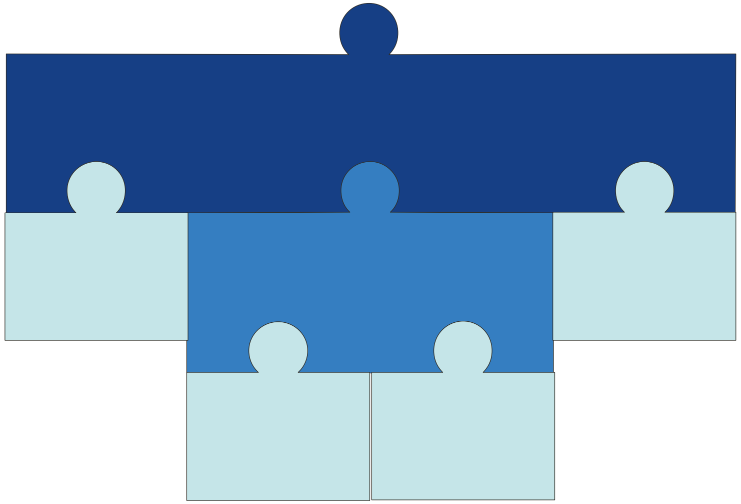
JOpera: an Autonomic Platform for Service Composition

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

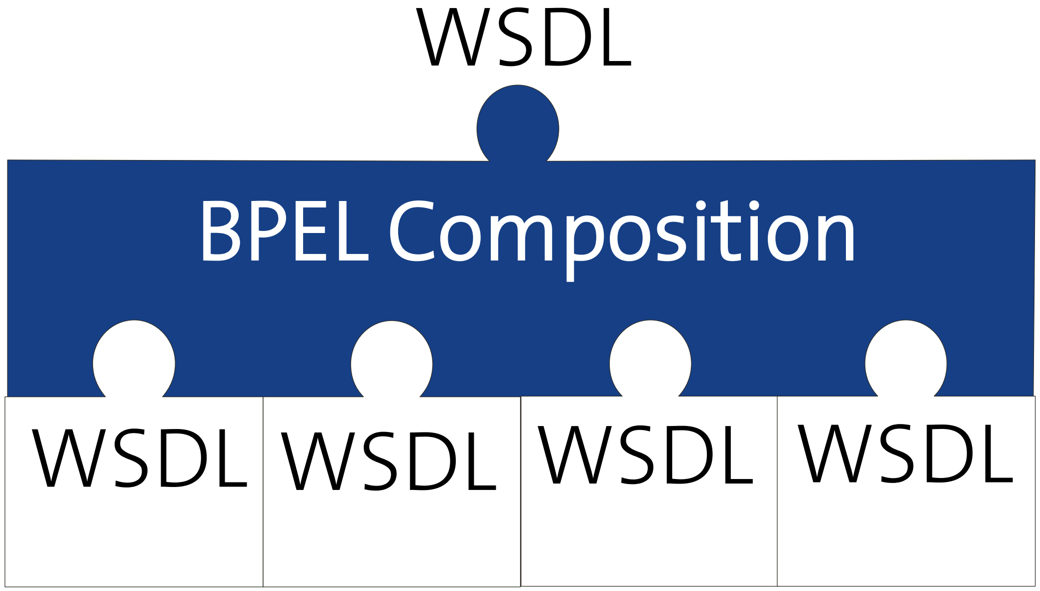
Service Composition is Recursive

The result of a service composition is a service



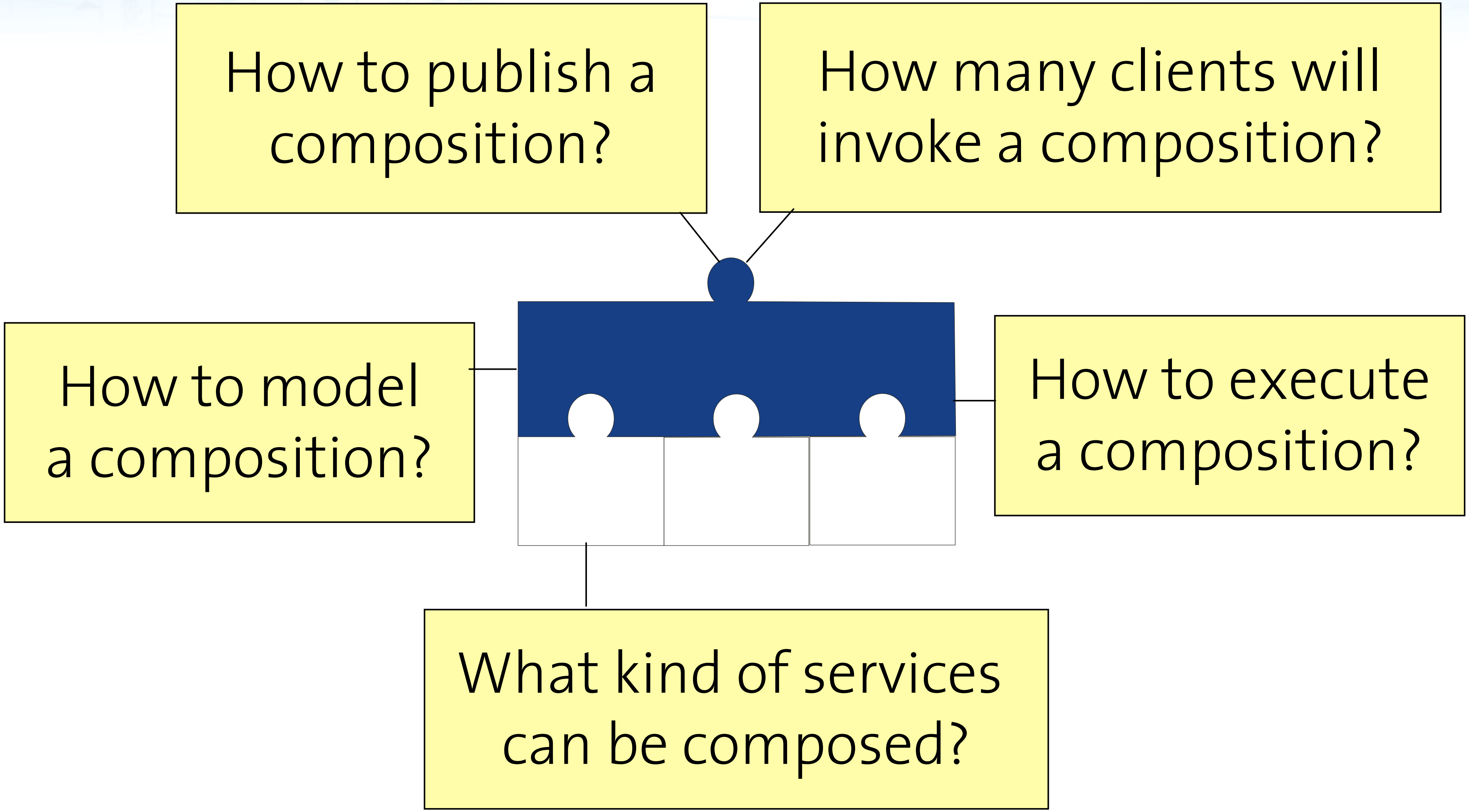
What are the implications?

Example: WS-BPEL and Service Oriented Architectures



Web Service Interfaces

Recursive Service Composition Problems



How to publish a composition?

How many clients will invoke a composition?

How to model a composition?

How to execute a composition?

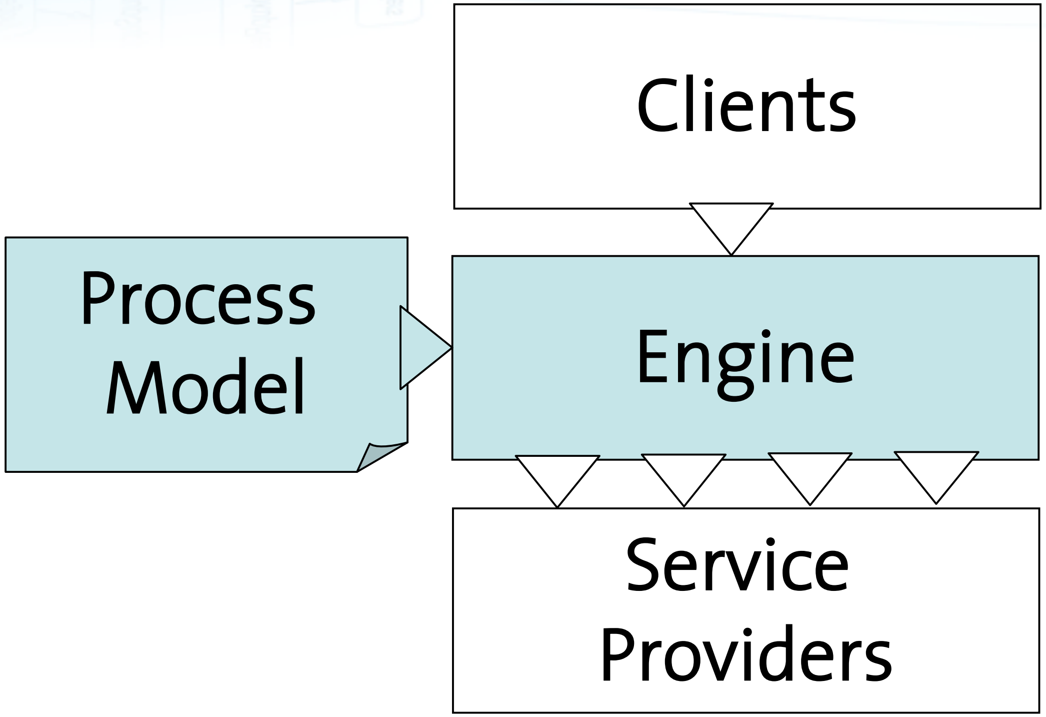
What kind of services can be composed?

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, JVLC2005]

