A SOA PROGRAMMING MODEL BASED ON AGENTS & ARTIFACTS

Alessandro Ricci
a.ricci@unibo.it
DEIS, Università di Bologna
Sede di Cesena

AGENDA

- Service-Oriented Architectures (SOA) & Web Services
- State-of-the-art programming models for SOA: the industry point of view
  - Component-based approaches (SCA, INDIGO, JBI)
- simpA-WS: An agent-oriented programming model for SOA
  - A&A applied to SOA
- simpA-WS: current technology

SERVICE-ORIENTED ARCHITECTURE

- Reference model in the state-of-the-arts for conceiving / designing / engineering complex distributed inter-organization Internet-based applications
  - based on Web Services as open standards
- [SOA-slides]

SOA PROGRAMMING MODELS: STATE OF THE ART FROM THE INDUSTRY

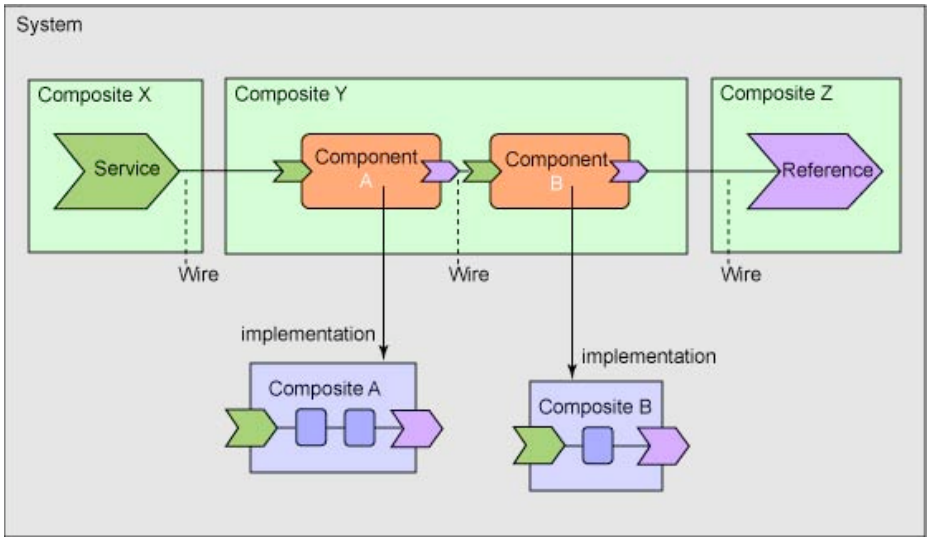
- Main proposals
  - Service Component Architecture
    - IBM, BEA, Oracle, IONA, SAP,...
  - Web-Service Enhancements (WSE, ex INDIGO)
    - Microsoft
  - Java Business Integration (JBI)
    - Java & JSR Community process
- Component-oriented / object-oriented level of abstraction
  - common point of the proposals

SERVICE-COMPONENT ARCHITECTURE (SCA)

