Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A Flexible System for Visual Service Composition

Cesare Pautasso

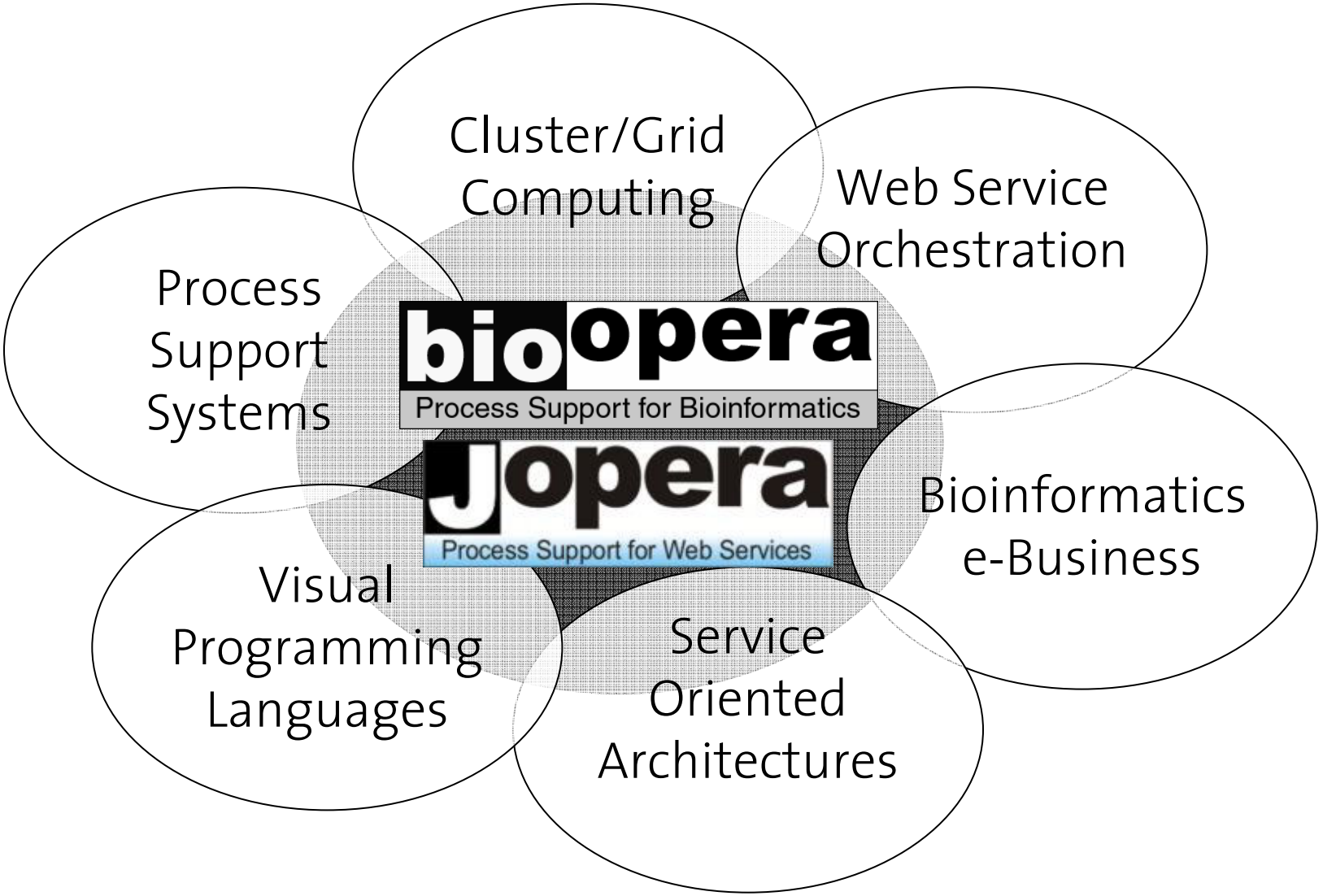
Information and Communication Systems Research Group
Department of Computer Science
Swiss Federal Institute of Technology, Zurich (ETHZ)

pautasso@inf.ethz.ch

http://www.iks.ethz.ch/jopera

30.4.4

Context


Cluster/Grid Computing
Web Service Orchestration
Process Support Systems
bioopera
Process Support for Bioinformatics
Jopera
Process Support for Web Services
Bioinformatics e-Business
Visual Programming Languages
Service Oriented Architectures

REVIEW

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*.

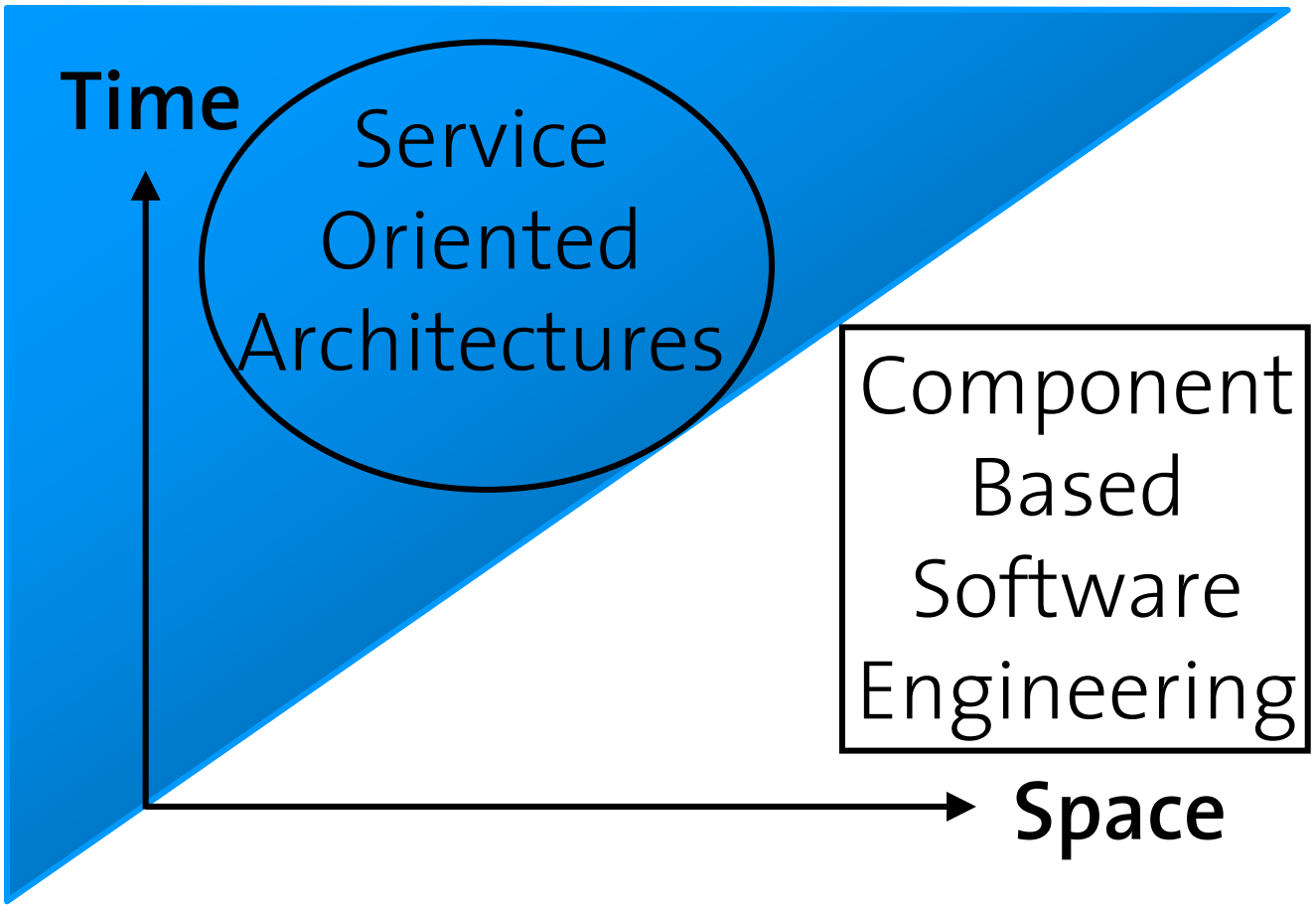
Gio Wiederhold
Peter Wegner
Stefano Ceri
Communications of the ACM
Volume 35
Issue 11
November 1992
Pages: 89 - 99

Outline

1. A **Language** for Service Composition
   - Visual syntax
   - Process oriented

2. A **System** for Service Composition
   - Efficiency through compilation
   - Flexibility

How to model composition



Web Services should be composed with a visual programming language

print “hello world!”

```
<!-- HelloWorld BPEL Process -->
<process name="helloworld"
         targetNamespace="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
             -->
        <variable name="output"
```