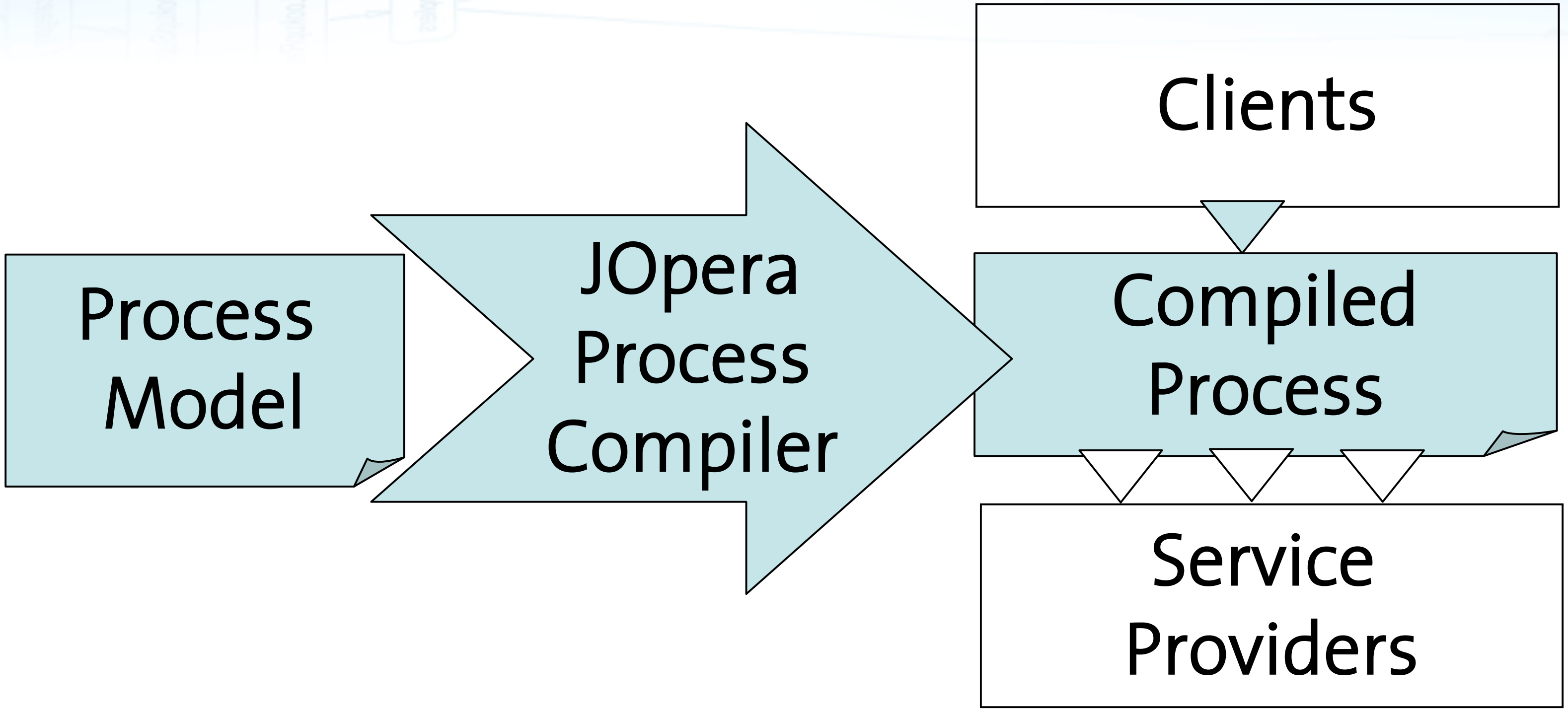
Executing Processes



Interpreted Execution:
The engine interprets the process model

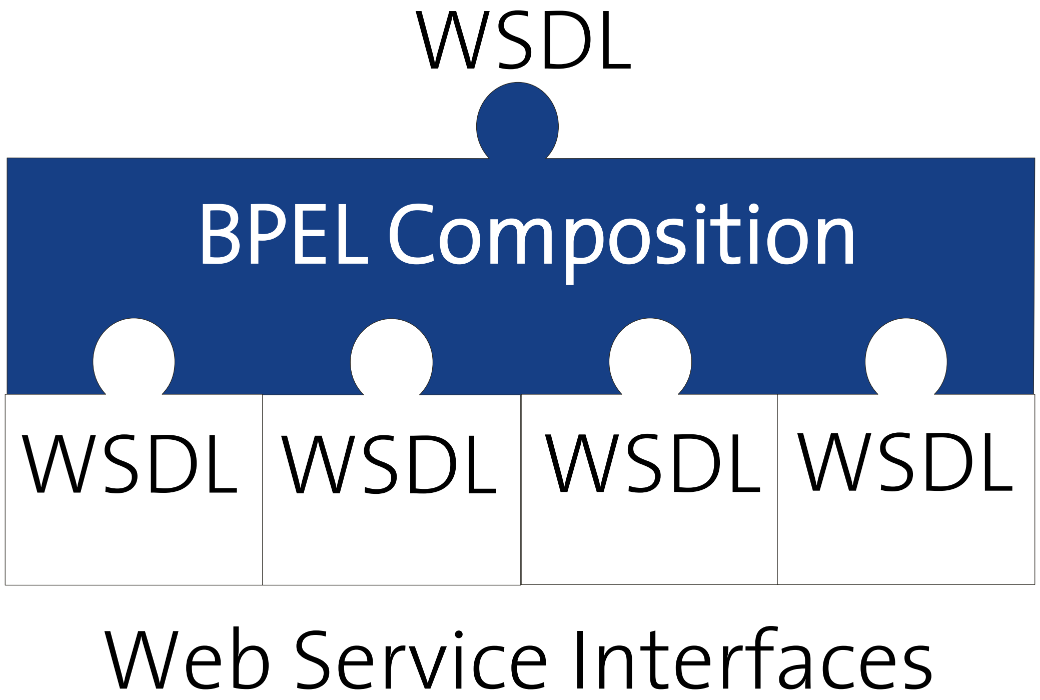
- Requirements:
  - Efficiency, Scalability, Reliability

Executing Compositions (Compiled)



- For efficient execution, in JOpera process models are compiled to Java bytecode

What kind of services can be composed with WS-BPEL?



Assumption:

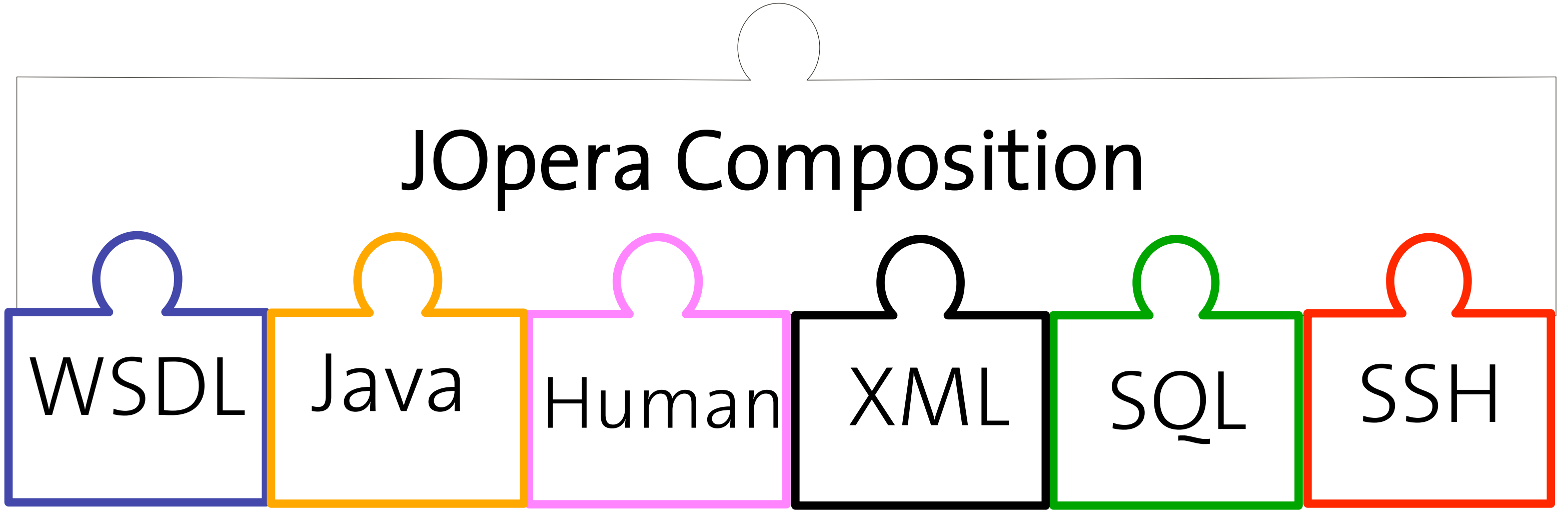
Web Services (SOAP/WSDL) are the only kind of services to be composed

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



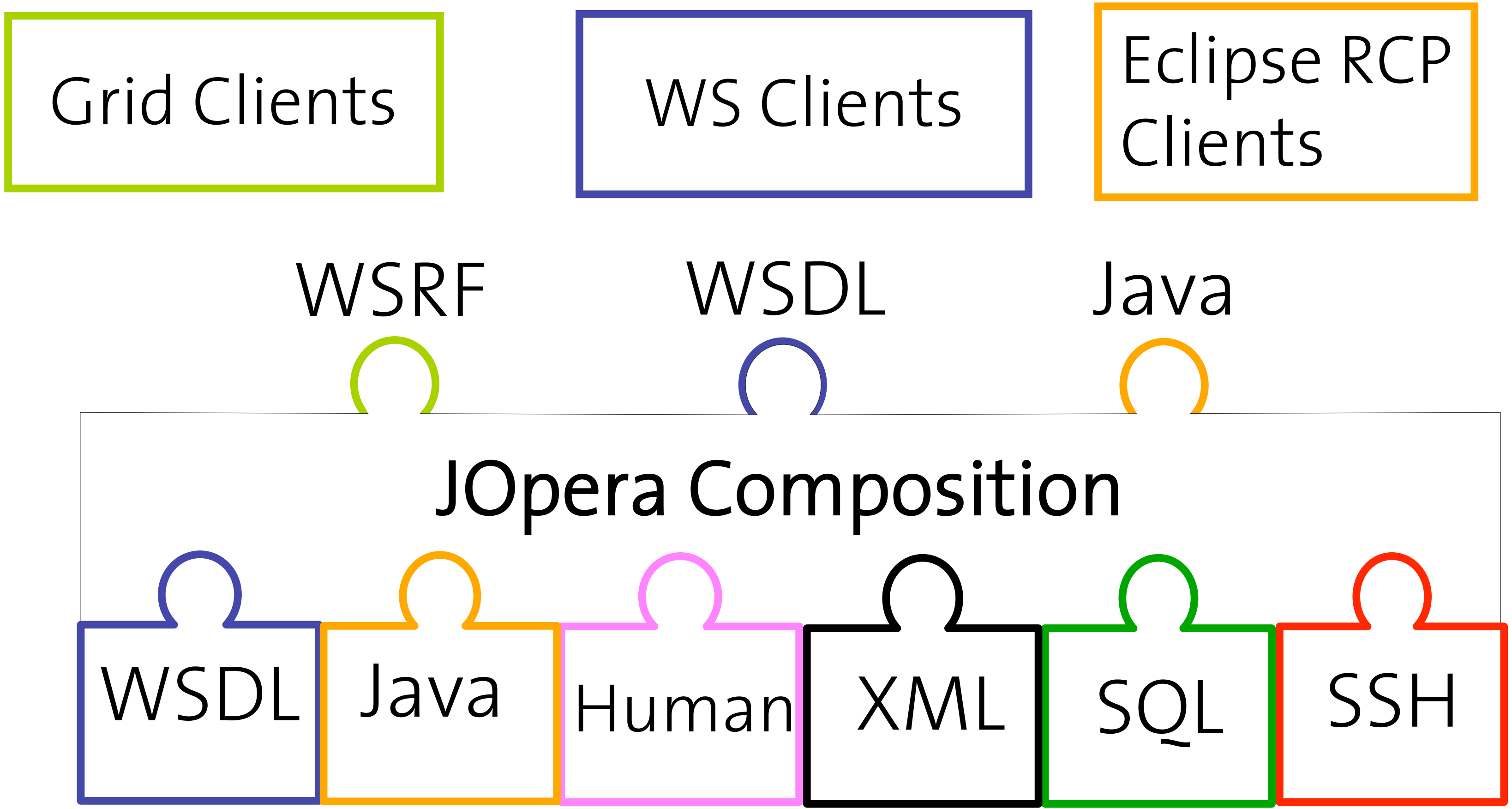
- Snippets
- Methods

- XSLT
- XPath

[VLDB/TES2004]