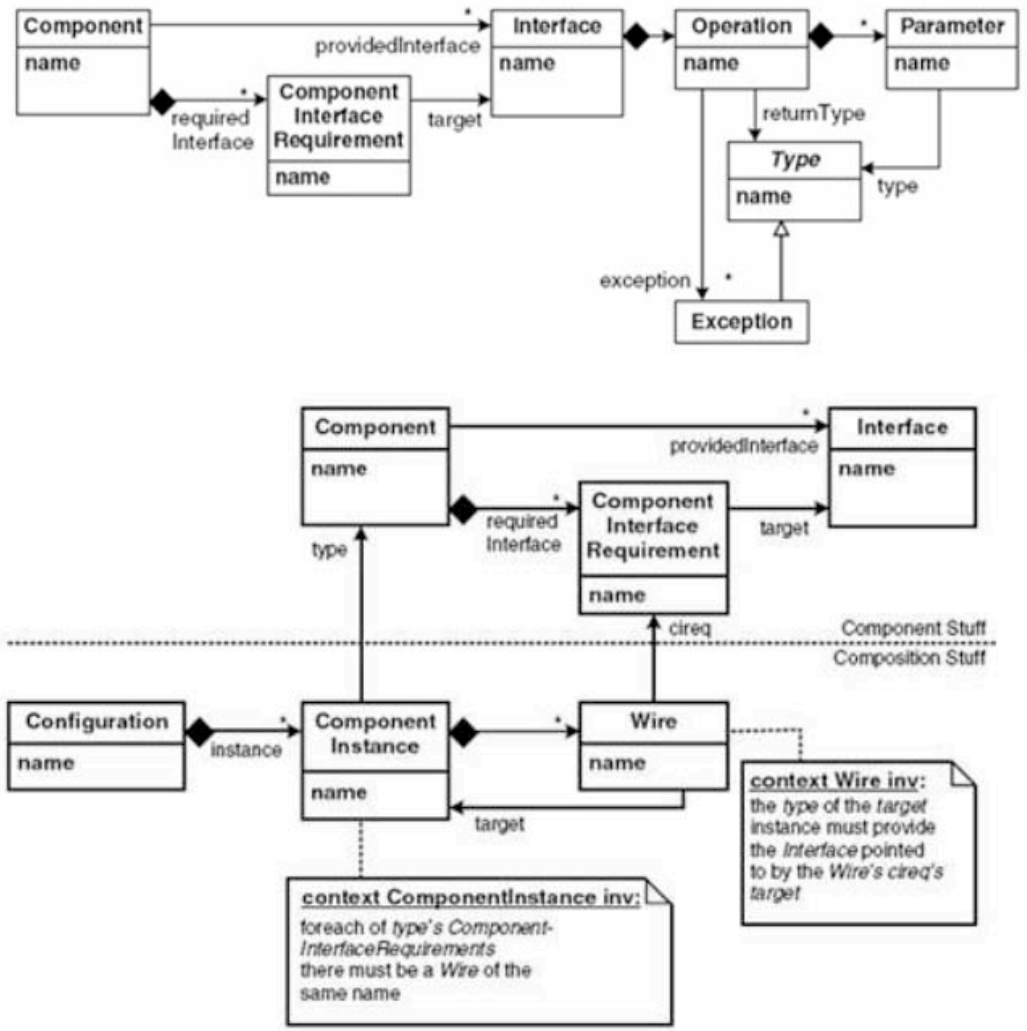
- http://www-128.ibm.com/developerworks/library/specification/ws-sca/
- "Service Component Architecture (SCA) is a set of specifications which describe a model for building applications and systems using a Service-Oriented Architecture.
  - SCA extends and complements prior approaches to implementing services, and SCA builds on open standards such as Web services.
  - SCA encourages an SOA organization of business application code based on components that implement business logic, which offer their capabilities through service-oriented interfaces and which consume functions offered by other components through service-oriented interfaces, called service references.

COMPONENTS AND WIRES

- SCA divides up the steps in building a service-oriented application into two major parts:
  - The implementation of service components which provide services and consume other services.
  - The assembly of sets of components to build business applications, through the wiring of service references to services.