Web Service Composition/XML

- Many non visual standards (XML based) for composition have been proposed:
  - BPEL4WS (XLANG/WSFL)
  - WS-Coordination/WSCL/WSCI
  - BPSS (ebXML)
- XML is great for interoperability between software systems, but not well suited for human programmers
- Most existing composition tools work directly or just above the XML

How to program Composition

Process Modeling Languages (Workflow Systems)

XML-based (**BPEL4WS**+**BPML**)

Visual Composition Languages (**JOpera**)

Traditional Programming Languages (**Java, C#**)

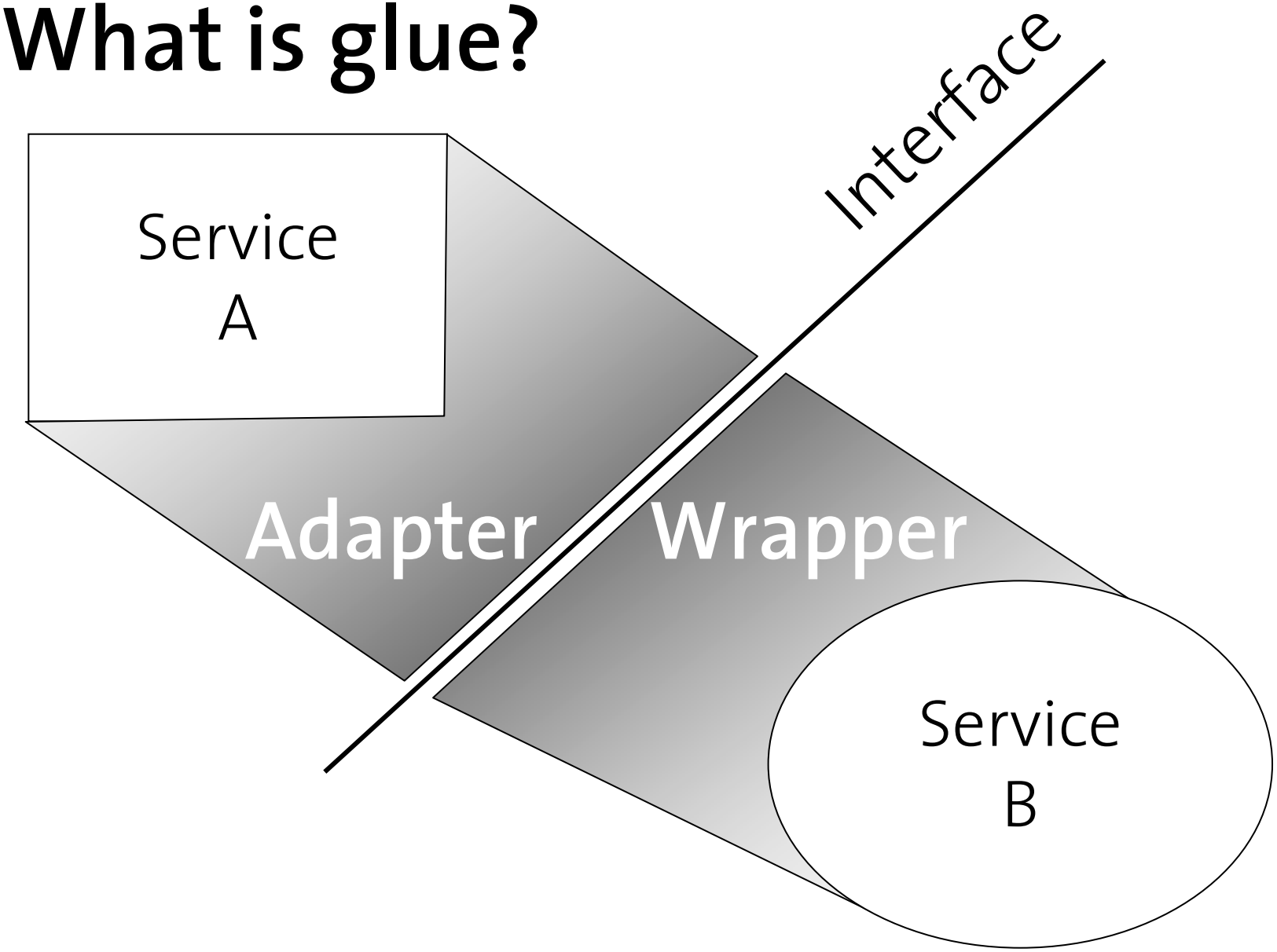
Data Model-driven Languages (**WebML**)

Ad-hoc Languages (**XL**)

Our Goals

1. Glue language for Web Services
   - Define an intuitive, graph-based, simple visual composition language
   - Use *ad hoc* constructs and extensions sparingly
   - Keep XML hidden at all times
2. Rapid, user-friendly development of applications composed of services
   - Adapt mismatching service interfaces
   - Use (if possible) semantics to guide the user

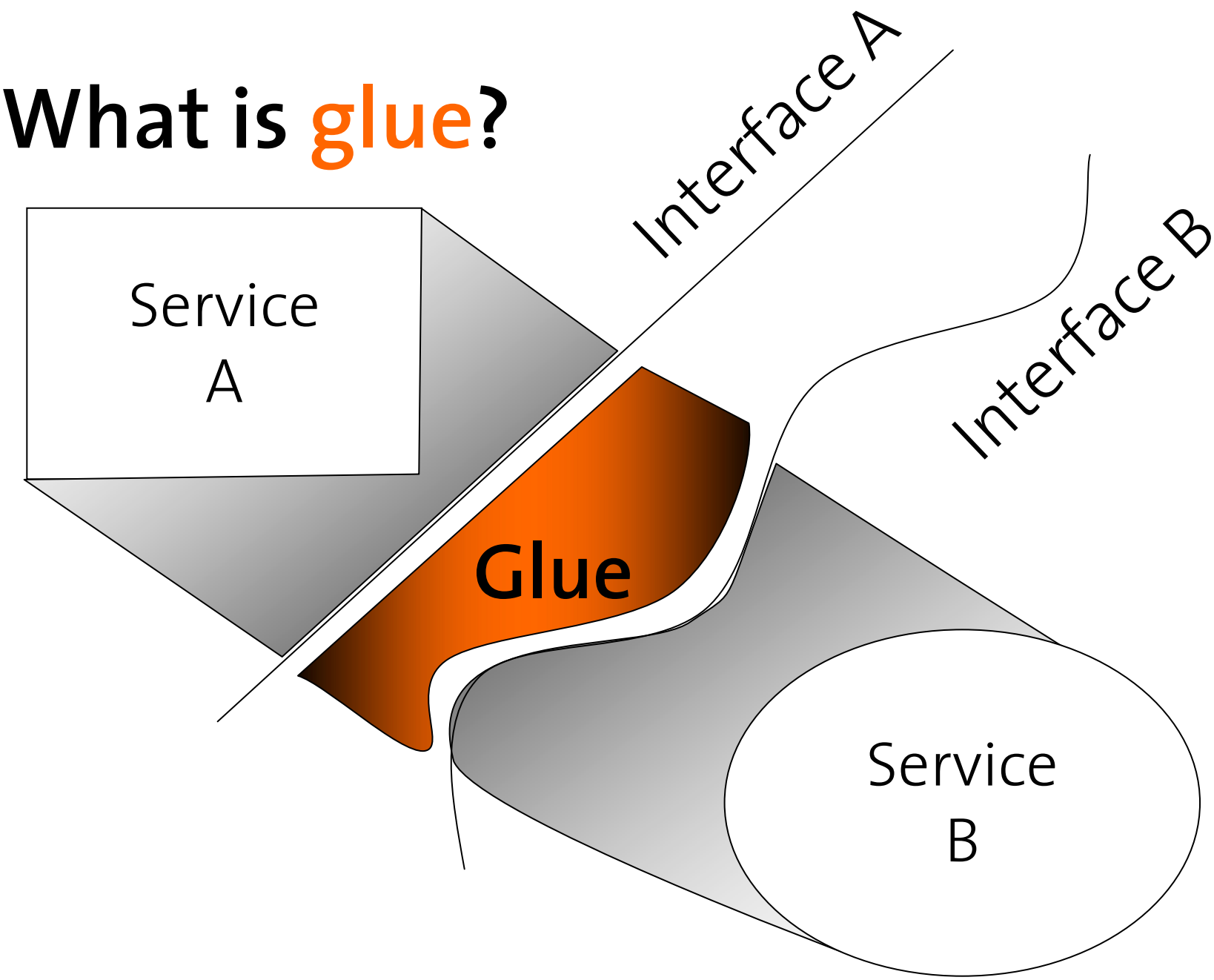
What is glue?



