Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
http://www.ethz.ch

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

Information and Communication Systems Research Group
http://www.iks.inf.ethz.ch

# JOpera: A Flexible System for Visual Service Composition

Cesare Pautasso
pautasso@inf.ethz.ch
Department of Computer Science
Swiss Federal Institute of Technology (ETHZ)
Zurich, Switzerland
http://www.iks.ethz.ch/jopera

JOpera is a visual tool
for modeling and executing
distributed processes
composed out of reusable services.

## Modeling Service Oriented Architectures

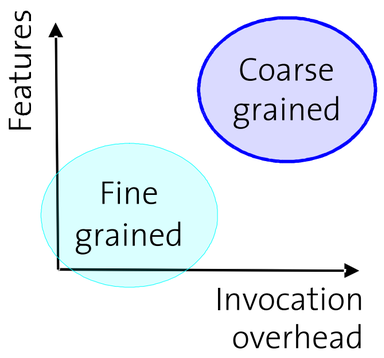
Composition and Components should be modeled orthogonally

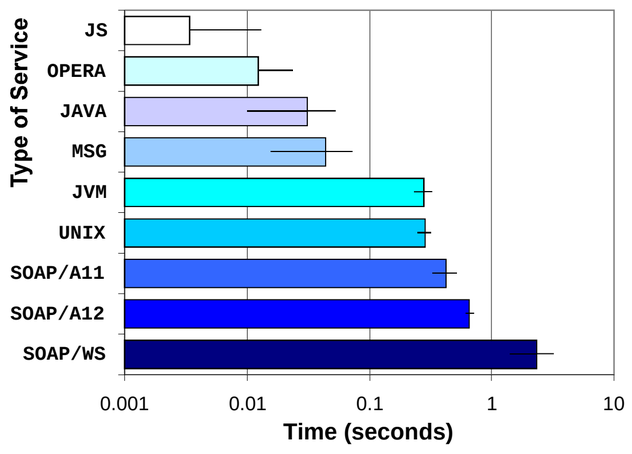
## Open Service Meta-Model

“A service is an instantiated configured system that is run by a providing organization. That is, it is fully grounded. Ultimately, it includes the power supply to the server machines as well as the organization that somehow manages to pay the power bill.”

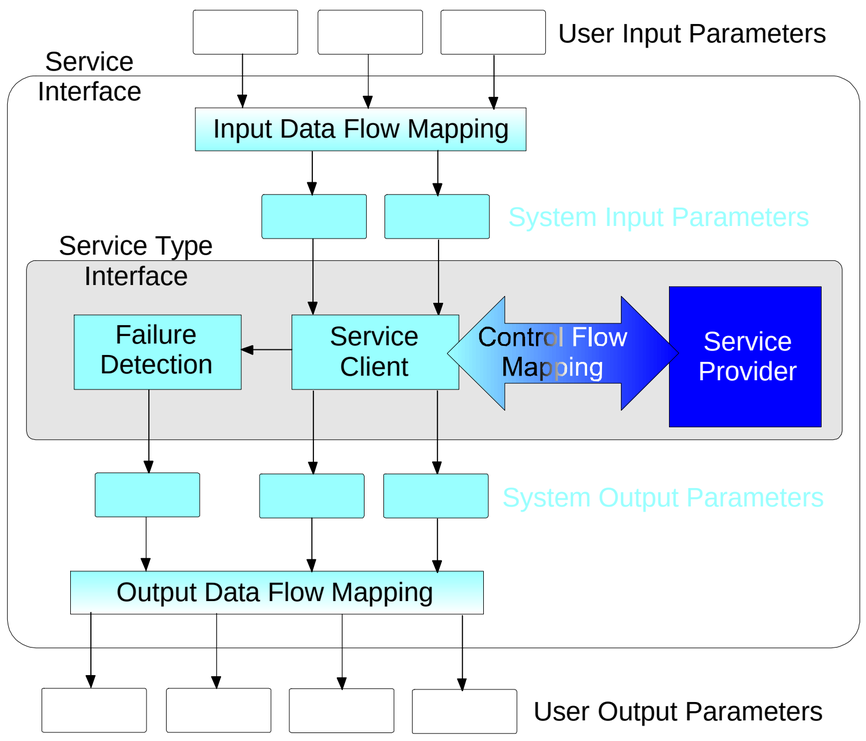
(Szyperski, ICSE 2003)

Web Services are one Kind of Service




- Composition does not have to be constrained
  to a particular type of service
  (like coarse grained Web services)
- JOpera gives freedom of choice
  to pick the most appropriate kind of services in terms of performance, reliability, security and convenience.