Publishing a composition with JOpera

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



Grid Clients

WS Clients

Eclipse RCP Clients

WSRF

WSDL

Java

JOpera Composition

WSDL

Java

Human

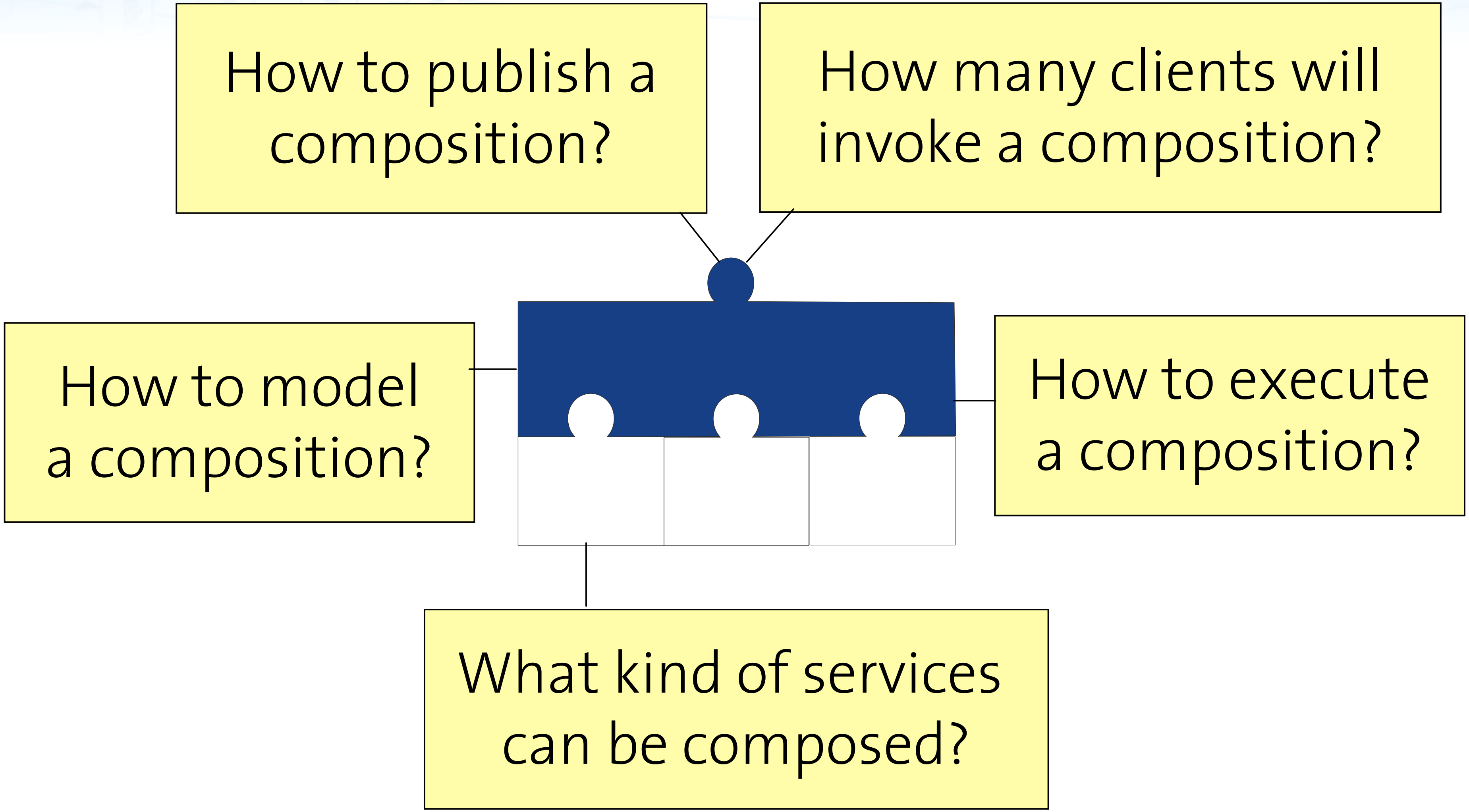
XML

SQL

SSH

[eScience2005, CCGrid2006]

Recursive Service Composition Problems



How to publish a composition?

How many clients will invoke a composition?

How to model a composition?

How to execute a composition?

What kind of services can be composed?

How many clients will invoke a process?

- Services built as **process-based compositions** of other services are published to be invoked by a large and unpredictable number of clients

Scalability on Clusters of Computers

- Process Management Infrastructure needs to scale (many clients, many conversations)
- Web Service Composition Engines run on **cluster** of computers to **handle large workloads [IJEC'04]**

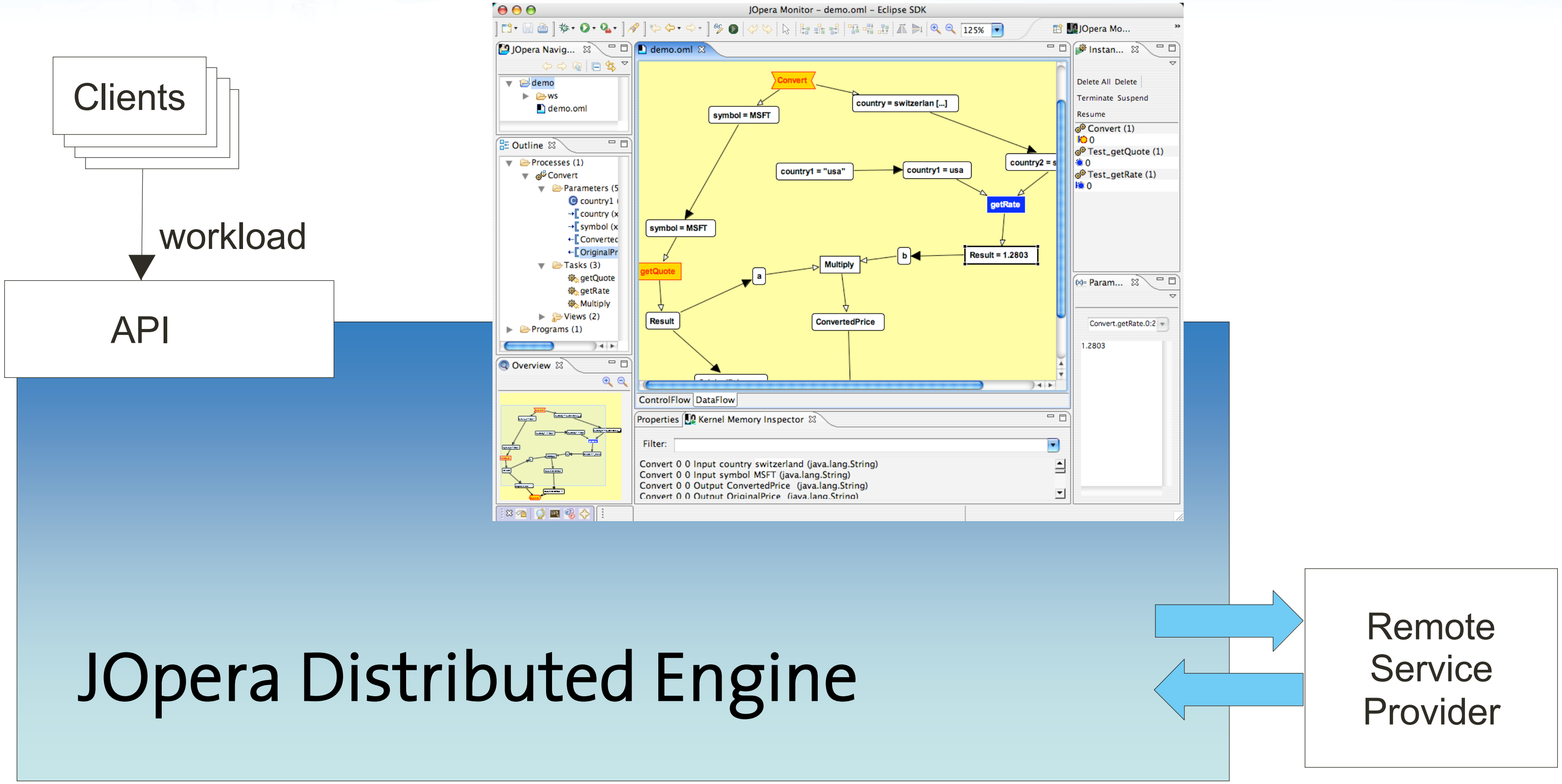
The Problem: How to Configure the Engine?

- The distributed engine needs to be configured:
  - Based on its **current (unpredictable) workload**
  - Based on the **available resources** of the cluster
- How many resources of the cluster should be assigned to the engine?
- Difficult to configure the engine *apriori*
- Difficult to manage the system *manually*

The Solution: Autonomic Computing

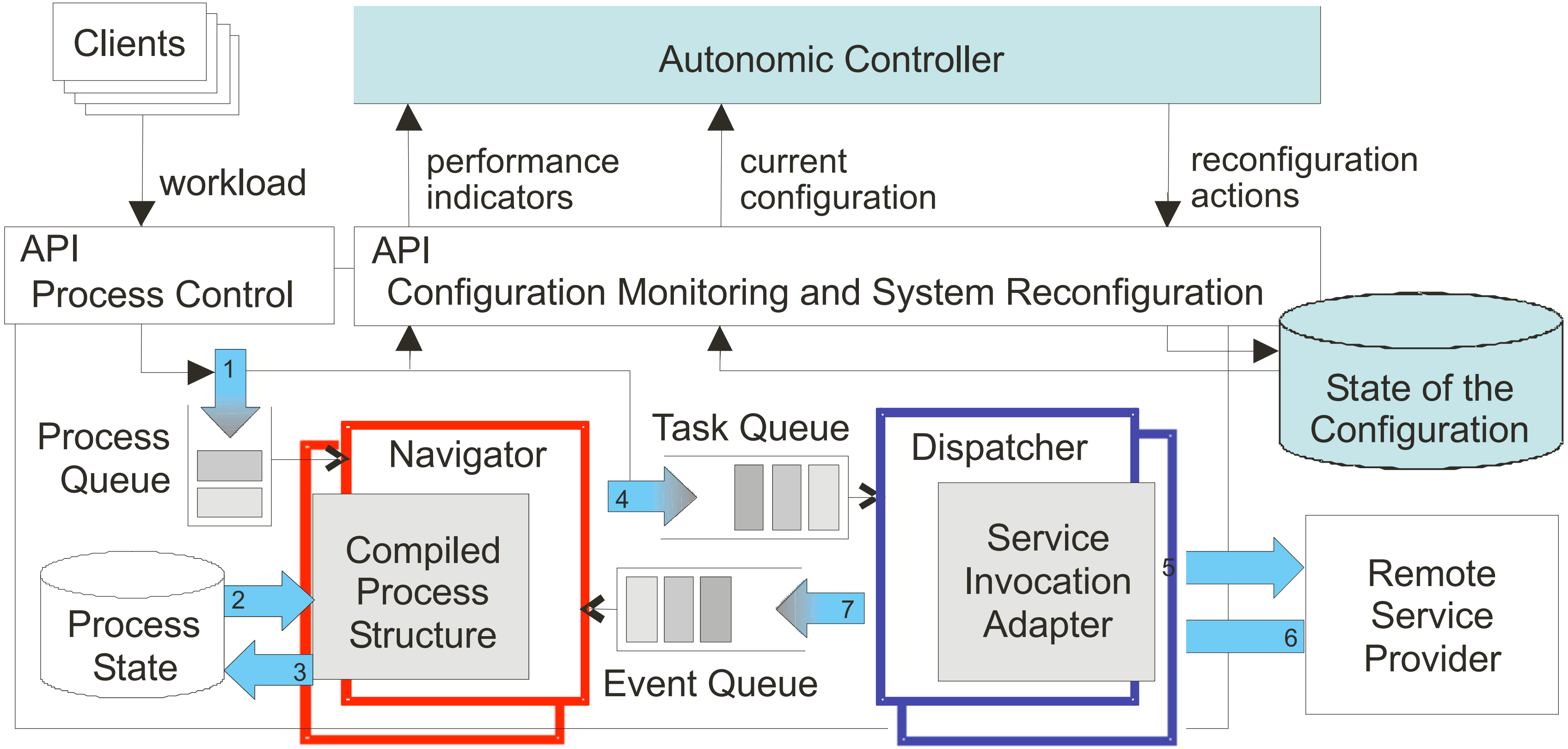
- The engine should configure itself
- Trade-off between two goals:
  - *Best Performance (response time, throughput, …)*
  - *Best Resource Allocation (size of the cluster)*
- Requirements for the distributed engine design:
  - Support on-the-fly reconfiguration
  - Provide access to internal performance metrics
  - Expose an API for controlling the configuration