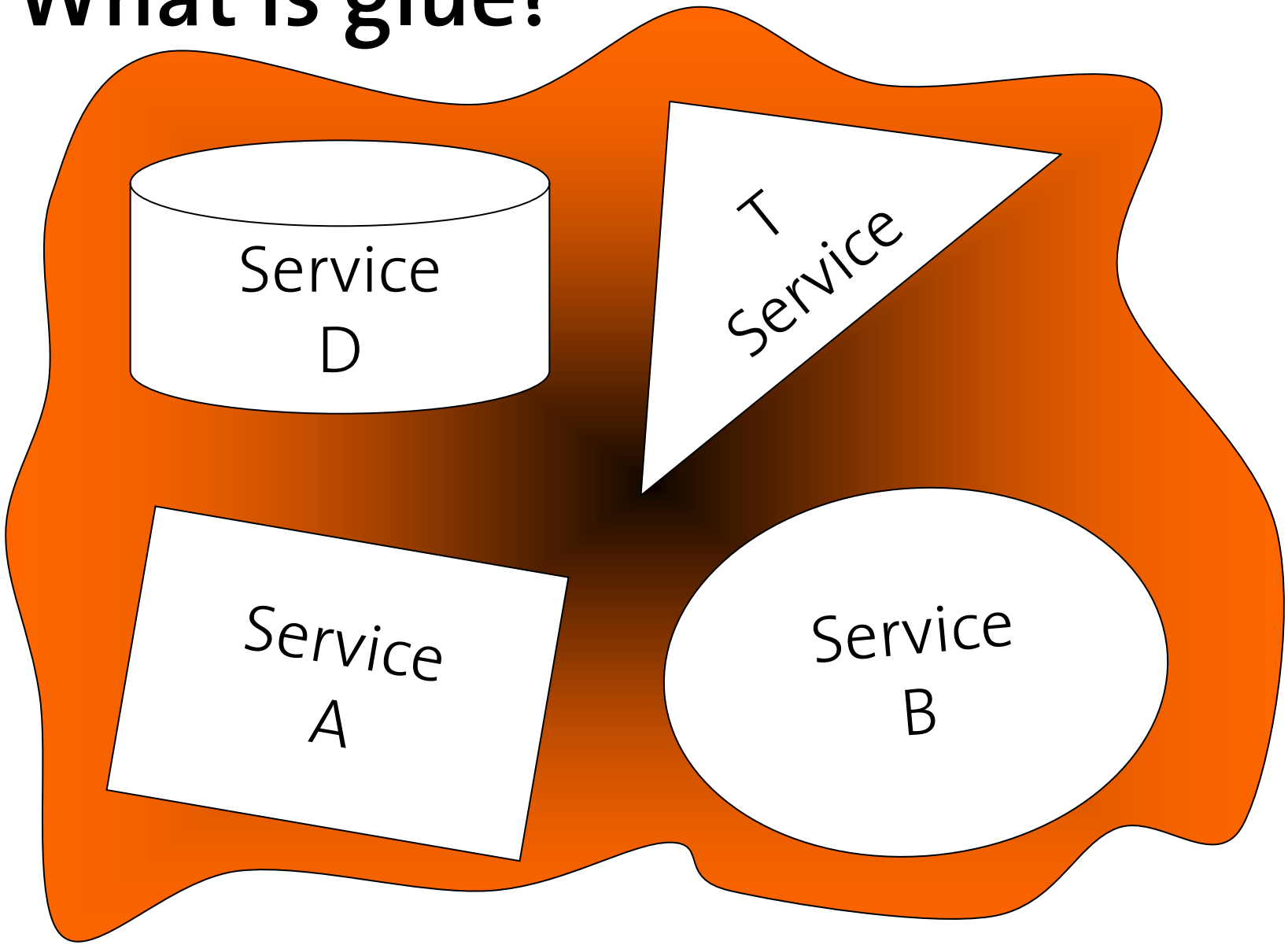
Wrappers and Adapters

- Extend an existing component
  - Adapt the interface to a different protocol
  - Change the data representation format
  - Switch between synch/a-synch invocation
  - Increase Security (Access Control, Client Authentication, Encryption)
  - Synchronize Access (Locking)
- In most cases, the original component is not modified when it is wrapped

What is glue?

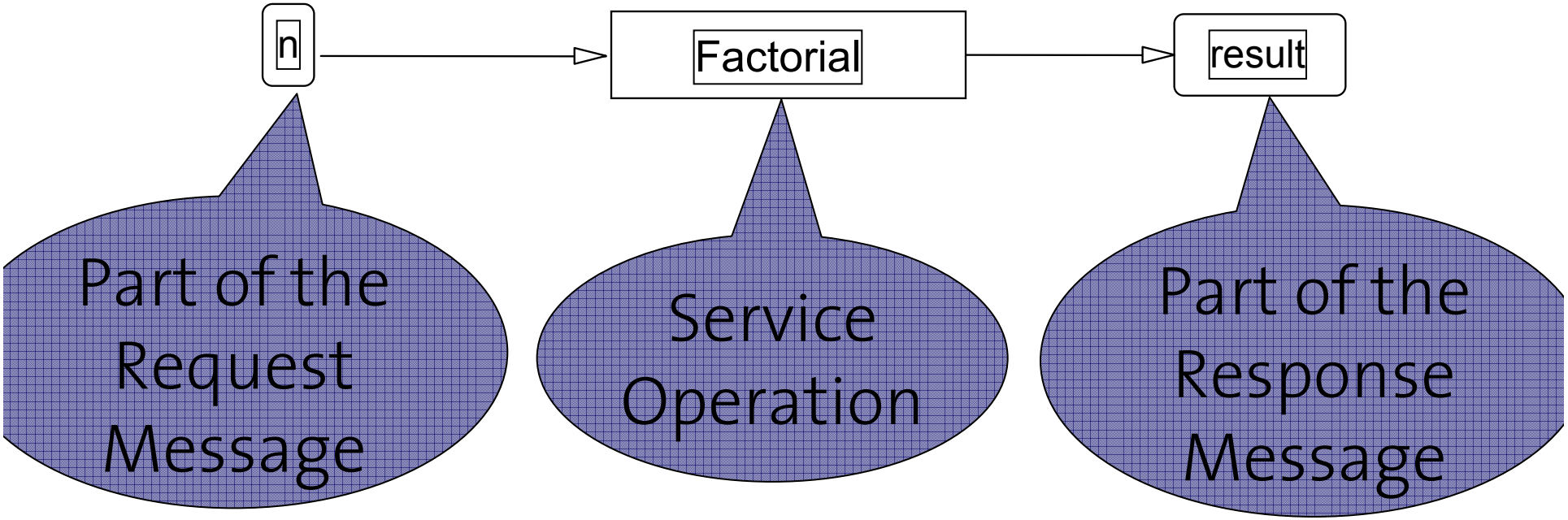


What is glue?



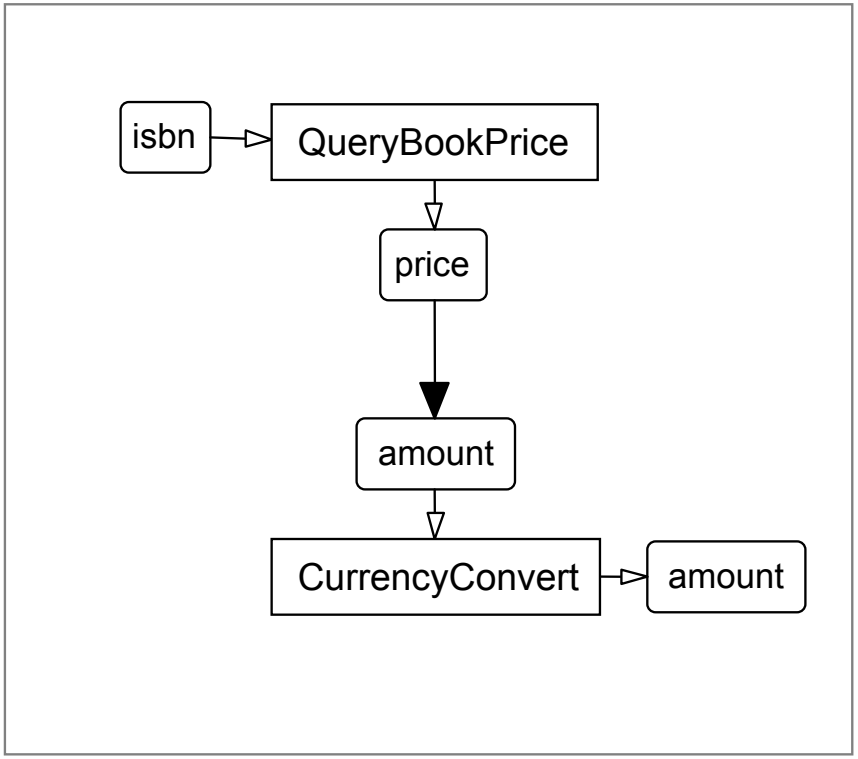
A Web service

- A Web service involves the execution of some **operation** which may require some **input** data parameters and may produce some **output** result

