: **Magne Glittum** - Address: **Knauslia 1, 3256 LARVIK**
Caller Position:**10.052222222222239, 59.042499999999947**
Closest Ambulance Location:**10.614077529332725, 60.329035910516858**

Climate Modeling on TERAGRID

- Continuous processing of satellite feeds for climate modeling and weather forecasting
- JOpera a key part of the infrastructure to glue together the data and analysis services into Grid workflows

Cyberinfrastructure for e-Science at the National Center for Supercomputing Applications

- Grid Workflows important part of the Service Oriented Grid middleware stack
- JOpera Pilot Application: porting the data flow based "Data 2 Knowledge" toolkit to Eclipse

Why users like JOpera

- High Level Workflow Language
  - Data and Control Aspects (Graphical Representation)
  - Recursion, Iteration, Parallelism and Pipelining Constructs
- Open and Extensible Component Model
  - Run existing code without changes
  - Synchronous, Asynchronous, Streaming interaction
  - Web services support (Axis, WSIF)
  - Secure access to remote file systems and hosts (SSH, SCP)
  - Easy to integrate with existing schedulers (Condor already supported)

Why users like JOpera

- High Level Workflow Language
  - Data and Control Aspects (Graphical Representation)
  - Recursion, Iteration, Parallelism and Pipelining Constructs
- Open and Extensible Component Model
  - Run existing code without changes
  - Synchronous, Asynchronous, Streaming interaction
  - Web services support (Axis, WSIF)
  - Secure access to remote file systems and hosts (SSH, SCP)
  - Easy to integrate with existing schedulers (Condor already supported)
- Strong Eclipse Foundation
  - Platform Independent (Eclipse/Java)
  - Flexible, Extensible, Modular and Embeddable

Conclusion

- **Modeling** overlay programs
  - Flow-based **composition language** (Visual & XML)
  - Development and Debugging tools for Eclipse
  - Invocation of heterogeneous services
- **Execution** of the overlay programs
  - Efficiency (compiled to Java bytecode)
  - Distributed engine (on the overlay)
  - Autonomic platform (self-healing, self-tuning)
  - Integration with Arigatoni for dynamic service discovery and their orchestration

JOpera Team

Cesare Pautasso Thomas Heinis Bioern Bioernstad

Andreas Bur Fabian Pichler Patrick Jayet

Adrian Listyo Sandra Brockmann Christoph Schwank

Dennis Rietmann Dominique Schneider Markus Egli

Michael Lorenzi Christian Rupp Markus Haller Axel Wathne

Antonio Caliano Oliver Deak Reto Schaeppi

Nicholas Born Philip Frey Patrick Moor

Special Thanks

Claus Hagen Win Bausch Gustavo Alonso Michael Hallett

References on the language

[ICWS2006] Bioern Bioernstad, Cesare Pautasso, Gustavo Alonso, **Control the Flow: How to Safely Compose Streaming Services into Business Processes**, In: the 2006 IEEE International Conference on Services Computing (SCC 2006), Chicago, September 2006

[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

inf Informatik
Computer Scienc

IKS Information and Communication Systems Research Group

Cesare Pautasso Thomas Heinis Bioern Bioernstad Andreas Bur Fabian Pichler Patrick Moor Adrian Listyo Patrick Jayet Sandra Brockmann Christoph Schwank
Dennis Rietmann Dominique Schneider Markus Egli Michael Lorenzi Christian Rupp Markus Haller Axel Wathne Antonio Caliano Oliver Deak
Reto Schaeppi Nicholas Born Philip Frey Claus Hagen Win Bausch Gustavo Alonso

Thank you for your feedback:

www.jopera.org

Free Download