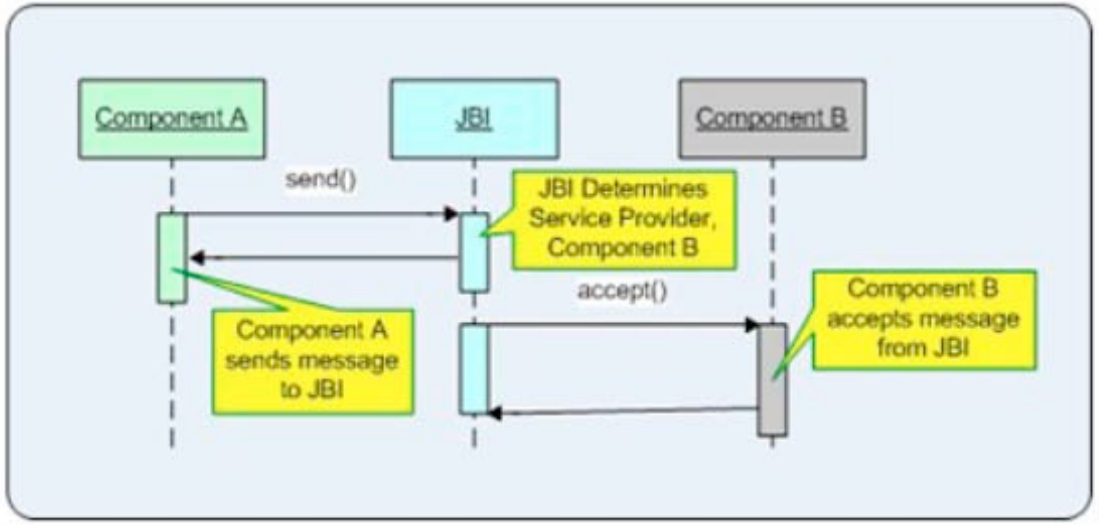


SCA COMPLETE META-MODEL



JBI

- "JBI is a pluggable architecture consisting of a container and plug-ins. The container hosts plug-in components that communicate via message routers. Architecturally, components interact via an abstract service model-a messaging model that resides at a level of abstraction above any particular protocol or message-encoding format."



Microsoft WSE

- "Web Services Enhancements for Microsoft .NET (WSE) is a supported add-on to Microsoft Visual Studio .NET and the Microsoft .NET Framework providing developers the latest advanced Web services capabilities to keep pace with the evolving Web services protocol specifications."

SO Entities	OO Entities	Annotations
Service contract	Interface	Annotate interface with *[ServiceContract]*
Service operation	Method	Annotate interface method with *[OperationContract]*
Implementation class	Class	Annotate class with *[ServiceBehavior]* and derive it from service contract interface.
Implementation method	Method	Annotate method with *[OperationBehavior]*
Data contract	Class	Annotate class with *[DataContract]* and its members with *[DataMember]*