To be invoked a service interface is associated with a type.
A **service type** defines:

- The **system parameters** describing the information required to interact with a service using a certain protocol
- How to **transfer control** (synchronous or asynchronous)
- How to **map data** flow (input and output)
- A **failure detection** strategy

## Visual Composition Language

Composition (or glue) languages model how components (or services) are connected within an application’s architecture

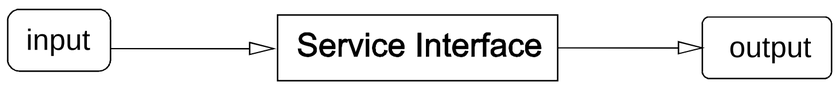
JOpera provides a visual service composition language

JOpera uses a simple visual syntax based on graphs to model the interaction between different services as a Process.
The structure of a Process is defined by drawing:

- the **Data Flow** between parameters of service interfaces
- the **Control Flow** defining the partial order of service invocations, branches and exception handling behavior

Main language features

- **Nesting:** composite services are composable
- **Recursion:** a composite service can invoke itself
- **Iteration:** list-based loops and control flow loops
- **Reflection:** model the interaction with the environment
- **Dynamic late binding** of interfaces to implementations
- **Interface Adaptation** of data and interaction style mismatches can be solved with the same visual syntax used to compose the services

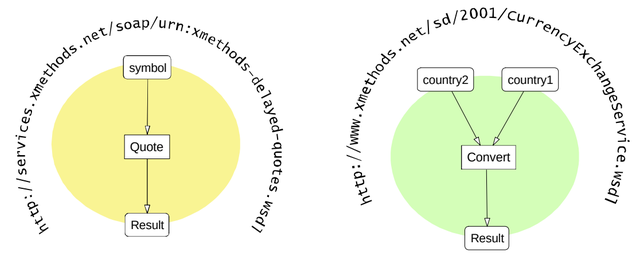



Services are composed at the level of their interfaces.

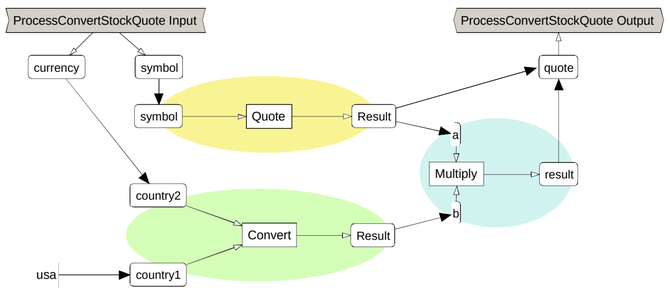
- Very little assumptions are made about the mechanisms that are used to access the service implementations.
- The JOpera visual composition language is independent of the mechanisms and protocols involved.

## Rapid Service Composition

**1** Import theWSDL interface of the Web services




**2** Connect the data parameters of the services
Add integration and adaptation logic




To be invoked a service interface is associated with a type.
A **service type** defines:

- The **system parameters** describing the information required to interact with a service using a certain protocol
- How to **transfer control** (synchronous or asynchronous)
- How to **map data** flow (input and output)
- A **failure detection** strategy

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
http://www.ethz.ch

Jopera
Process Support for Web Services
http://www.iks.ethz.ch/jopera

IKS Information and Communication Systems Research Group
http://www.iks.inf.ethz.ch

# JOpera: A Flexible System for Visual Service Composition

JOpera is a flexible, open and scalable platform for efficiently executing processes

Cesare Pautasso
pautasso@inf.ethz.ch
Department of Computer Science
Swiss Federal Institute of Technology (ETHZ)
Zurich, Switzerland
http://www.iks.ethz.ch/jopera