JOpera Distributed Engine Architecture

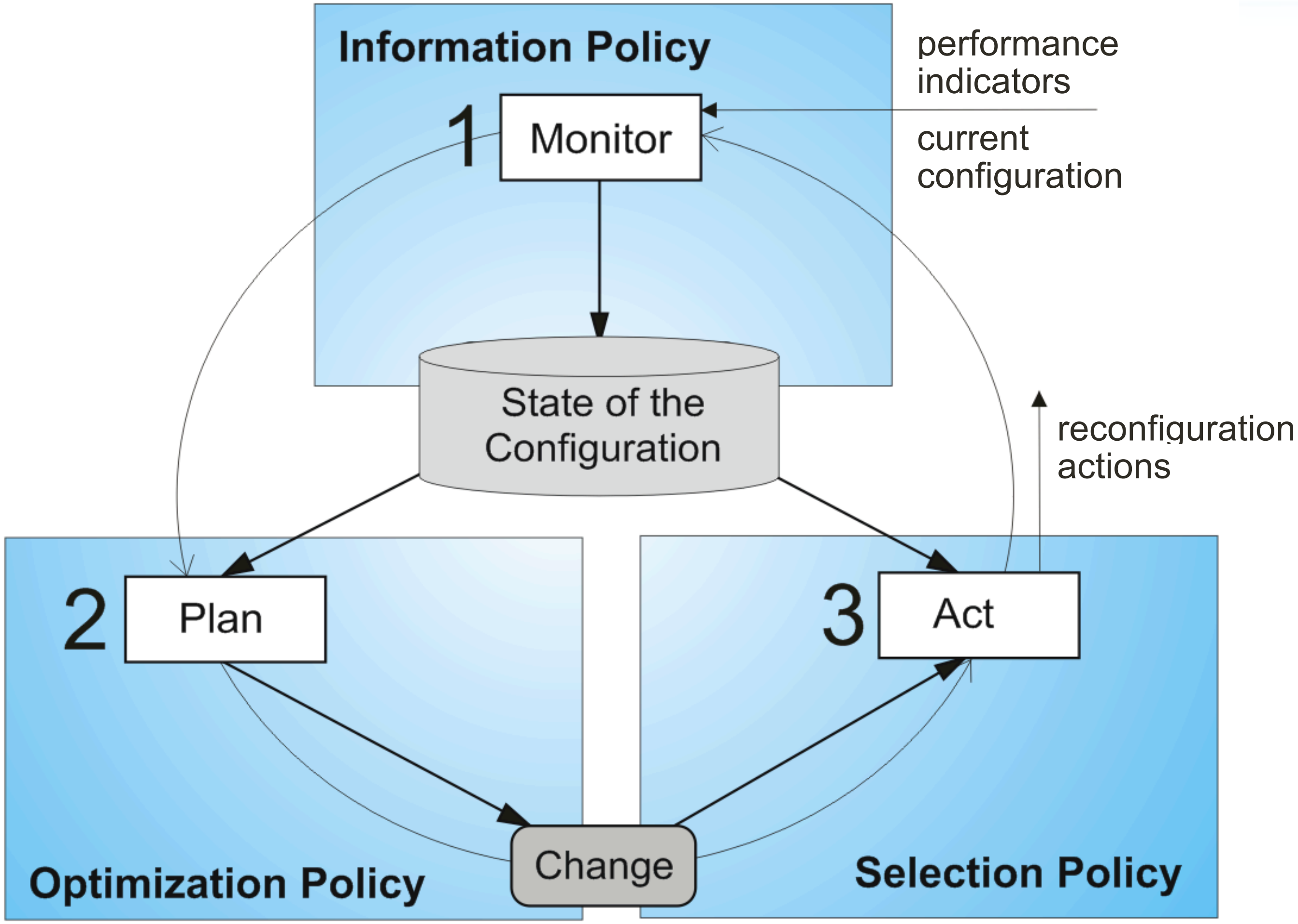


```
Convert 0 0 Input country switzerland (java.lang.String)
Convert 0 0 Input symbol MSFT (java.lang.String)
Convert 0 0 Output ConvertedPrice  (java.lang.String)
Convert 0 0 Output OriginalPrice  (java.lang.String)
```

Adding Self-Management



Autonomic Controller Algorithm



Autonomic Controller Policies

- Information Policy
  - Define which variables should be monitored

  Queue Length, Number of Navigator/Dispatcher Threads

- Optimization Policy
  - Map Monitored Variable to Reconfiguration Actions

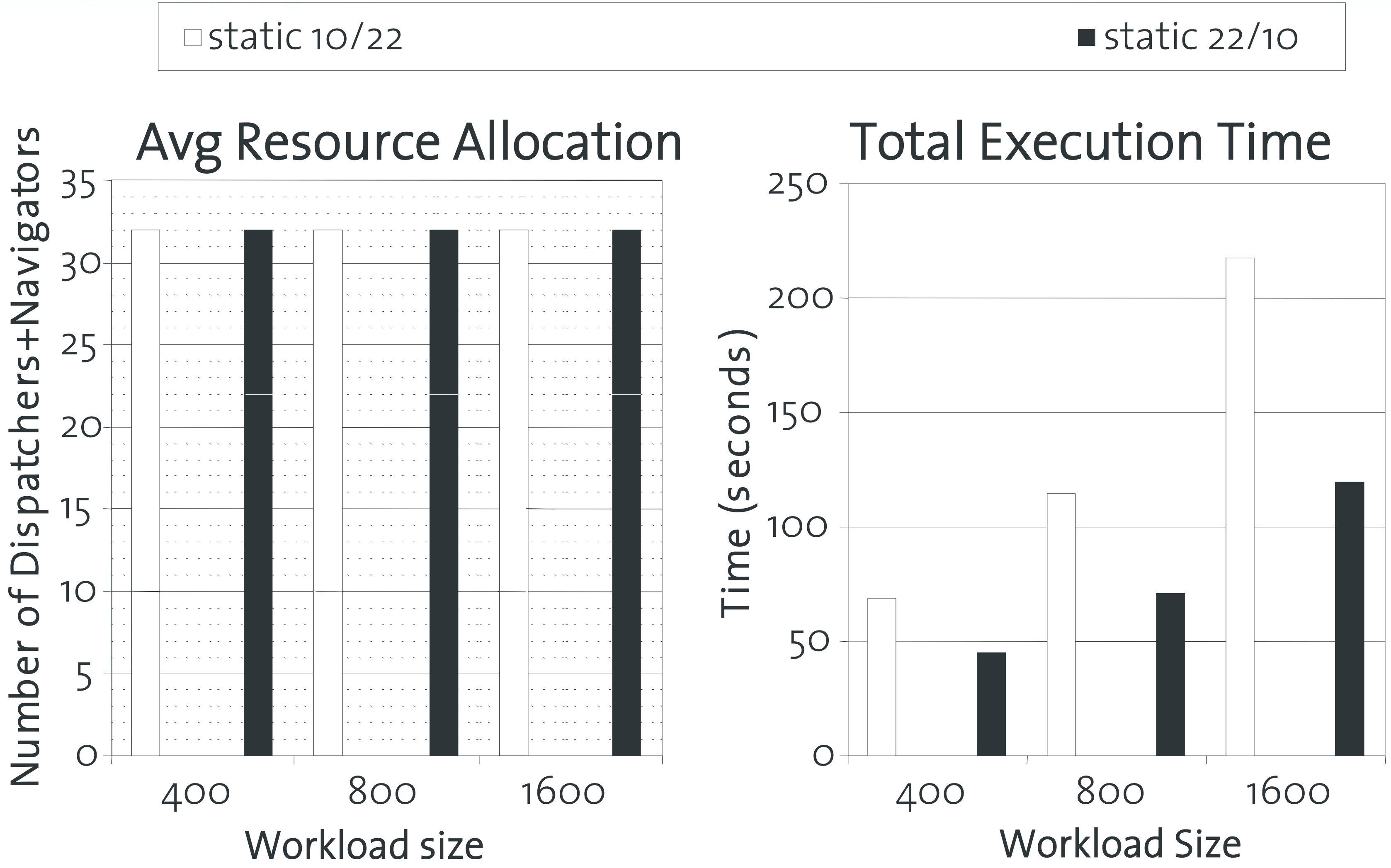
    1. Simple Threshold Policy
    2. Differential Policy
    3. Proportional Policy

- Selection Policy
  - Choose how to implement a reconfiguration plan

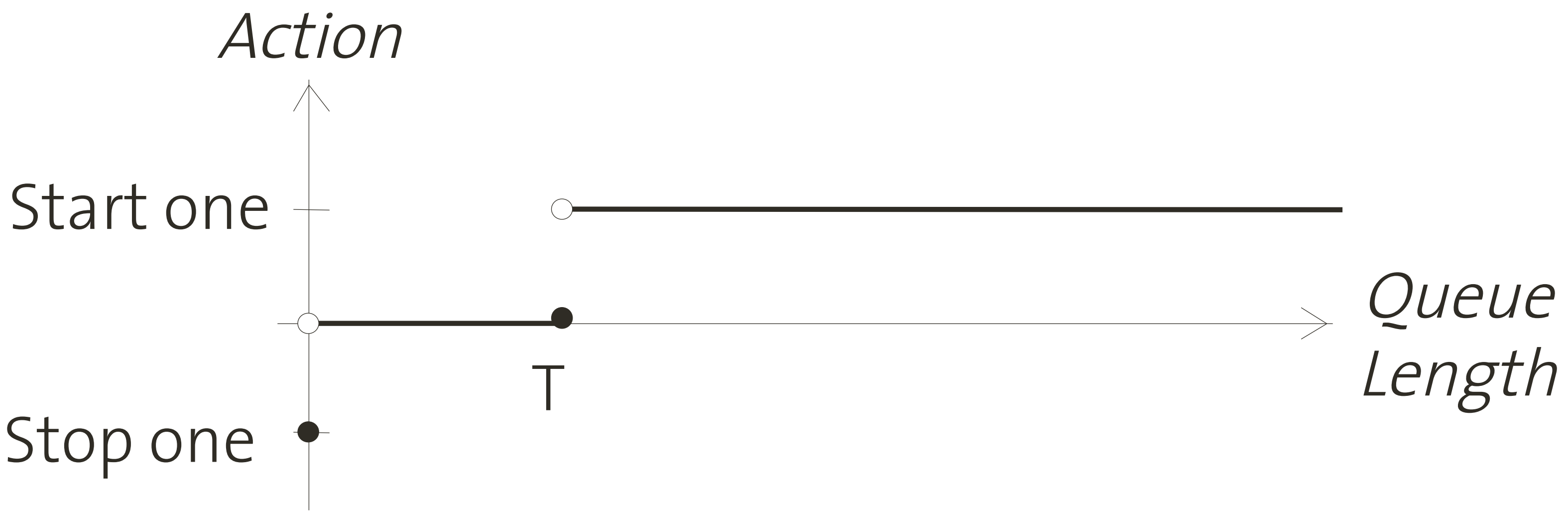
Evaluation of the Control Policies

- Workload: **Peak Response Benchmark**
  - 800 concurrent processes initiated at the same time
- Performance Indicators:
  - Total Execution Time
  - Average Resource Allocation
- 32 node cluster environment (one thread/node)
- Baseline: **Static Manual Configuration**
  - Fast: 10 Navigators, 22 Dispatchers
  - Slow: 22 Navigators, 10 Dispatchers