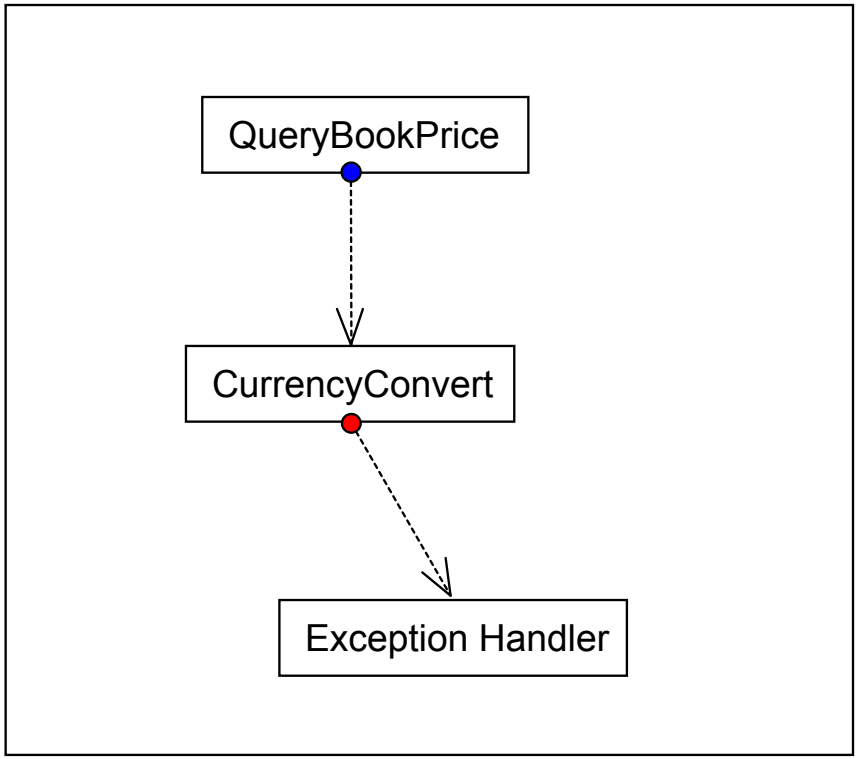


JOpera Visual Composition Language

- Web Services are composed using a **Process**, defined by two separate graphs:
- **Data Flow**



- **Control Flow**



JOpera Visual Composition Language

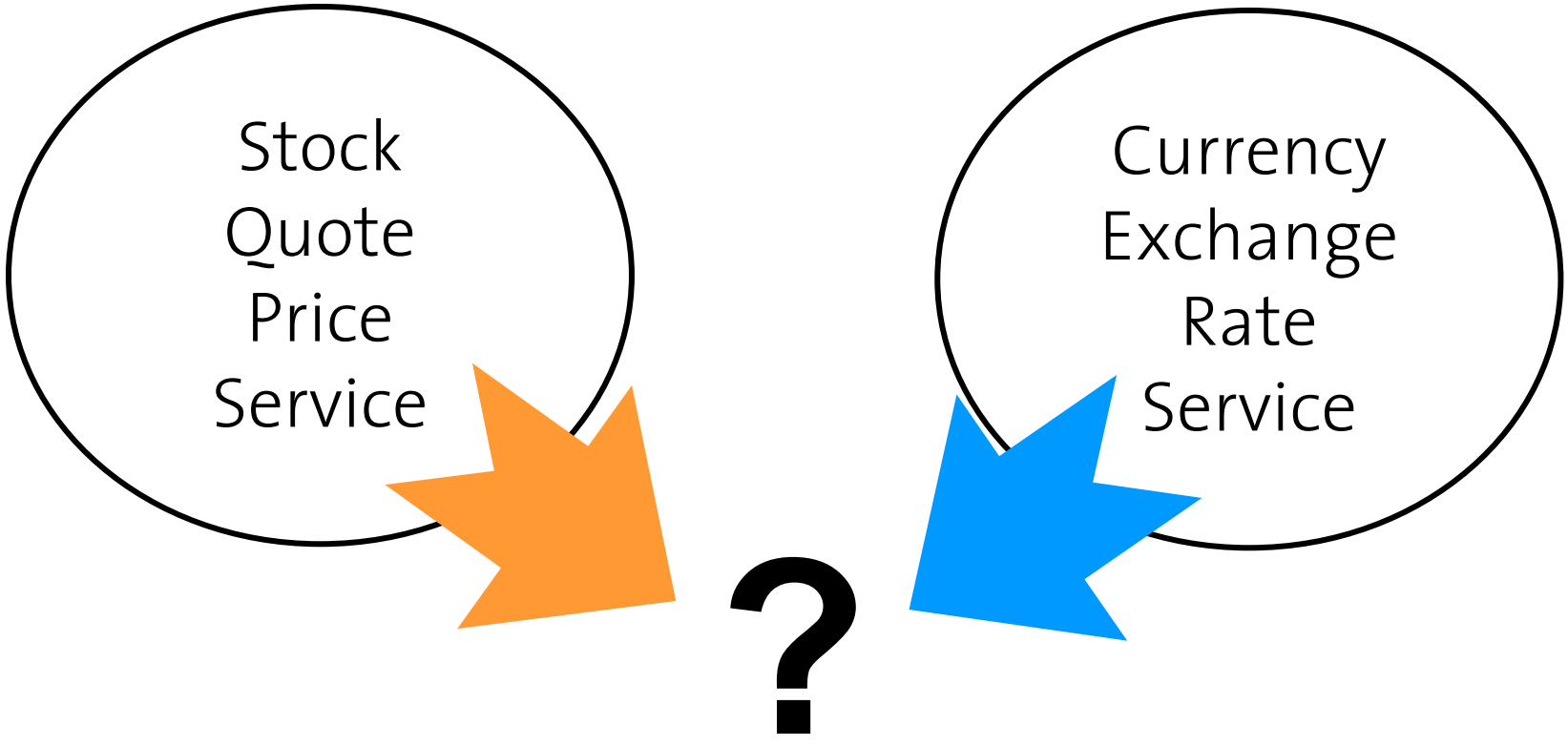
- Processes model service composition
- **Data flow** based visual language
- Explicit control flow (conditions, exceptions)
- Nesting and recursion with **sub-processes**
- **Iteration** and **recursion**
  - Split/Merge operators for list-based loops (parallel or sequential)
- Reflection (dynamic late binding)
- Visual adaptation of mismatching interfaces

Process Development Life Cycle

1. Select components from a **library**:
   - (Legacy?) UNIX Applications
   - Web Services (Import a WSDL description)
   - Reuse other existing processes
   - Java scripts (more...)
2. Build a Process by **visually** connecting the parameters of the selected components
3. Run and Monitor the Process' execution
4. Test, Debug, and Modify the Process
5. Publish the Process as Web Service