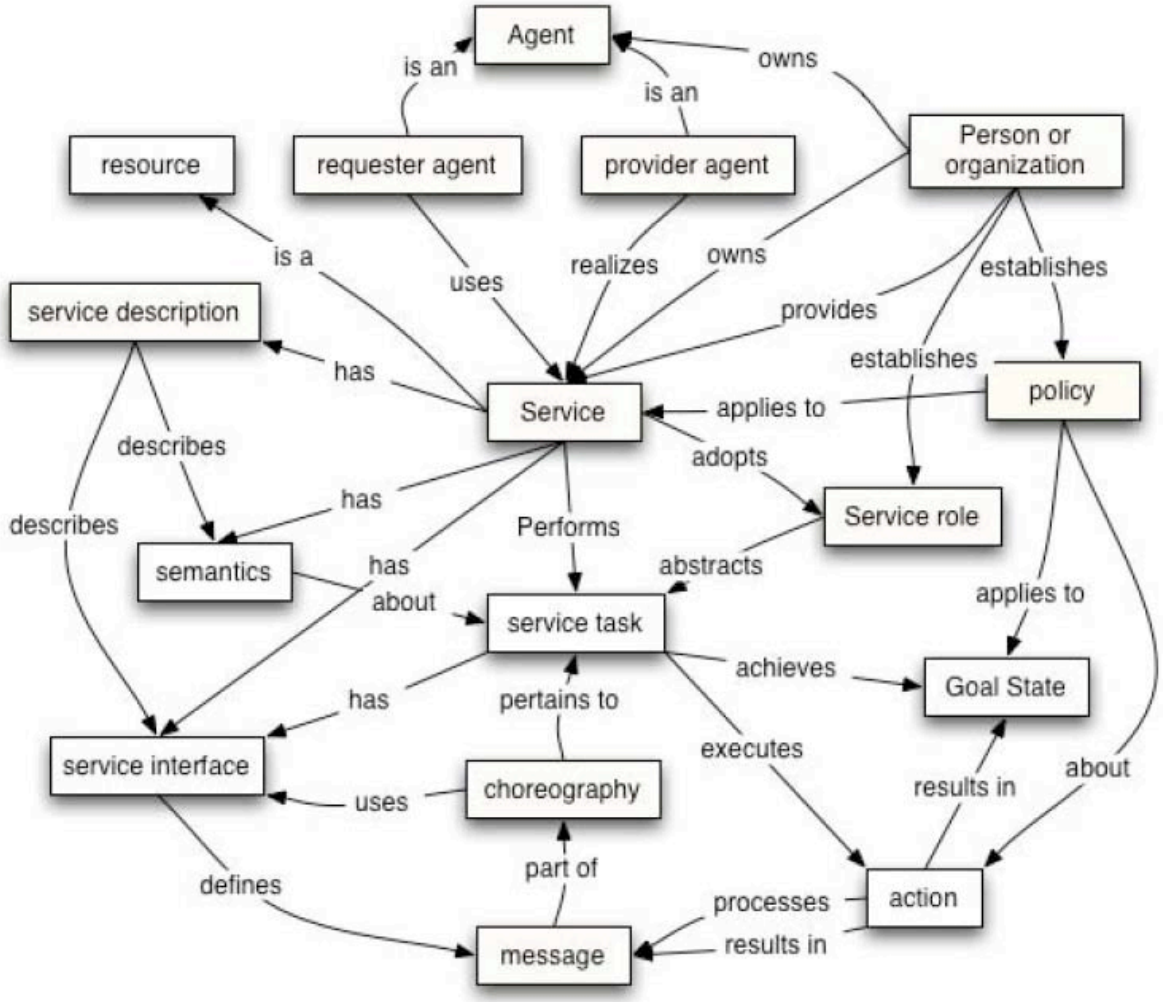
COMPONENT-BASED PERSPECTIVE: STRENGTH & LIMITS

- Well-known strength...
  - reuse, encapsulation, ...
  - dynamic composability
- ...but also limits
  - no explicit support for modelling *activities* and *processes*
    - reuse, encapsulation,composability limited only to passive components of the system
  - no explicit support for managing interaction of multiple cooperating activities

A PROGRAMMING MODEL BASED ON AGENTS & ARTIFACTS

- Designing and programming services & client applications in terms of agents and artifacts
  - a service / user application as one or multiple workspaces composed by set of interacting agents cooperatively using some kinds of artifacts
- Task / activity oriented design & programming
  - using agents to model / encapsulate / program individual business activities & their control
  - using artifacts to model / encapsulate / program resources and tools used by business activities