Baseline: Slow/Fast Static Configuration

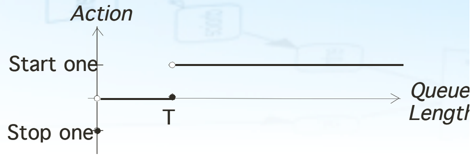


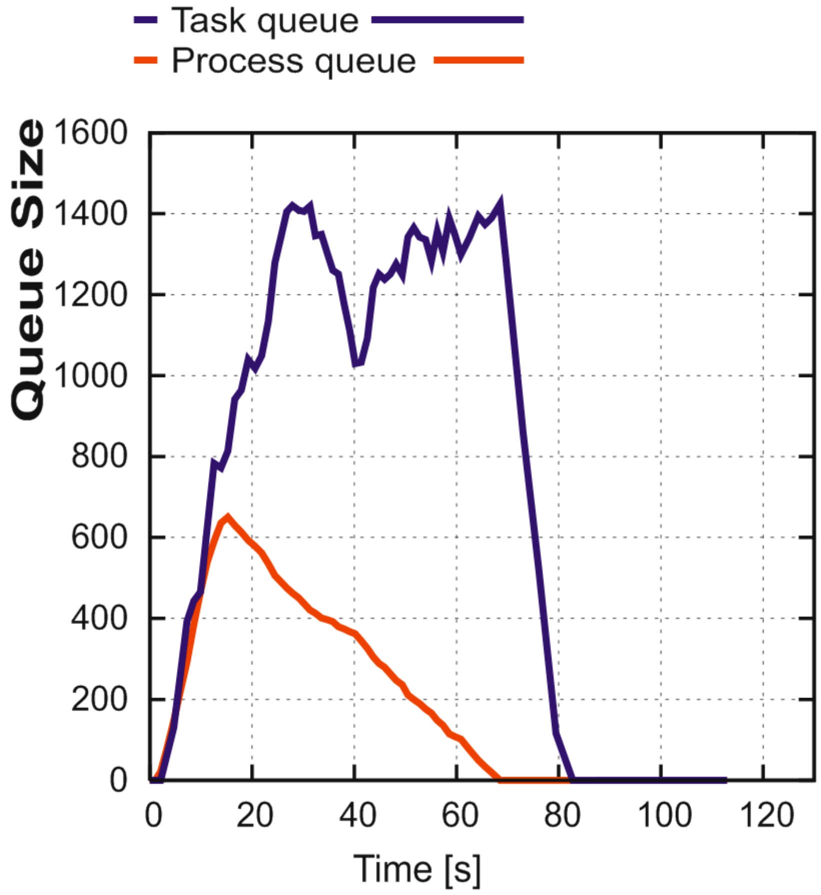
1. Simple Threshold Policy

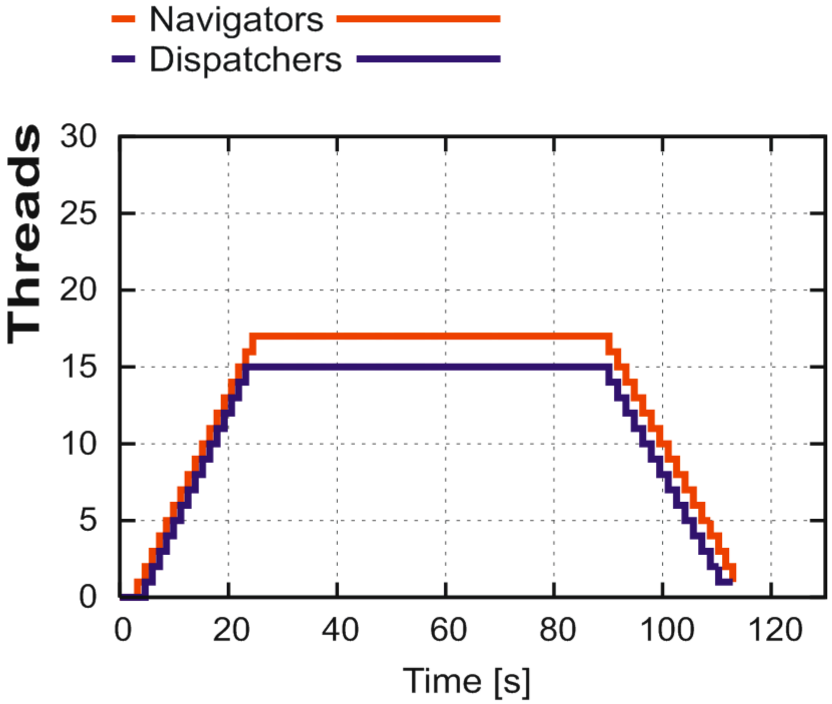


- Start one thread if Queue Length > T
- Stop one thread if Queue Length = 0

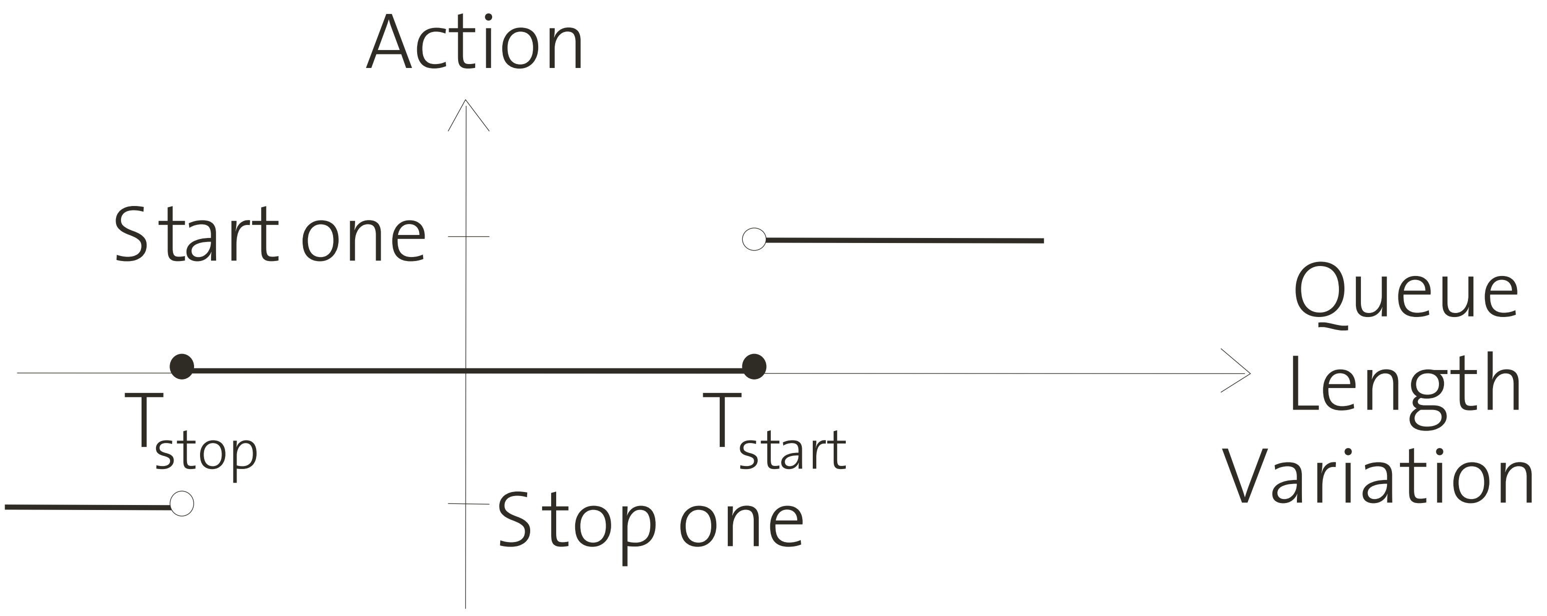
Tracing the Simple Threshold Policy





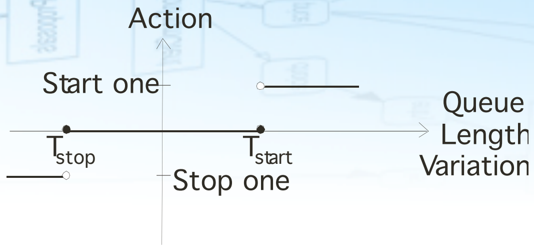


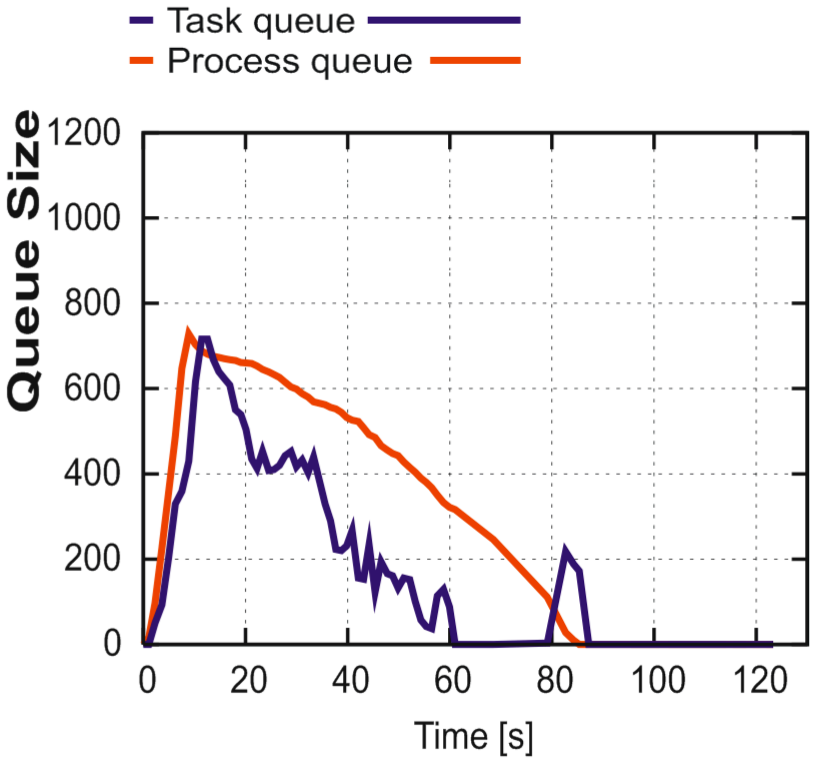
2. Differential Policy

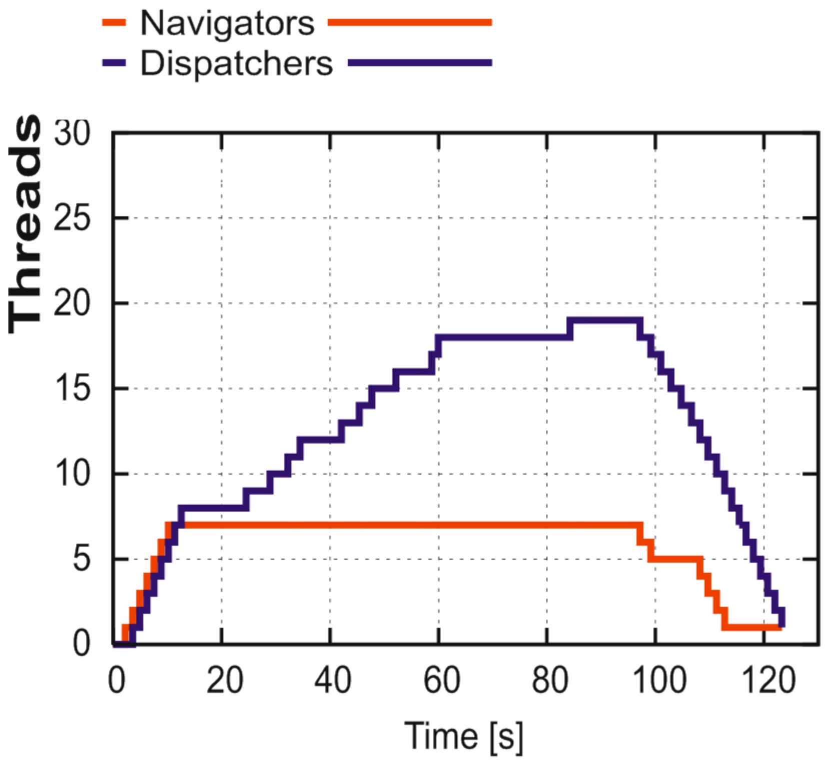


- Start one thread if Queue Length Variation > T_{start}
- Stop one thread if Queue Length Variation < T_{stop}