## Scalable Execution of the Composition

Compilation to a flexible runtime platform ensures efficiency of the execution

### Compiler vs. Interpreter

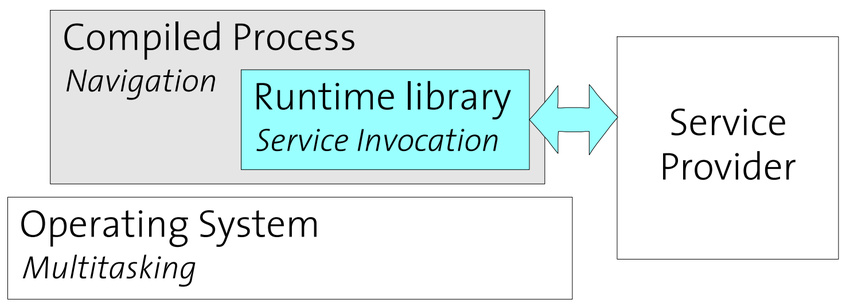
Process-based service composition tools will not gain widespread acceptance if they cannot deliver a level of performances comparable to traditional programming languages.

In JOpera we apply compilation techniques to achieve efficient execution of the visual language. However, compiling to Java code is not enough because of the following issues:

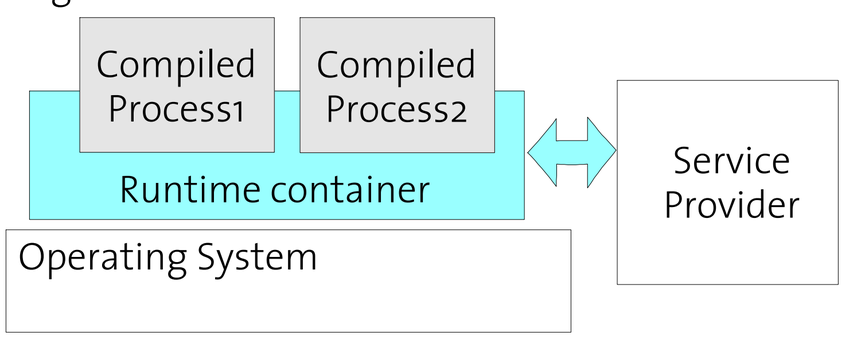
- Performance Minimize overhead
- Persistence Recoverable execution
- Scalability High number of concurrent executions
- Portability Platform independence
- Flexibility Dynamic late binding
- Monitoring User can track the progress of the execution