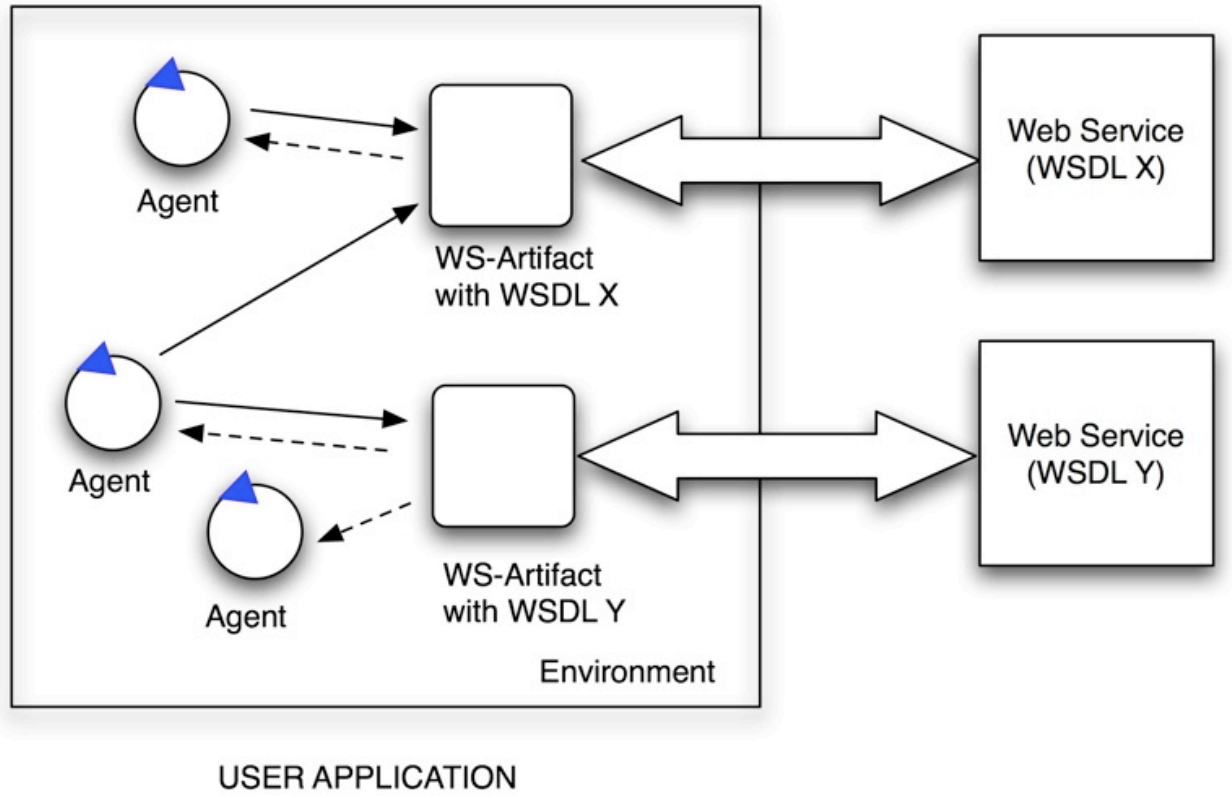
AGENTS AND WS ARCHITECTURE ACCORDING TO W3C



simpA-WS

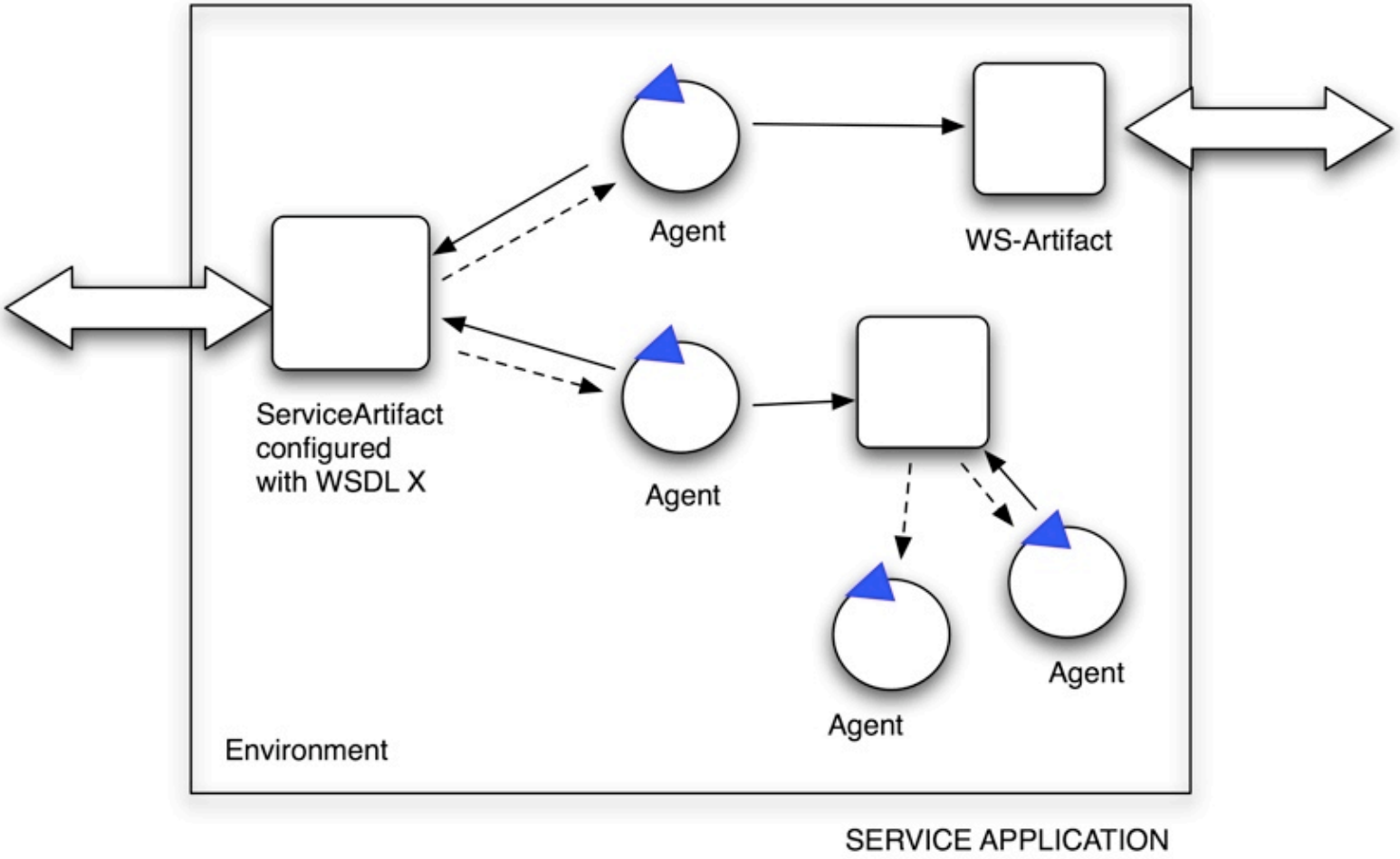
- Library / environment to develop SOA / WS application based using the simpA framework
  - Programming model based on agents and artifacts
  - simpA project: http://www.alice.unibo.it/projects/simpa
- Programming...
  - user applications
    - in terms of one or multiple simpA agents that can access and use Web Services through suitable artifacts
  - services
    - in terms of one or multiple simpA agents that implement a Web Service, using suitable artifacts to interact with service users
- Open-source project / technology
  - http://www.alice.unibo.it/projects/simpa-ws