JOpera Visual Composition Example

- Stock Quote Currency Conversion



Web services are good, but not enough

Composition should be orthogonal from the component types

Benefits of Web services

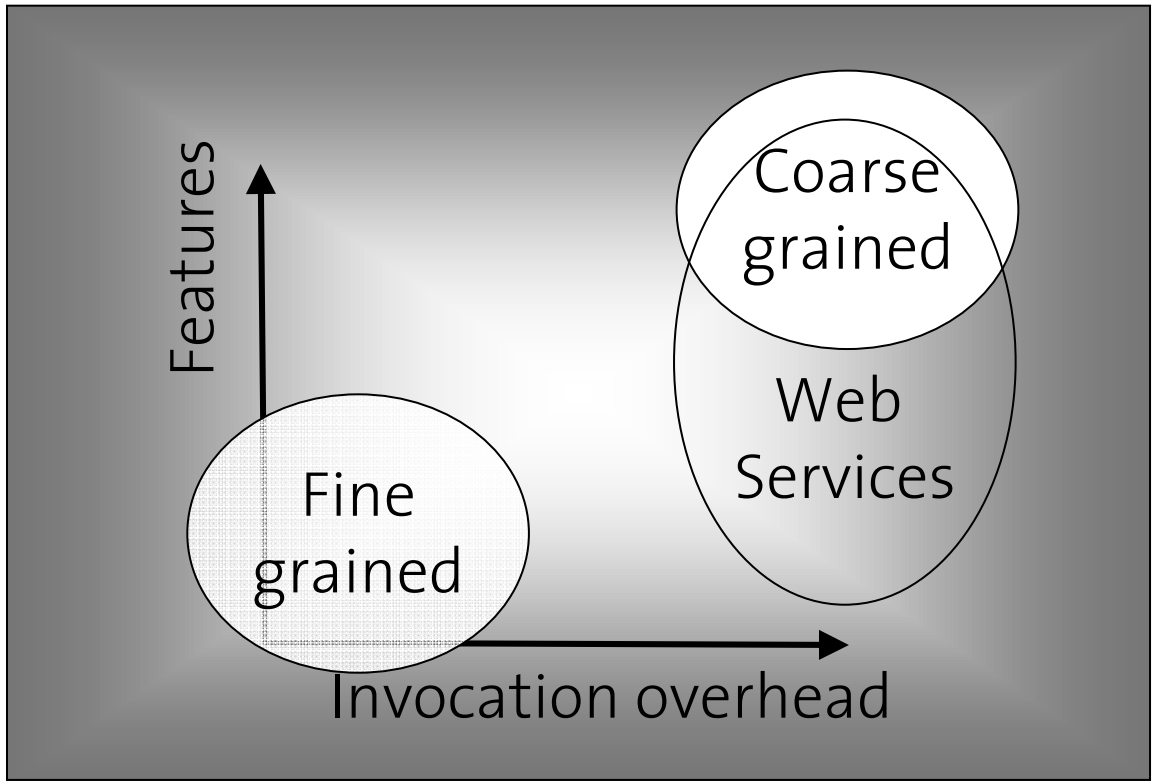
- W3C standards
- Programming Language neutral
- Platform independent
- Distributed software components can interoperate (.NET talks with Java)
- The infrastructure provides a lot of automatic support for basic tasks
- Foundation of the Semantic Web

Problems of composing ***only*** WS

- Web Services are coarse-grained
  - Try to multiply two numbers through a WS…
- Existing components must be wrapped as a Web Service
  - Wrapping imposes both a performance penalty and additional development costs
- Adaptation between mismatching services is also a service?

Service Type Classification

Web Services are one among many types of services that can be composed with JOpera



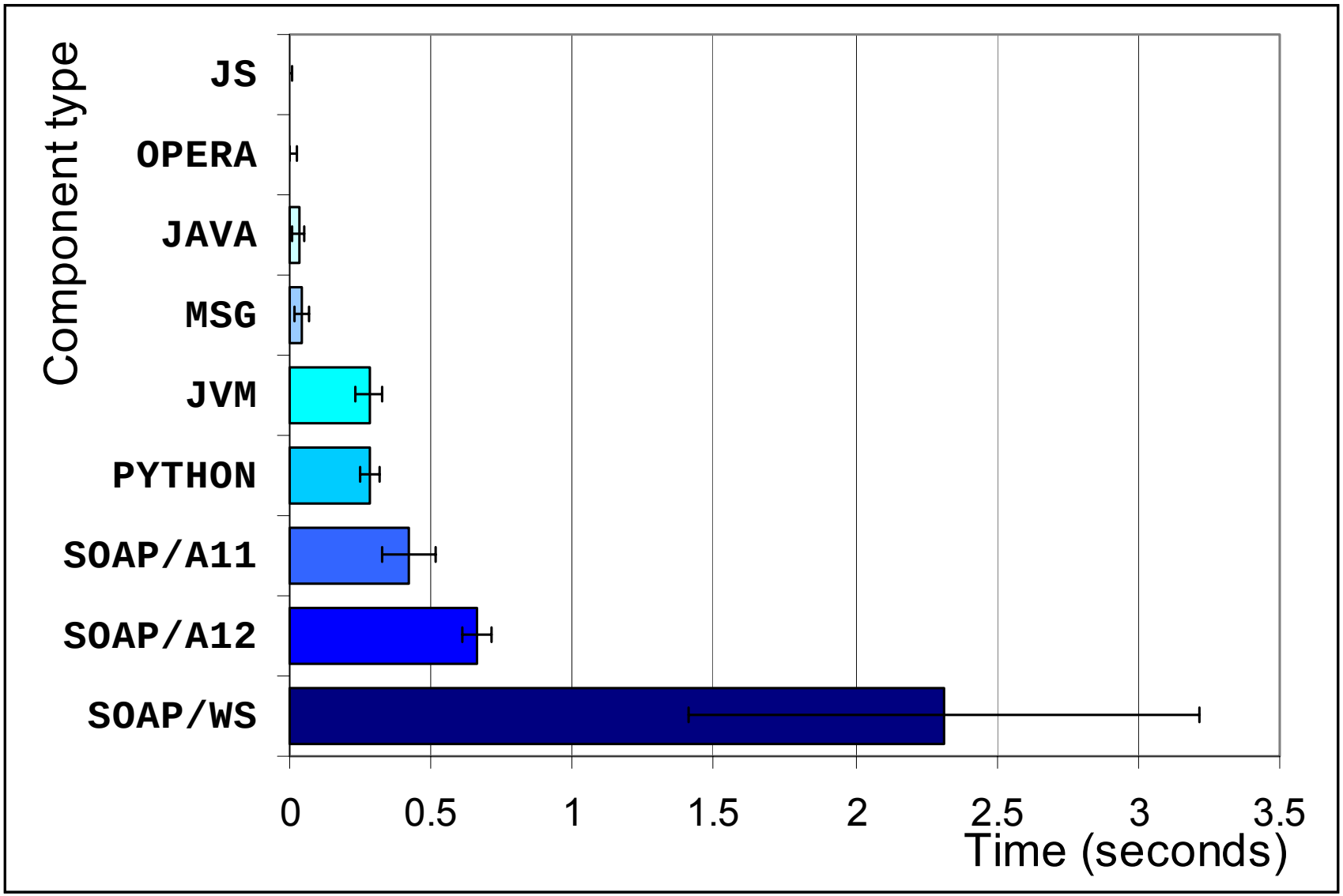
Types of Services

- Web Services (WSDL/SOAP)
- Java Classes (EJB, RMI) and scripts
- UNIX/Windows Shell Commands (Pipes)
- JOpera Processes
- XML Transformations (X-Path, XSLT)
- Asynchronous Messaging (JMS)
- Database Queries (PL/SQL)
- Grid Services and Cluster job submissions
- CORBA, DCOM, .NET, CICS
- Add your own...

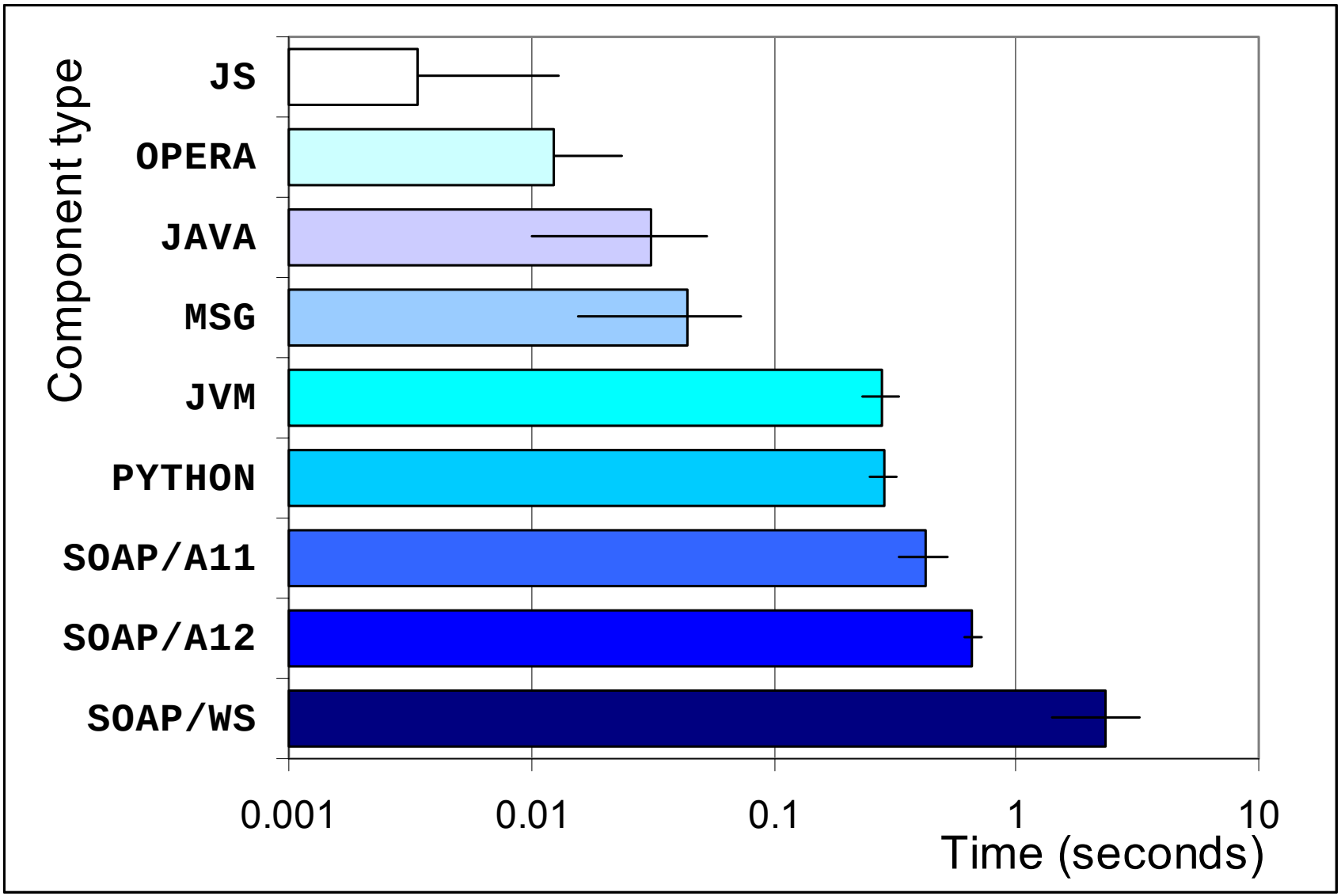
Why many service types?