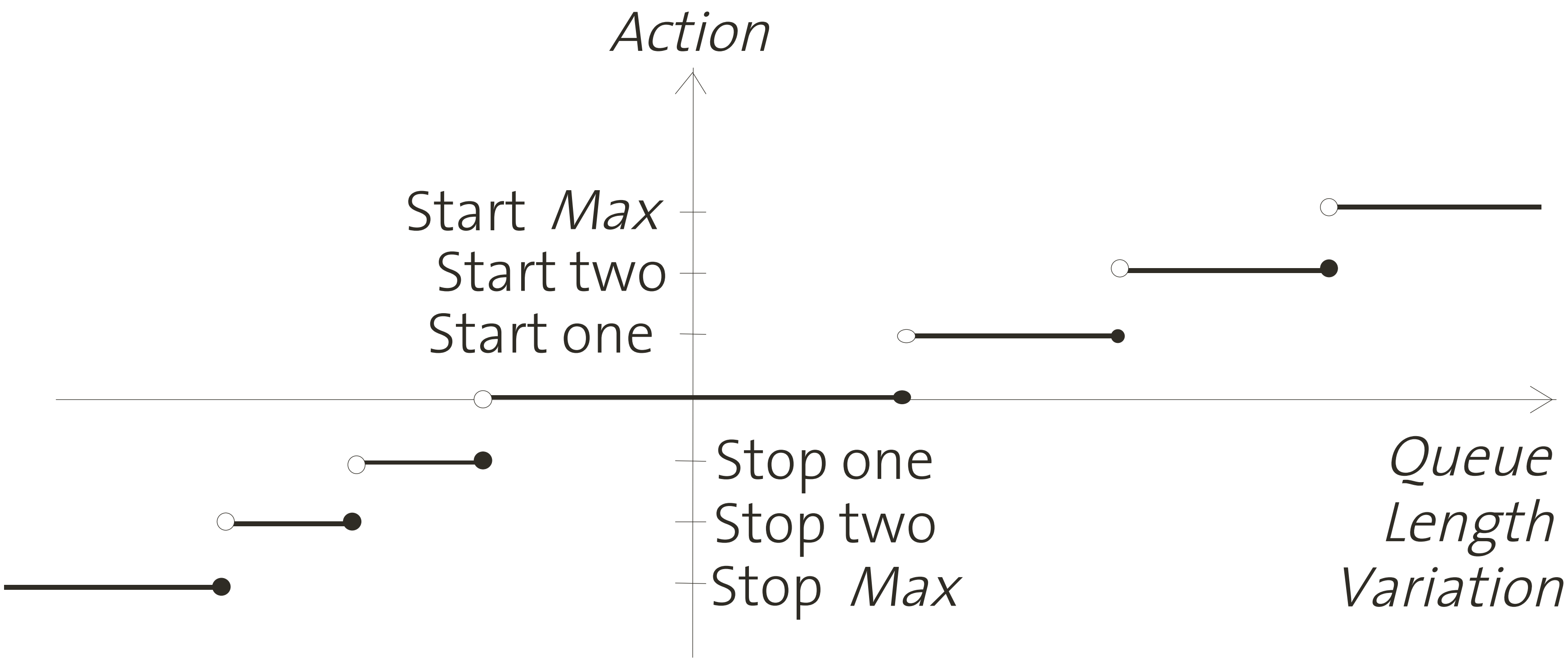
Tracing the Differential Policy



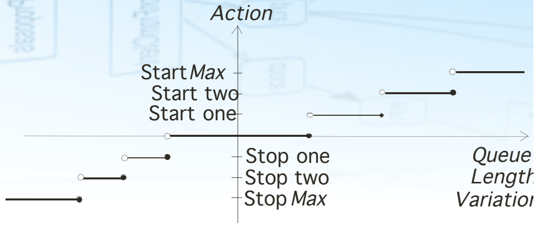




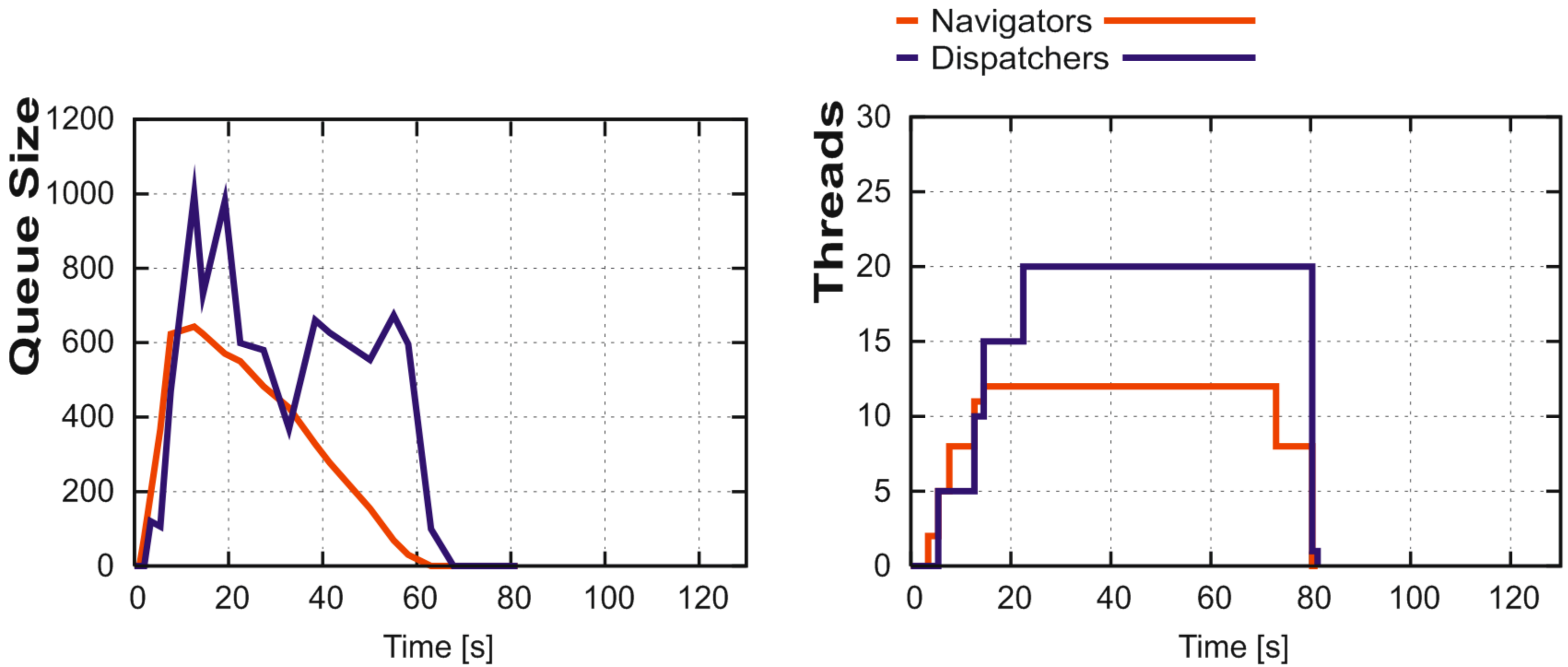
3. Proportional Policy



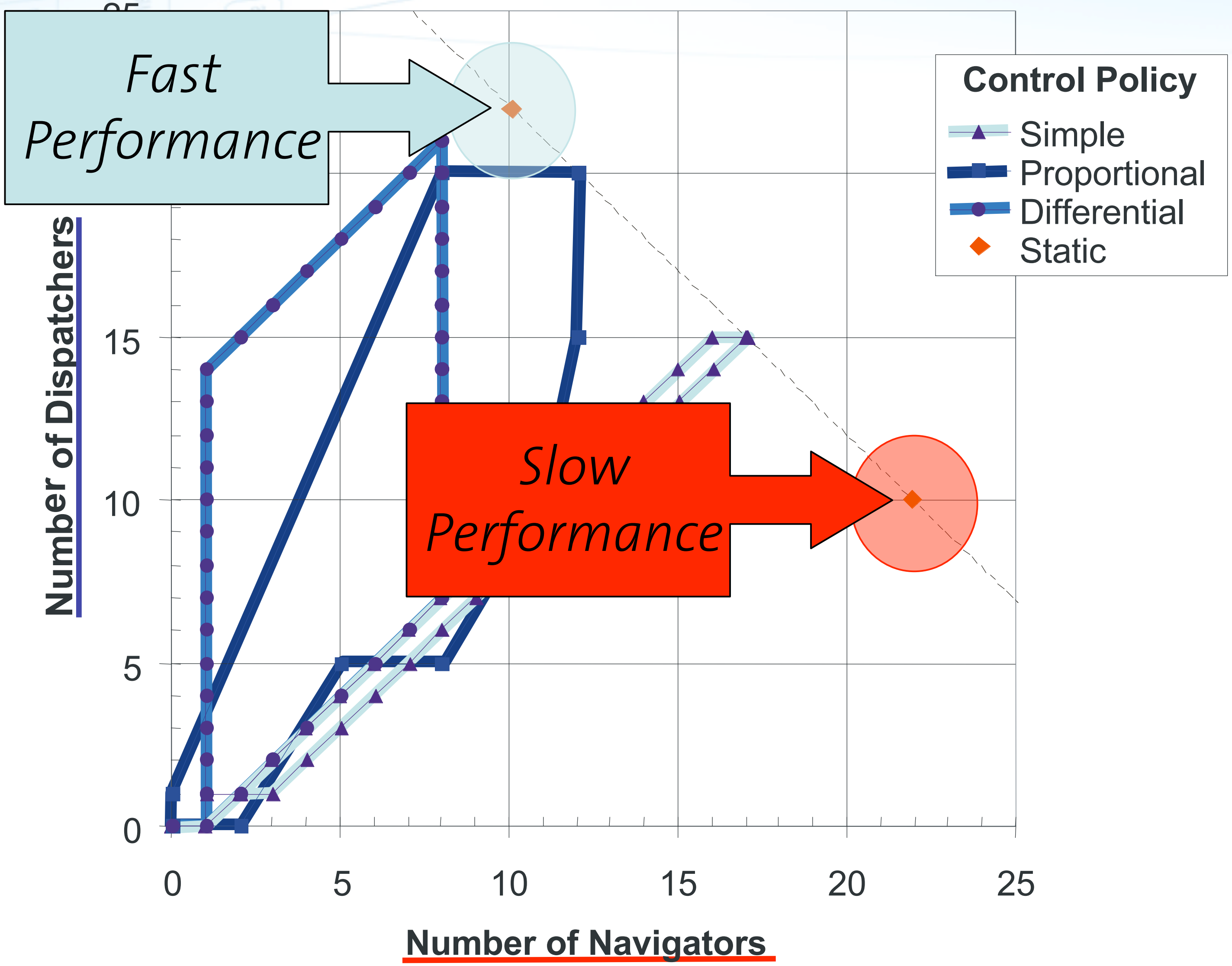
- Start/Stop *N* threads, proportional to the Queue Length Variation