simpA-WS:
PROGRAMMING USER APPLICATIONS

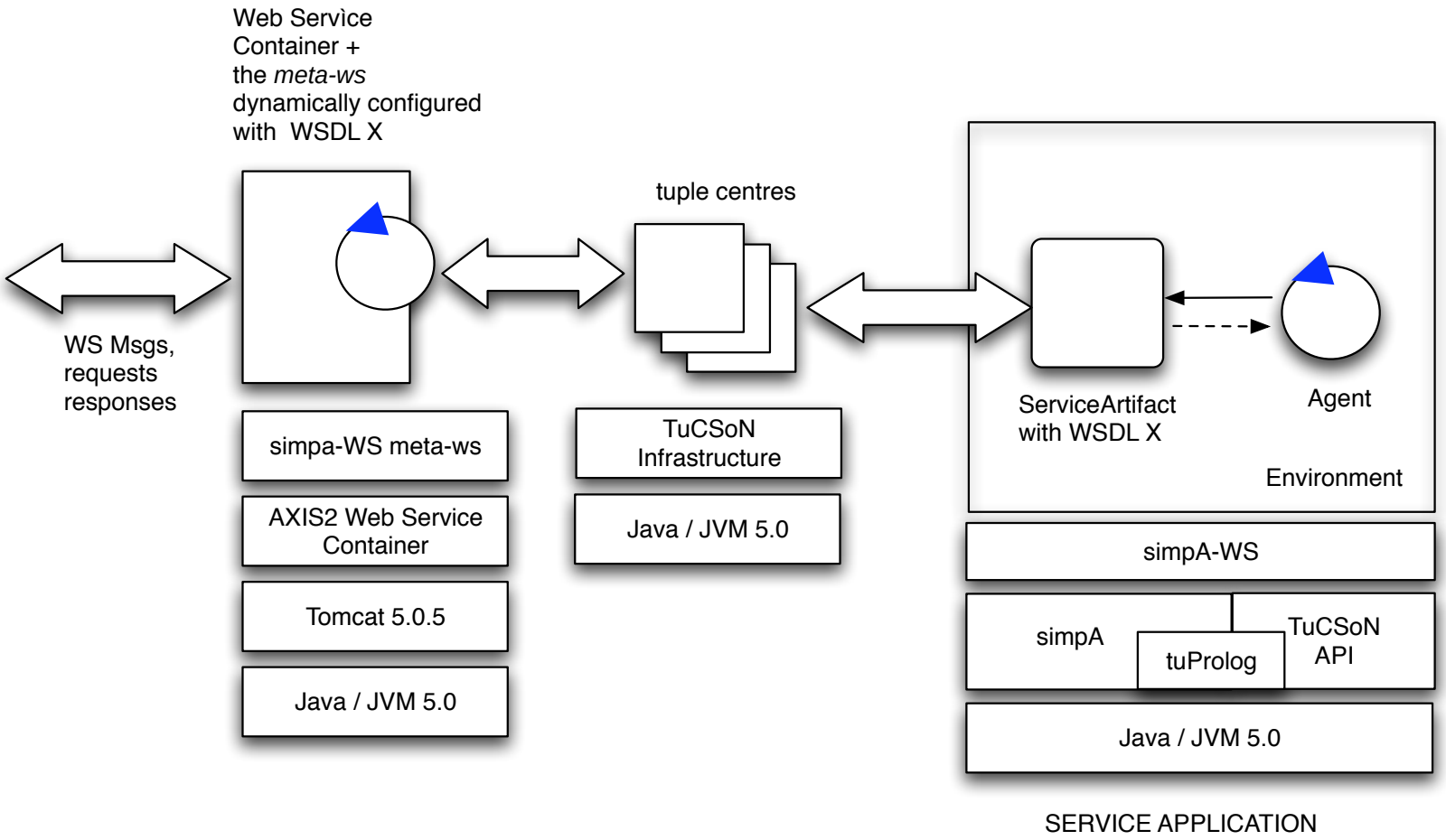


USER APPLICATION

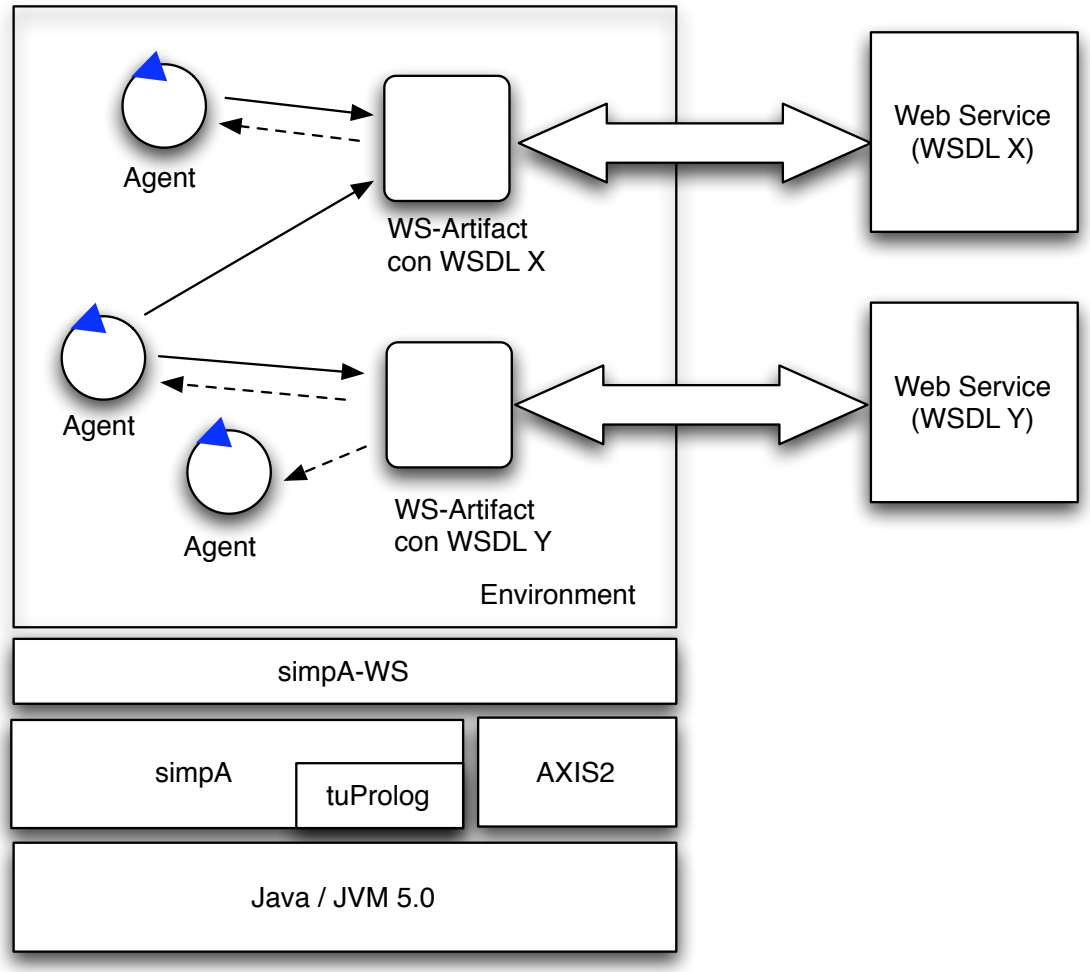
simpA-WS:
PROGRAMMING SERVICE APPLICATIONS