- The user can choose (or add) the most appropriate type of component:
  - Access Protocols and Mechanisms, Functionality, Performance, Reliability, Security, Convenience, Ease of use
- The composition language is simpler
  - Many constructs can be moved from the language to the component model
- The composition language can be applied to many different domains

Service Invocation Overhead



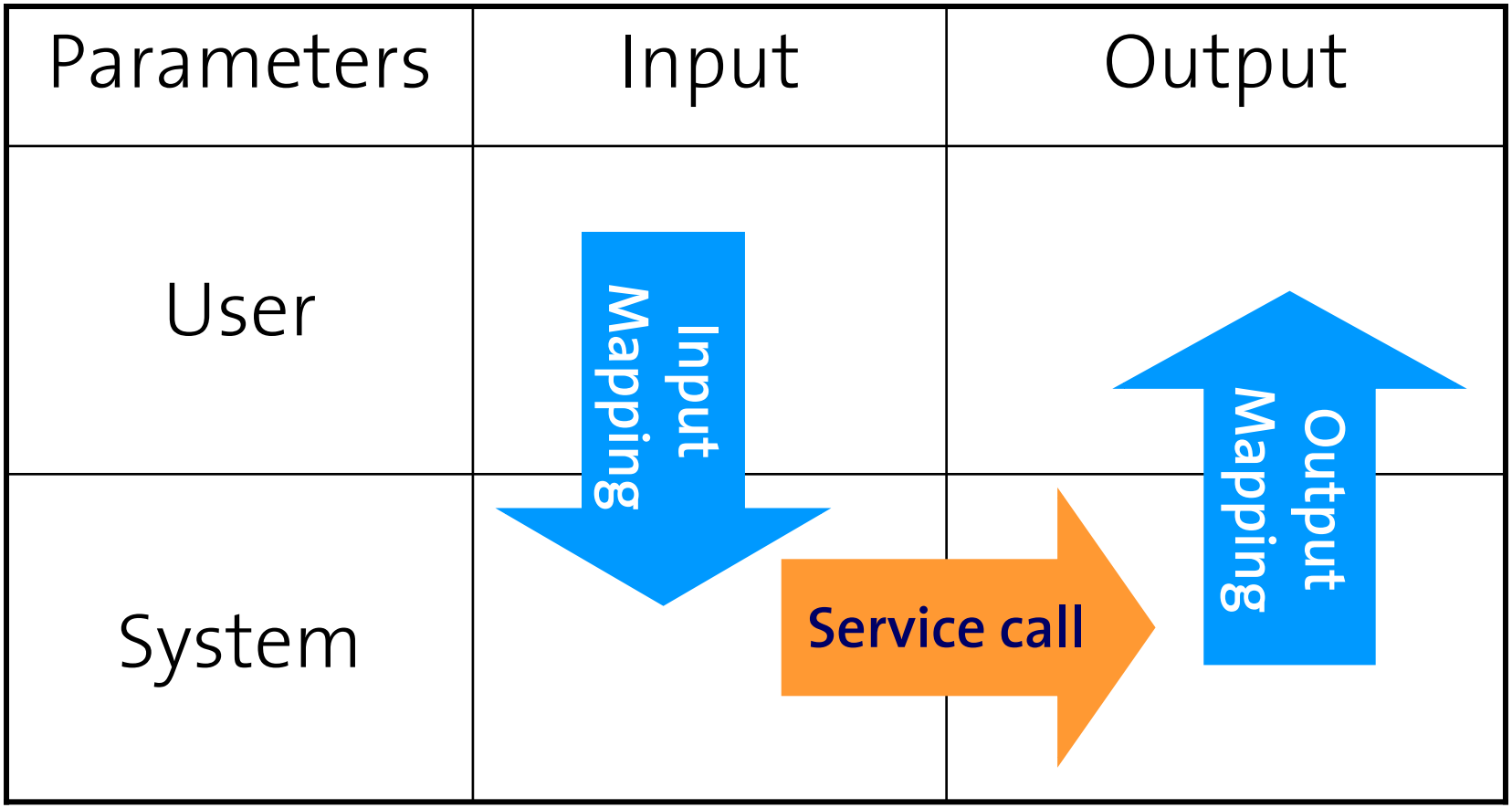
Service Invocation Overhead



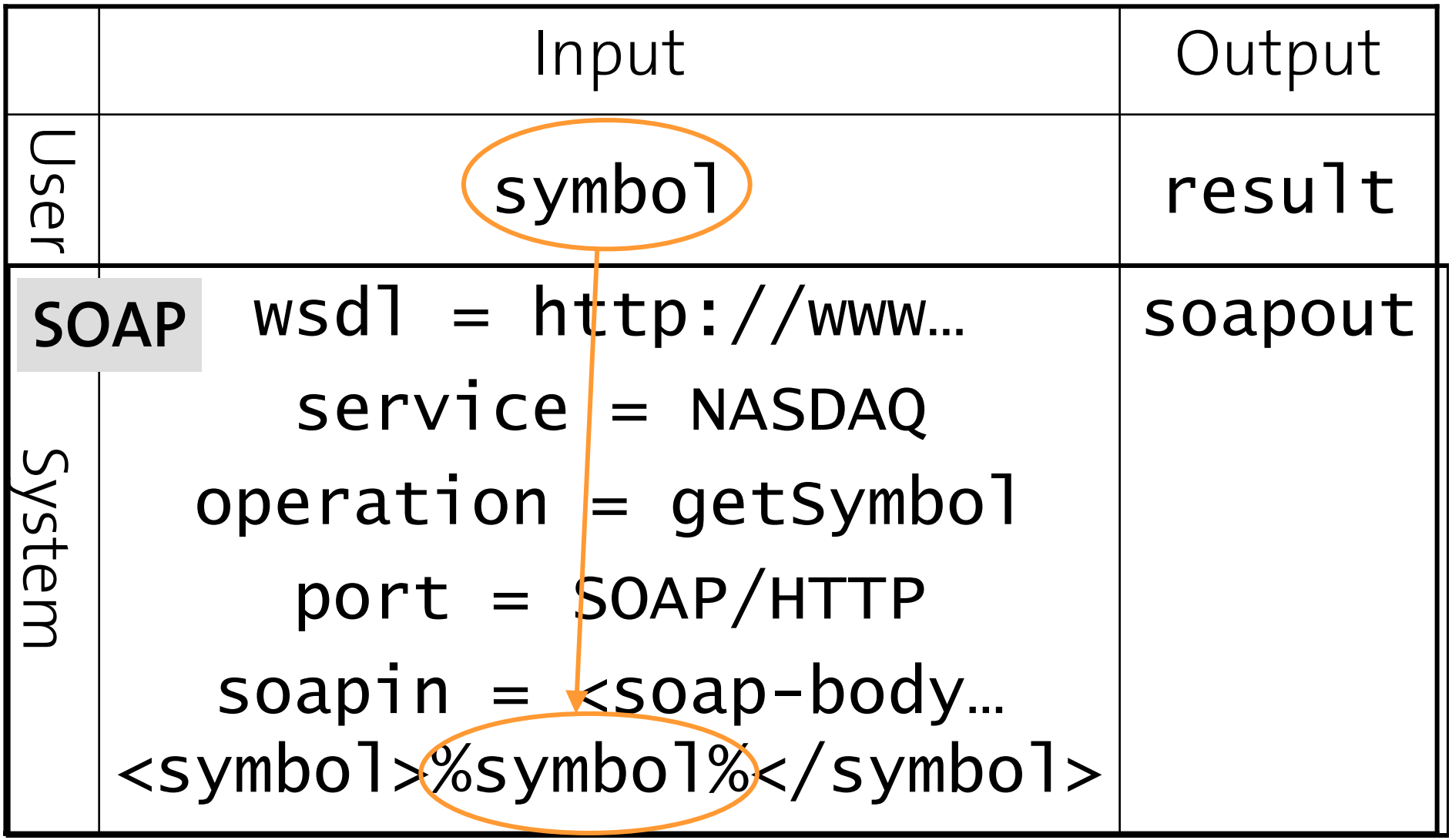
How to model types of services

- System Parameters
  - Describe and control the service invocation and its result
- Control flow
  - Start, Finish, Suspend, Resume, Kill
  - Immediate, synchronous, asynchronous