Alternatives for Compiling Processes

1. Stand-Alone Program




2. Plug-in

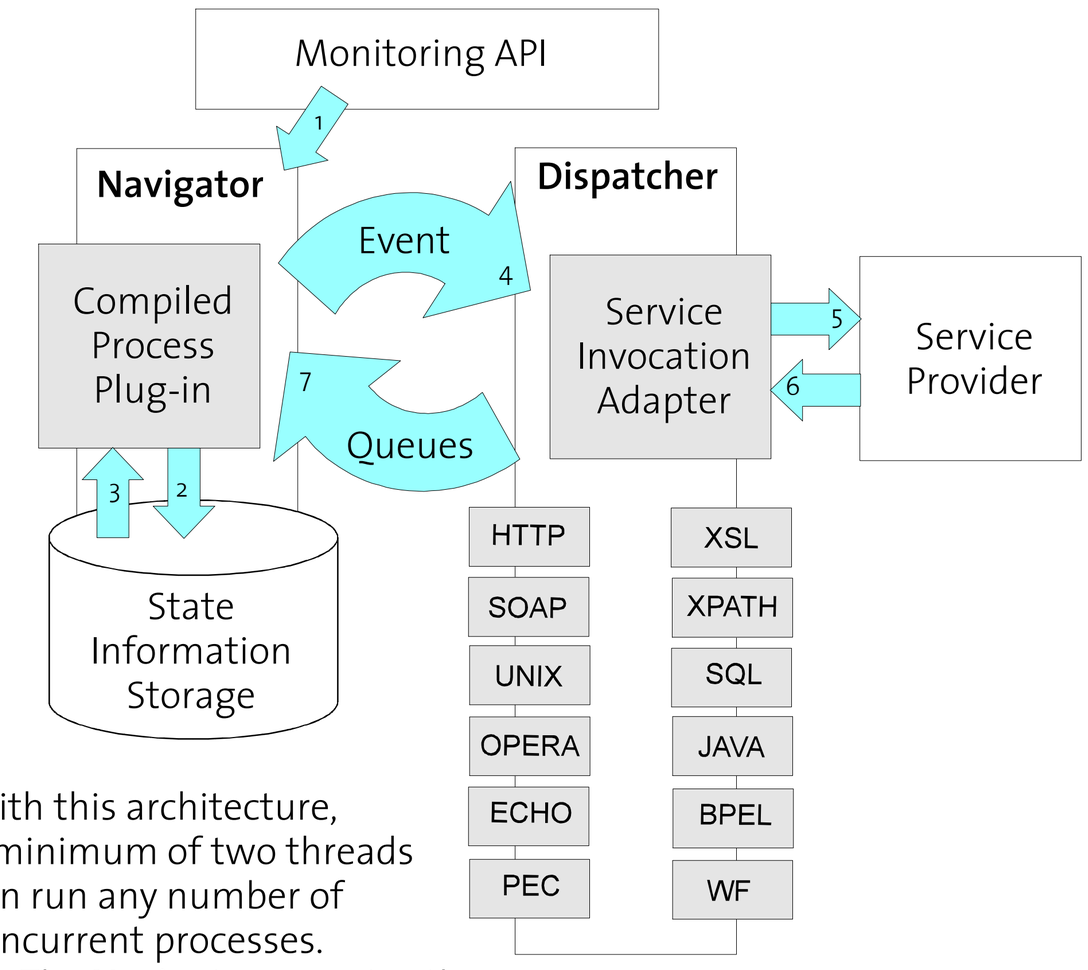



### Flexible Kernel Architecture

The JOpera Kernel is a flexible container of compiled processes. Flexibility is an important aspect:

- to interact with an open set of service types
- to offer different levels of performance in terms of reliability and scalability
- to be deployed embedded into other systems (application servers, development environments) or also run stand-alone.

Main Process Execution Loop

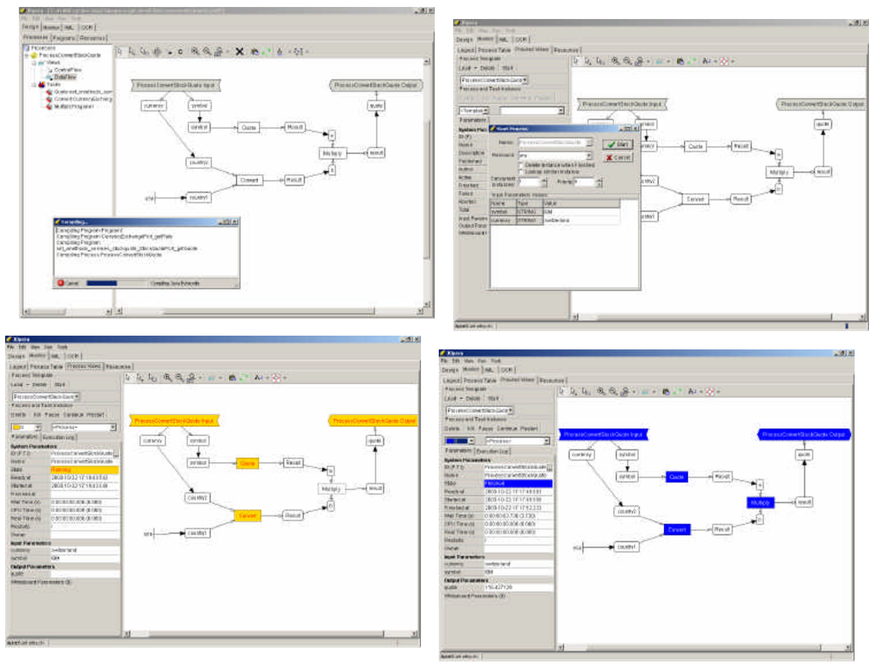



With this architecture, a minimum of two threads can run any number of concurrent processes.

- The **Navigator** executes the processes
- The **Dispatcher** invokes the services

They can be replicated across a cluster

**3** Run, debug and monitor the execution using the same visual syntax



## System Requirements

JOpera v1.69
Stable release, free download
http://www.iks.ethz.ch/jopera/download
The Visual Composition Environment runs on Windows 2k/xp
The Compiler and Runtime Kernel require the Java JDK 1.4

JOpera for Eclipse
coming soon
We are currently porting the Visual Composition Environment to Eclipse 3.0 with the GEF plugin
The Compiler and the Kernel can be plugged into Eclipse too!