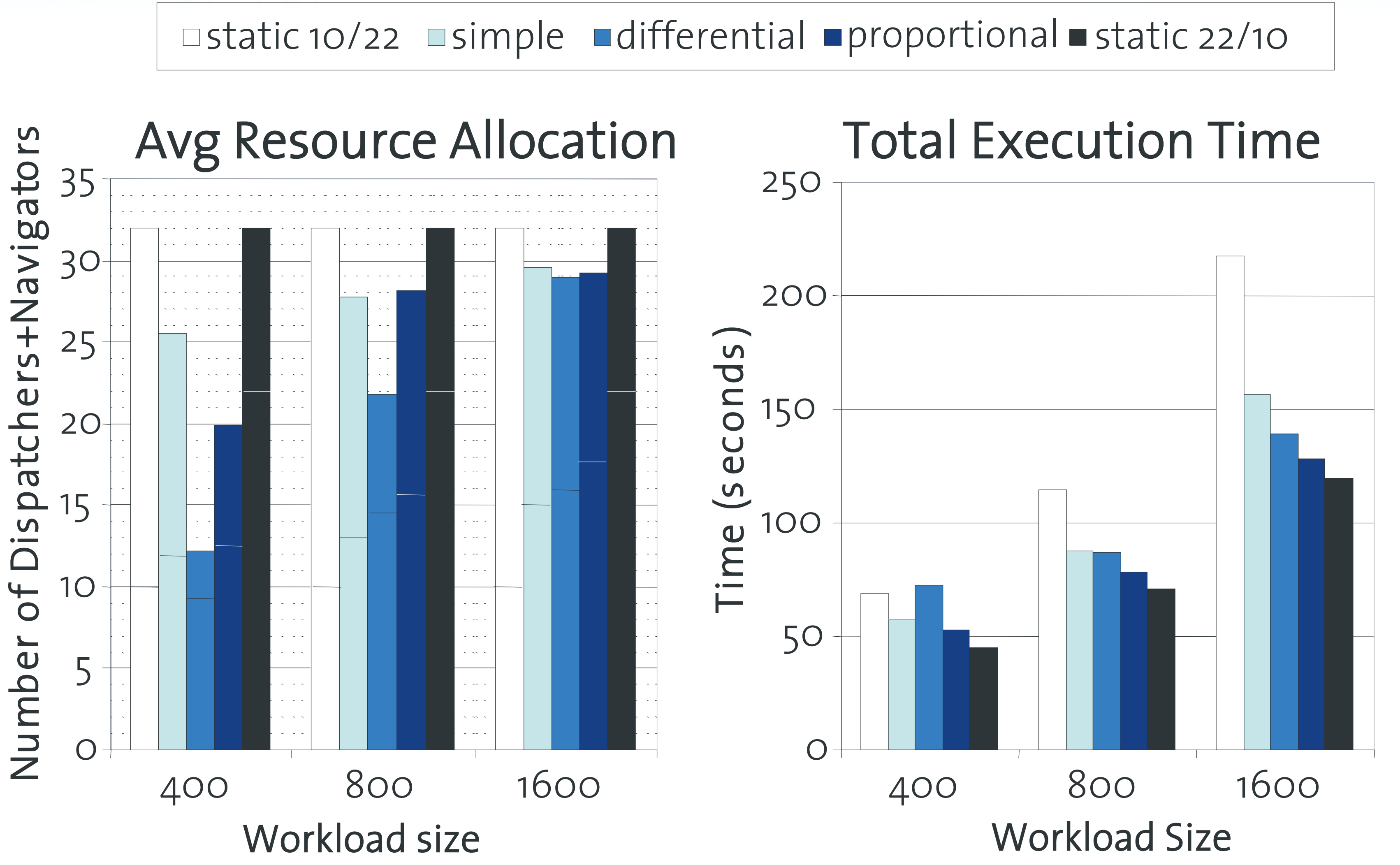
Tracing the Proportional Policy



State Space Comparison of the Policies



Performance Comparison of the Policies



Autonomic Execution Summary

- **Manual** configuration & management of a distributed process-based Web service composition engine is difficult and expensive
- To address this problem, we have shown how to apply **autonomic computing** techniques
- Our evaluation indicates that different control policies can be used to explore the trade-off between performance vs. resource utilization

[ICAC2005, ICWS2005]

Conclusion

- **Modeling** service composition behavior
  - Process-centric **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.

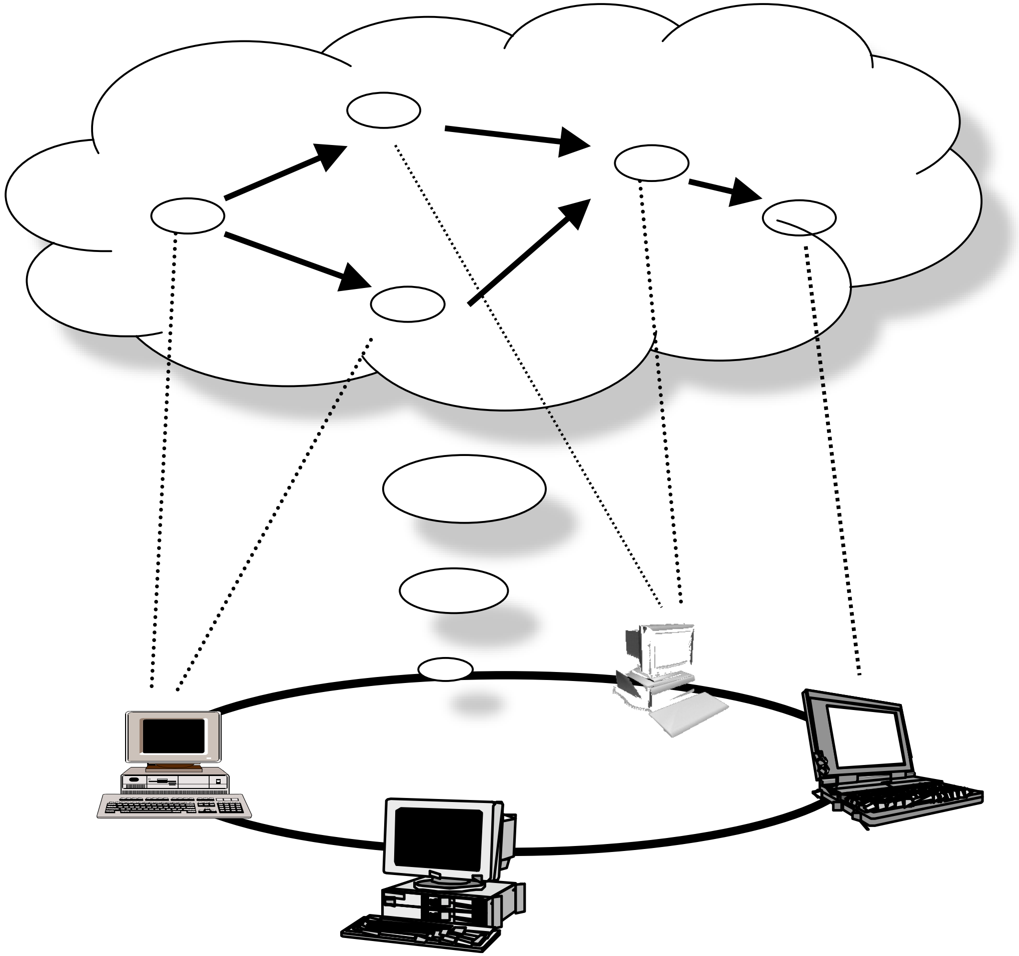
[e-SCIENCE2005] Thomas Heinis, Cesare Pautasso, Oliver Deak, Gustavo Alonso, **Publishing Persistent Grid Computations as WS Resources**, accepted to the 1st IEEE International Conference on e-Science and Grid Computing (e-Science 2005), Melbourne, Australia, December 2005.

[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

JOpera Demo

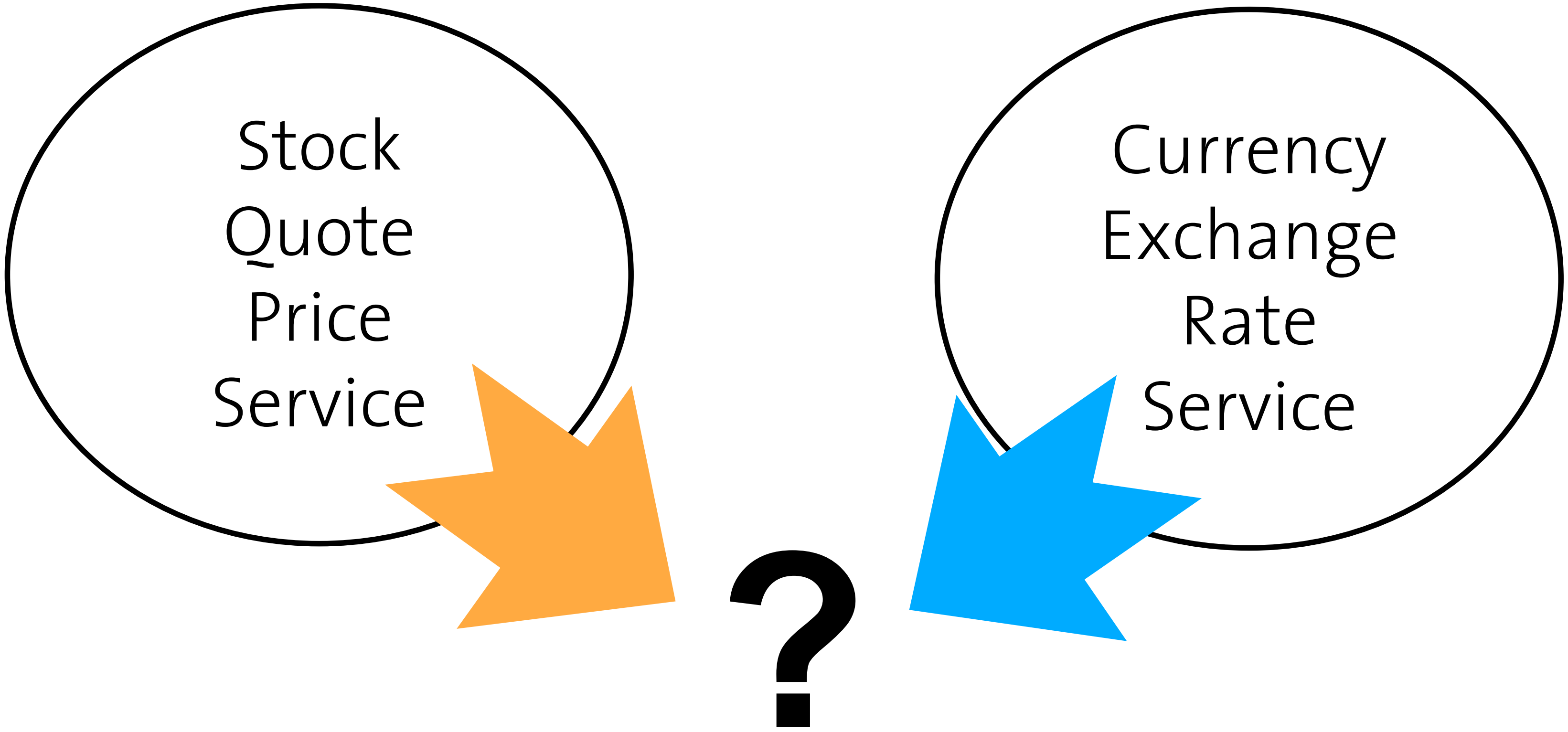


Demo: Bottom-up and Top-down Composition

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. Define what services are missing and add the necessary code snippets
5. Publish the process as Web Service