  - Scheduling among alternative providers
- Data flow mapping
- Failure detection

Data flow mapping

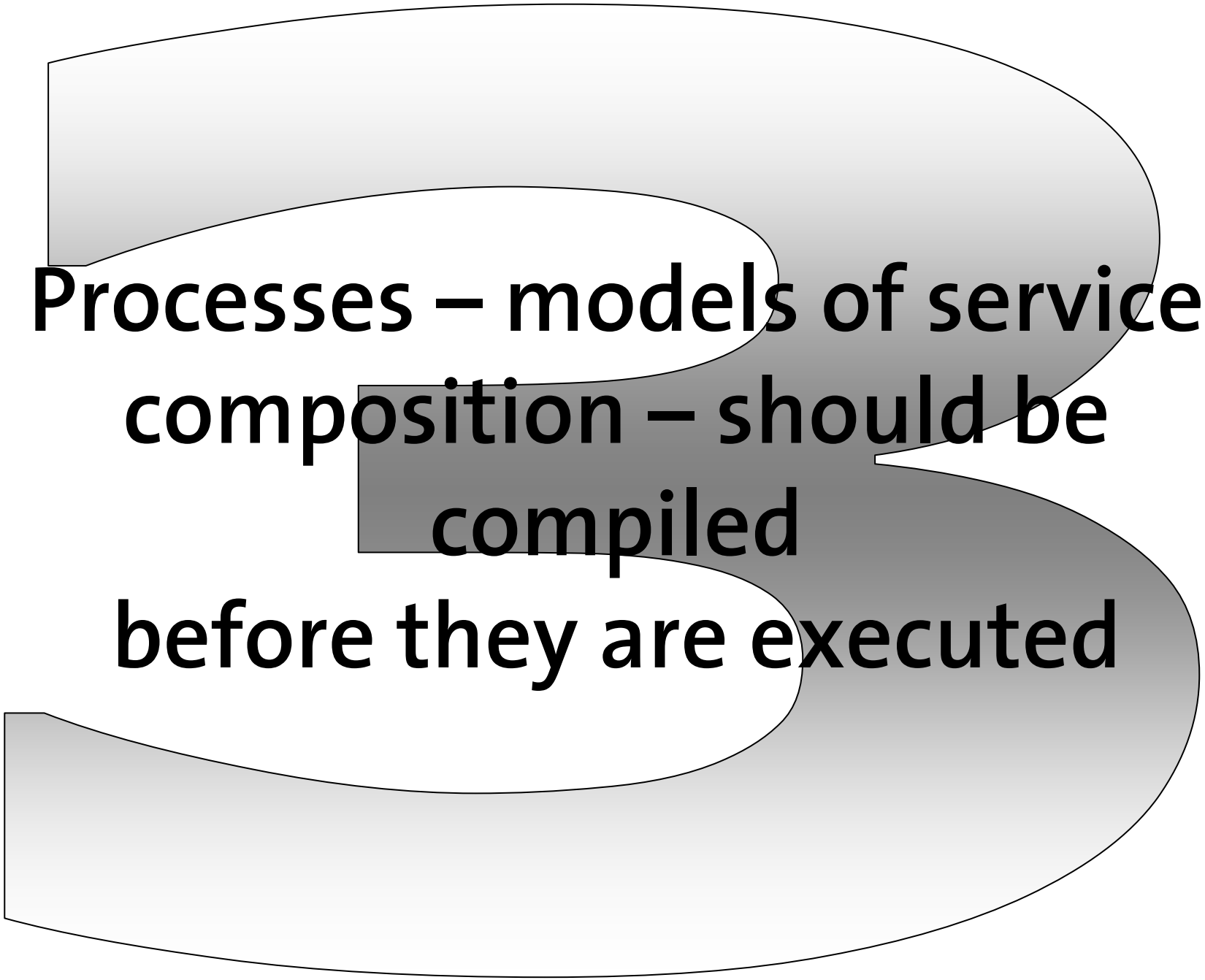


Example data flow mapping



Example data flow mapping

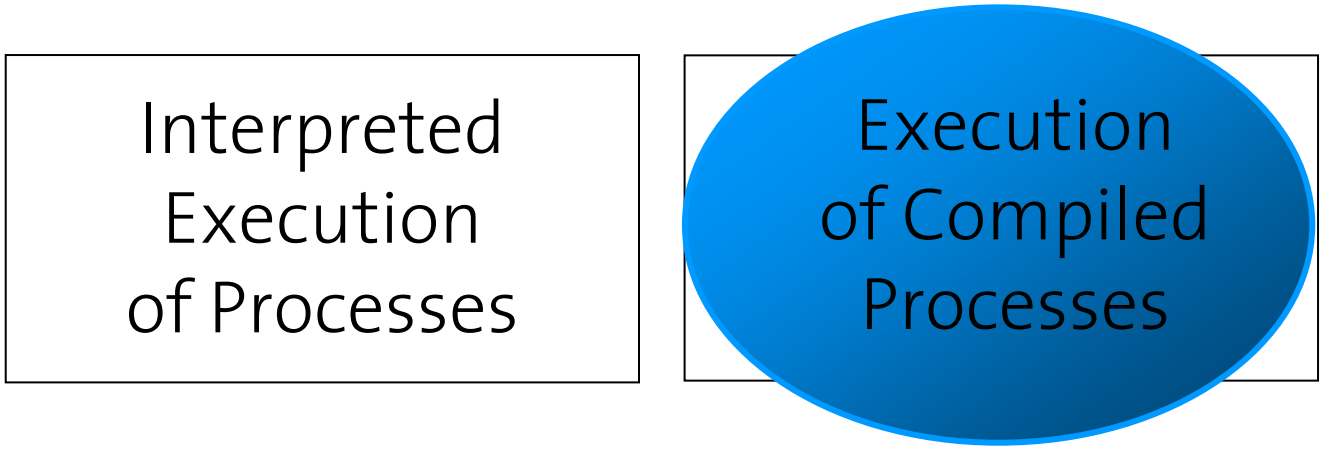
	Input	Output
User	`symbol`	`result`
System (SQL)	`connection = jdbc://oracle/db…` `query = SELECT PRICE FROM NASDAQ WHERE SYMBOL='%symbol%'`	`table`



Processes – models of service composition – should be compiled before they are executed

Process Compilation

Process based service composition tools will not gain a widespread acceptance if they cannot deliver a level of **performance** comparable to traditional programming languages.