simpA-WS ARCHITECTURE: SERVICE SIDE



simpA-WS ARCHITECTURE: USER/CLIENT SIDE



WS-USER APPLICATION

WS-ARTIFACT USE: SIMPLE CODE SNIPPETS

- [from simpA-WS manual & basic examples]

```
ArtifactId aid = getArtifactId("my-ws");

String arg0 =
  "<echoStructParam xmlns=\"http://examples.simpaws.alice.unibo.it/xsd\">"+
  "<varFloat>3.14</varFloat>"+
  "<varInt>303</varInt>"+
 "<varString>hello world</varString>"+
 "</echoStructParam>";

WSOp op = new WSOp(aId,"echoStruct",arg0);
SensorId sid = linkSensor(new DefaultSensor());
execOp(op,sid);
```

WS-ARTIFACT USE: SIMPLE CODE SNIPPETS

- exploiting data-binding (xmlbeans)

```
import it.unibo.alice.simpaws.examples.*;

EchoStructParamDocument doc =
  EchoStructParamDocument.Factory.newInstance();
SOAPStruct el = doc.addNewEchoStructParam();
el.setVarInt(101);
el.setVarFloat(3.1415f);
el.setVarString("hello world again");

op = new WSOp(aId,"echoStruct",doc.toString());
execOp(op,sid);
```

WS-ARTIFACT USE: SIMPLE CODE SNIPPETS

```
...
WSOp op = new WSOp(aId,"echoStruct",arg0);
SensorId sid = linkSensor(new DefaultSenso
execOp(op,sid);

Perception p = sense(sid,5000);
if (p!=null){
  WSEvent ev = (WSEvent)p.getEvent();

  if (ev.isResponse()){
    String response = ev.getMessage();
    log("got a response: "+response );
  } else if (ev.isFault()){
    log("got a fault: "+ ev.getMessage());
  } else {
    log("got a generic error: "+ev.getMessage
  }
} else {
  log("timeout.");
}
```

```
...
WSOp op = new WSOp(aId,"echoStruct",arg0);
SensorId sid = linkSensor(new DefaultSensor());
execOp(op,sid);

Perception p = sense(sid,5000);
if (p!=null){
  WSEvent ev = (WSEvent)p.getEvent();

  if (ev.isResponse()){
    String response = ev.getMessage();
    log("got a response: "+response );
    try {
      EchoStructReturnDocument resp =
            EchoStructReturnDocument.Factory.parse(response);
      SOAPStruct obj = resp.getEchoStructReturn();
      log("info: VarInt "+obj.getVarInt()+
          " VarFloat "+obj.getVarFloat()+
          " VarString "+obj.getVarString());
    } catch (Exception ex){
      log("invalid return message.");
    }
  } else if (ev.isFault()){
    log("got a fault: "+ ev.getMessage());
  } else {
    log("got a generic error: "+ev.getMessage());
  }
} else {
  log("timeout.");
```

A REAL-WORLD CASE STUDY: STIL PROJECT

- simpA-WS has been used to engineer the SOA-WS infrastructure for the "STIL" project
  - STIL = Strumenti Telematici per l'Interoperabilità delle reti di imprese: Logistica digitale integrata per l'Emilia-Romagna)
  - "un progetto della durata di 24 mesi finanziato dalla Regione Emilia-Romagna nell'ambito dell'Iniziativa 1.1 del Piano Telematico Regionale"
  - "Il progetto STIL è finalizzato alla valorizzazione della attività di ricerca del territorio regionale attraverso lo sviluppo di un centro di eccellenza regionale nella ricerca industriale di nuovi strumenti ICT per la realizzazione di un distretto digitale della logistica"
  - http://stil.pc.unicatt.it/
- [STIL slides]

AVAILABLE PROJECTS

- Using simpA-WS for engineering complex Web-Service / Service-Oriented Applications
  - choose a case study & engineer it with simpA-WS
- Extending simpA-WS with WS-* specification
  - WS-Security (work-in progress)
  - WS-Reliable-Messaging
  - WS-Addressing
  - ....