Example Scenario

- Stock Quote Currency Conversion

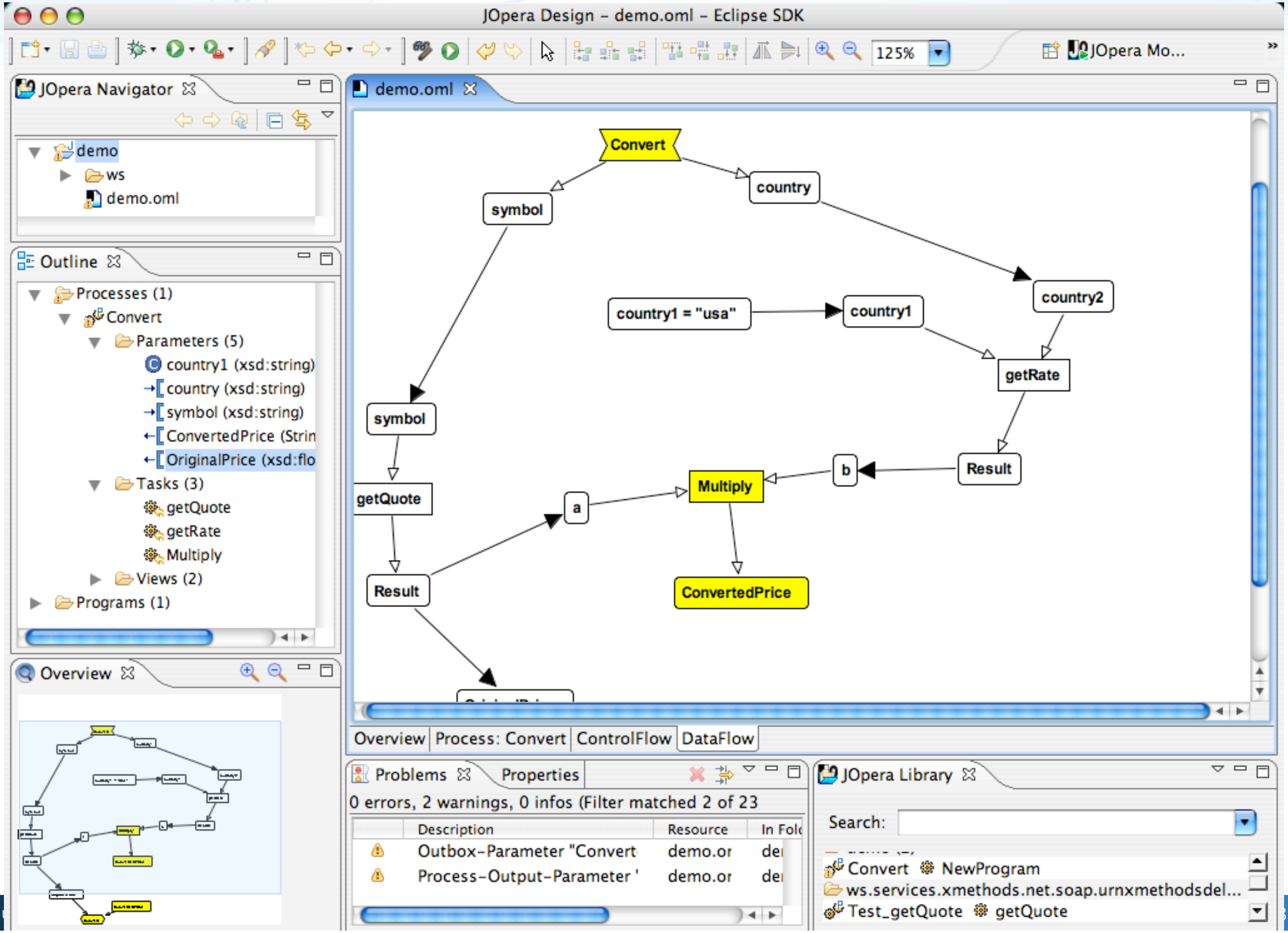


Stock Quote Price Service

Currency Exchange Rate Service

?

Drag, Drop and Connect



JOpera Design – demo.oml – Eclipse SDK

JOpera Navigator

- demo
  - ws
  - demo.oml

Outline

- Processes (1)
  - Convert
    - Parameters (5)
      - country1 (xsd:string)
      - country (xsd:string)
      - symbol (xsd:string)
      - ConvertedPrice (Strin
      - OriginalPrice (xsd:flo
    - Tasks (3)
      - getQuote
      - getRate
      - Multiply
    - Views (2)
- Programs (1)

Problems | Properties

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

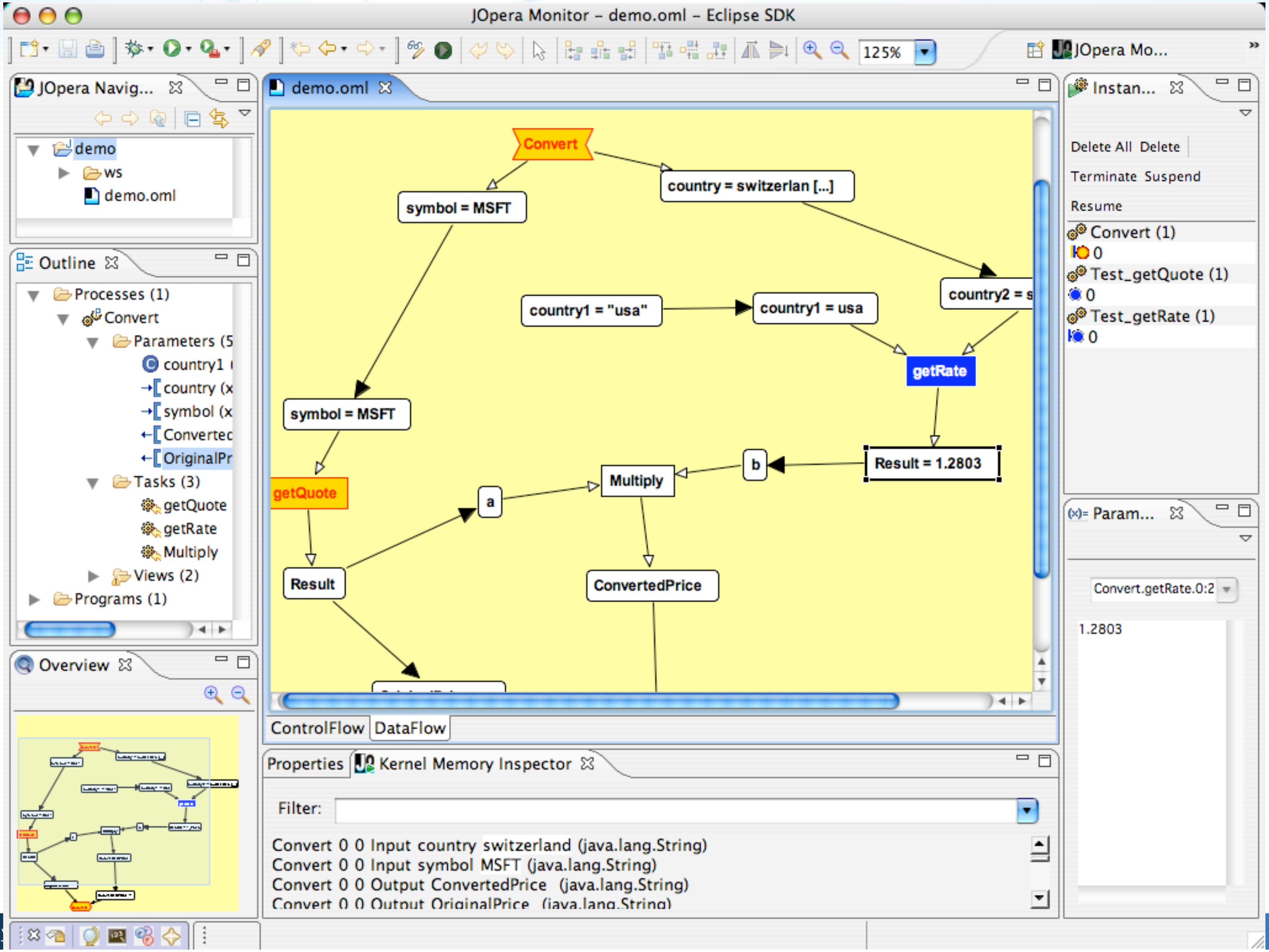
Description	Resource	In Fold
Outbox-Parameter "Convert	demo.or	de
Process-Output-Parameter '	demo.or	de

JOpera Library

Search:

- Convert NewProgram
- ws.services.xmethods.net.soap.urnxmethodsdel...
- Test_getQuote getQuote

Run, Monitor, Steer and Debug



JOpera Monitor – demo.oml – Eclipse SDK

Convert.getRate.0:2

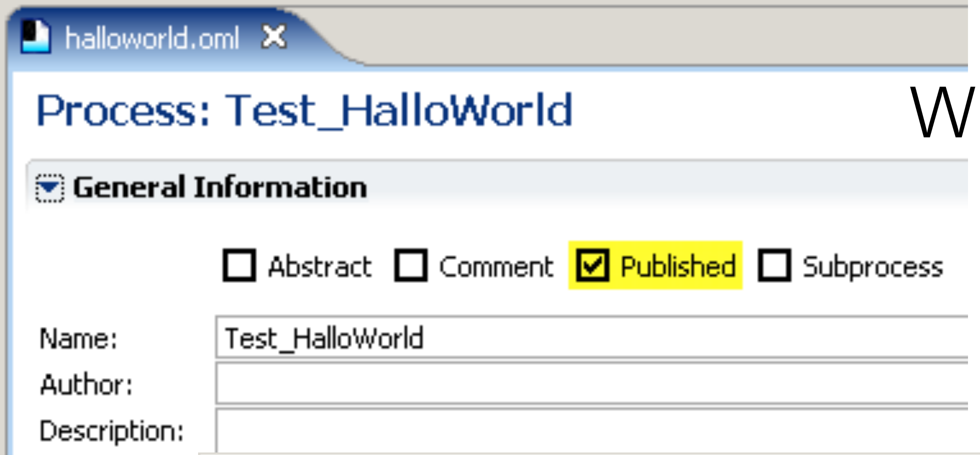
1.2803

Filter:

```
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!



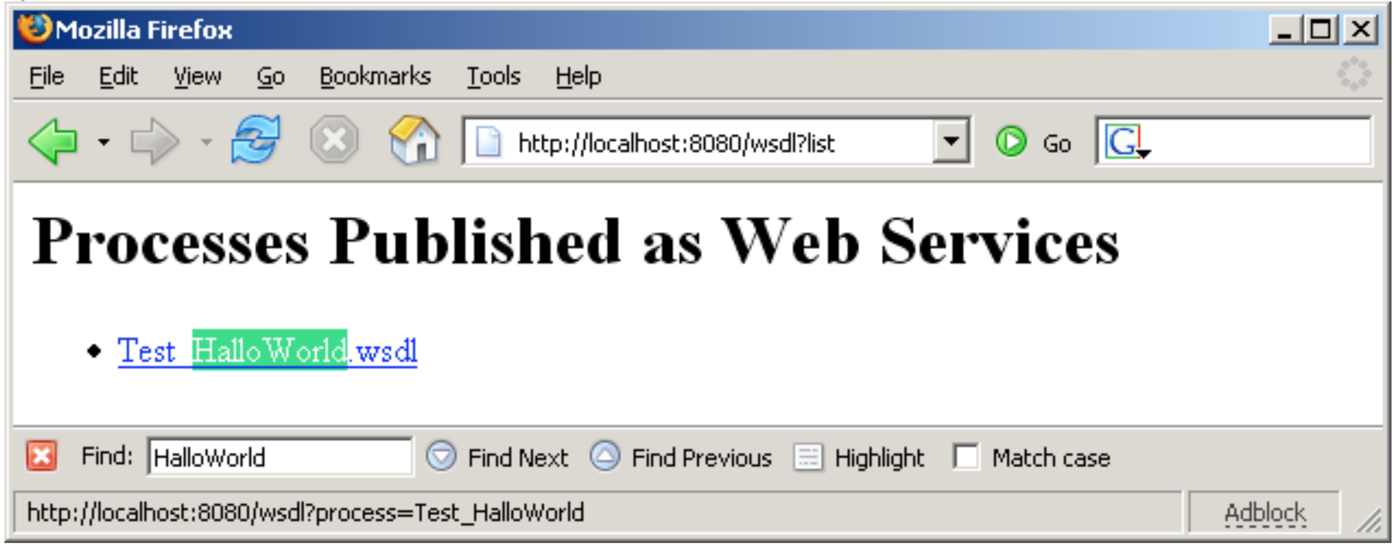
halloworld.oml

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

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

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IKS *Information and Communication Systems Research Group*

JOpera: an Autonomic Platform for Service Composition

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