Architecture

Jopera
Process Support for Web Services

1

GUI
Visual Process Development Environment

Web
HTML/SOAP Interface

Compiler
JVCL to Java

API
Process Monitoring Services

2

Kernel
Flexible Process Execution

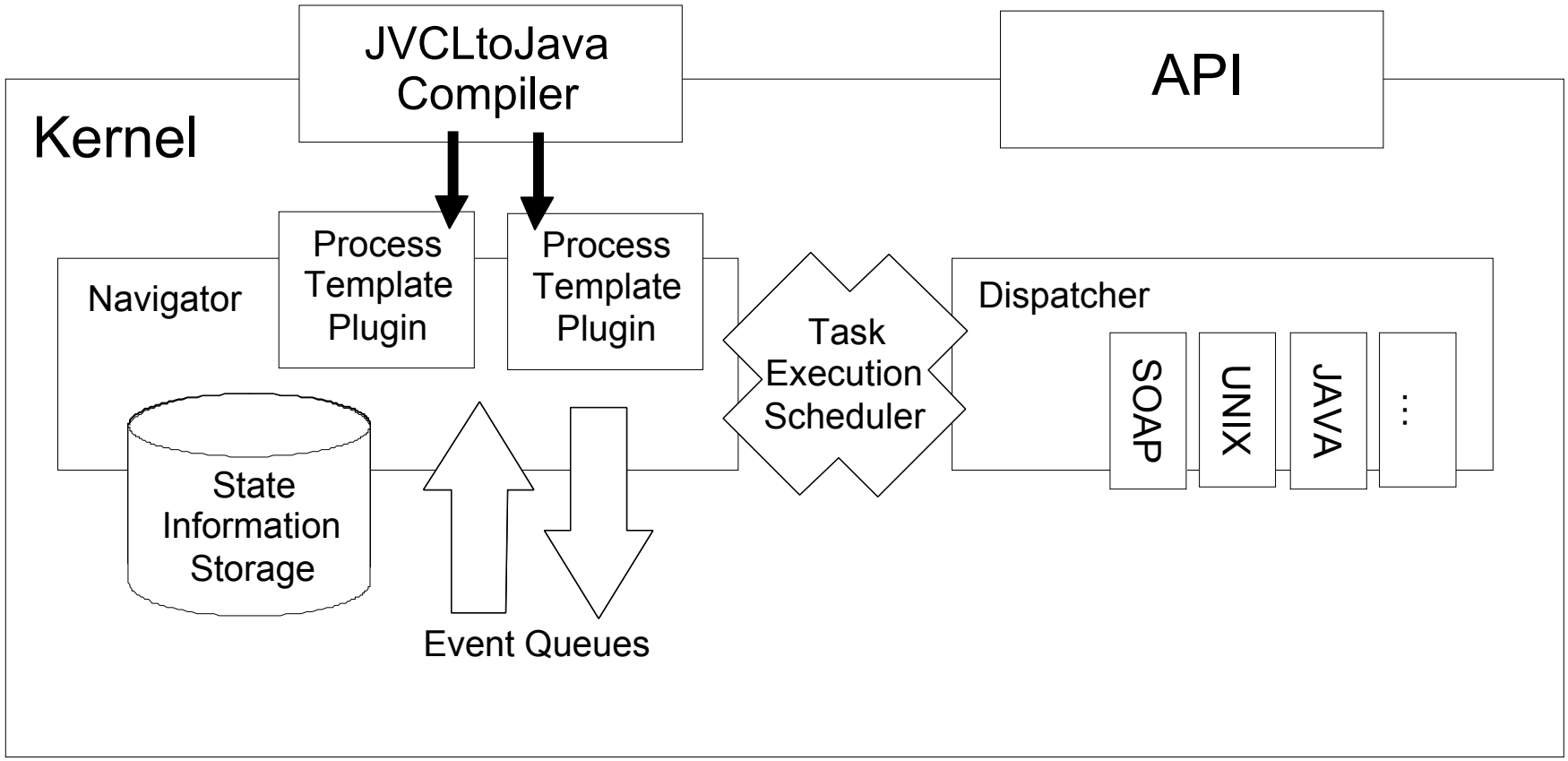
3

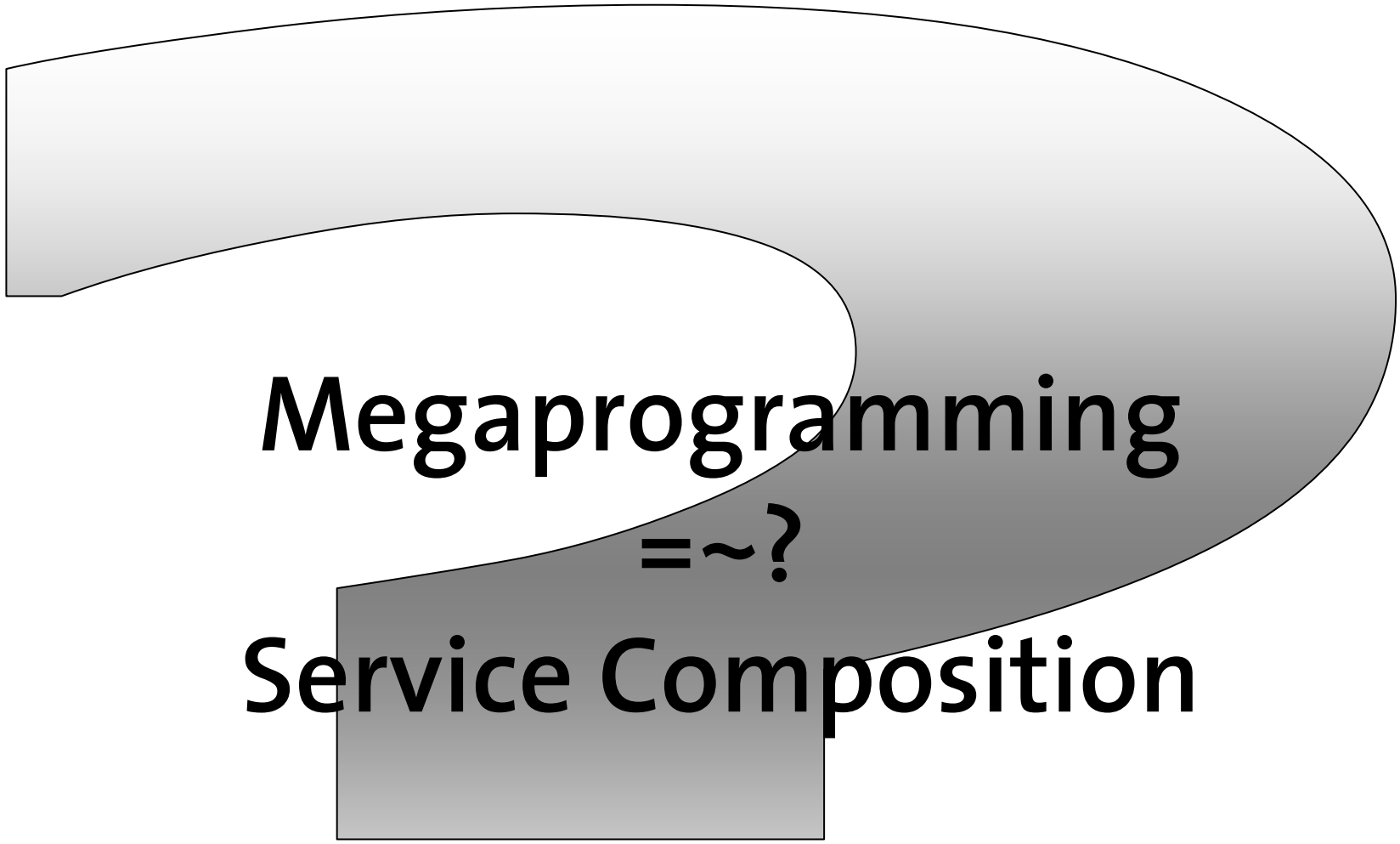
Service Invocation Adapters
Applications, Web Services, Components...

Flexibility

- Support for many types of services
  - Open component model
  - Adapters can be plugged in
- The same kernel architecture can be deployed in different configurations:
  - Adapt to required quality of service
    - Scalability
    - Reliability
  - Embedded or stand-alone process engine

Kernel Architecture (Centralized)




Megaprogramming
=~?
Service Composition

Conclusions

- (Web) services are composed using a **visual** programming language
- Web services are good, but **not enough**: composition is done at the service interface level, across component models
- Processes modeling service composition are **compiled** before they are executed

Available today

http://www.iks.ethz.ch/jopera

References

- **http://www.iks.ethz.ch/jopera**

- G. Alonso, W. Bausch, C. Pautasso, A. Kahn, M. Hallett. **Dependable Computing in Virtual Laboratories** In: Proc. of the 17th International Conference on Data Engineering (ICDE2001), Heidelberg, Germany, April 2001.
- W. Bausch, C. Pautasso, R. Schaeppi, G. Alonso. **BioOpera: Cluster-Aware Computing** In: Proc. of the 4th IEEE International Conference on Cluster Computing, Chicago, USA, September 2002.
- W. Bausch, C. Pautasso, G. Alonso **Programming for Dependability in a service-Based Grid** In: Proc. of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid), Tokyo, Japan, May 2003.
- C. Pautasso, G. 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.