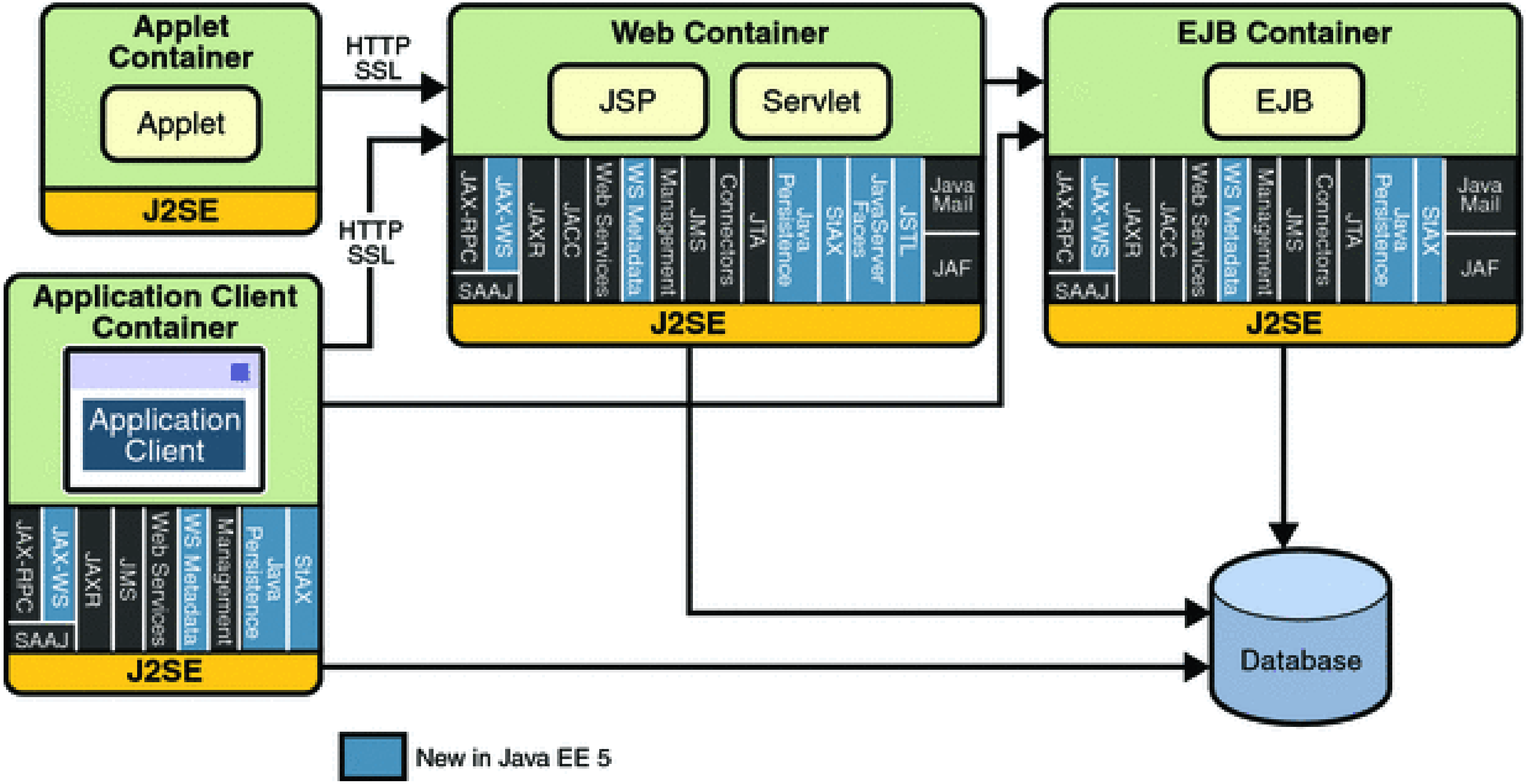
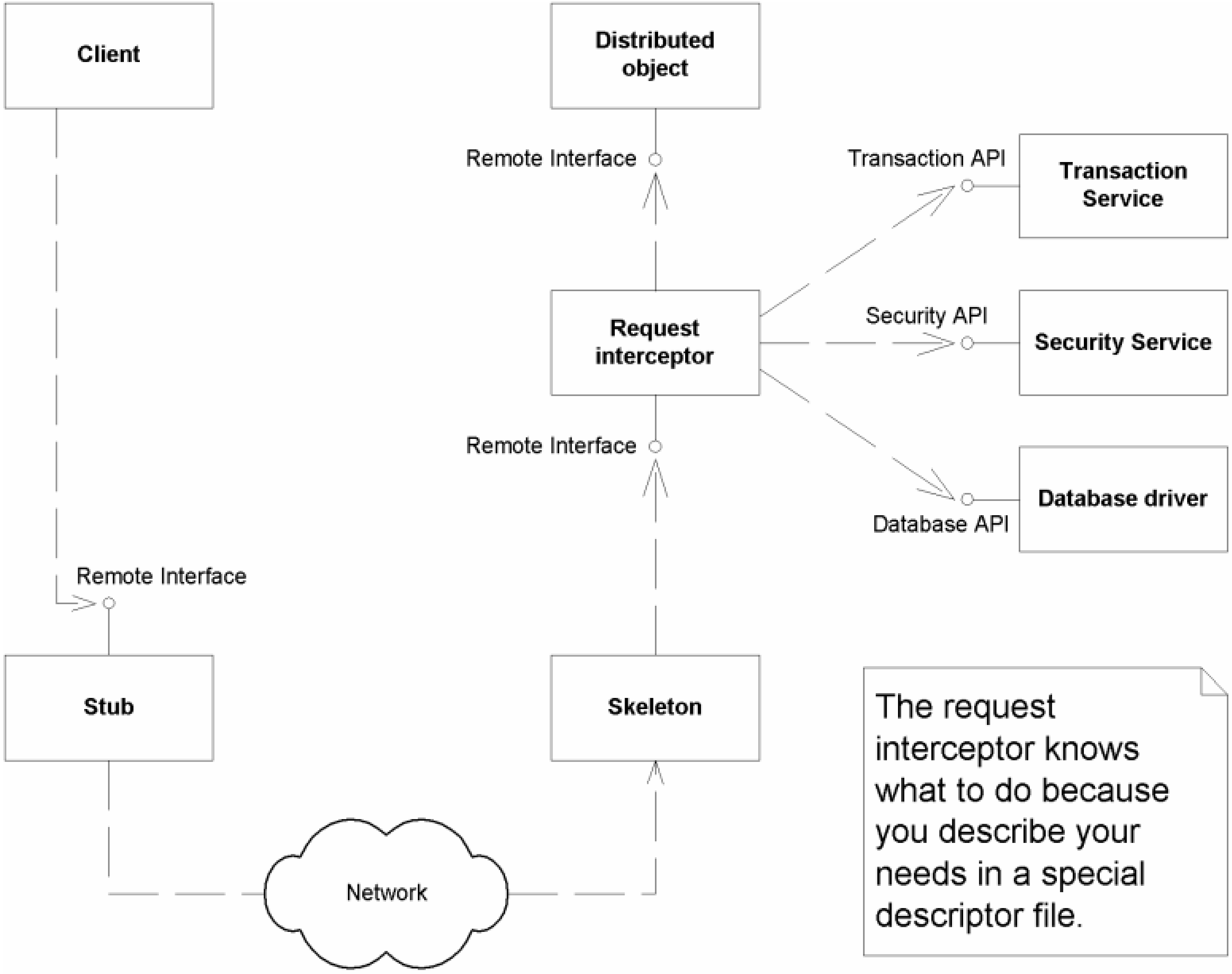
EJB

EJB in J2EE

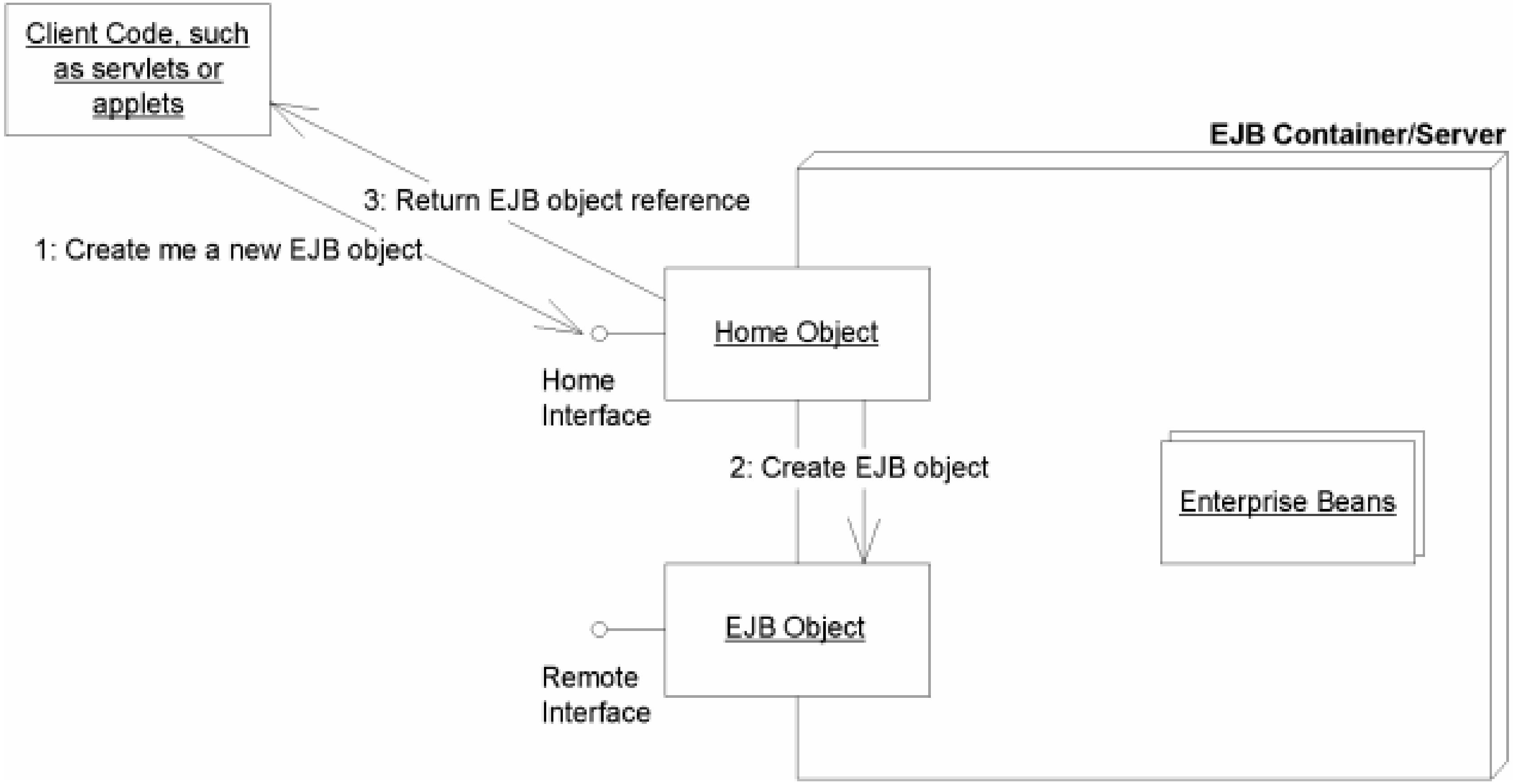

Applet Container
Applet
J2SE
HTTP SSL
HTTP SSL
Web Container
JSP
Servlet
JAX-RPC
JAX-WS
SAAJ
JAXR
JACC
Web Services
WS Metadata
Management
JMS
Connectors
JTA
Java Persistence
StAX
JavaServer Faces
JSTL
Java Mail
JAF
J2SE
EJB Container
EJB
JAX-RPC
JAX-WS
SAAJ
JAXR
JACC
Web Services
WS Metadata
Management
JMS
Connectors
JTA
Java Persistence
StAX
Java Mail
JAF
J2SE
Application Client Container
Application Client
JAX-RPC
JAX-WS
SAAJ
JAXR
JMS
Web Services
WS Metadata
Management
Java Persistence
StAX
J2SE
Database
New in Java EE 5

Basic principles

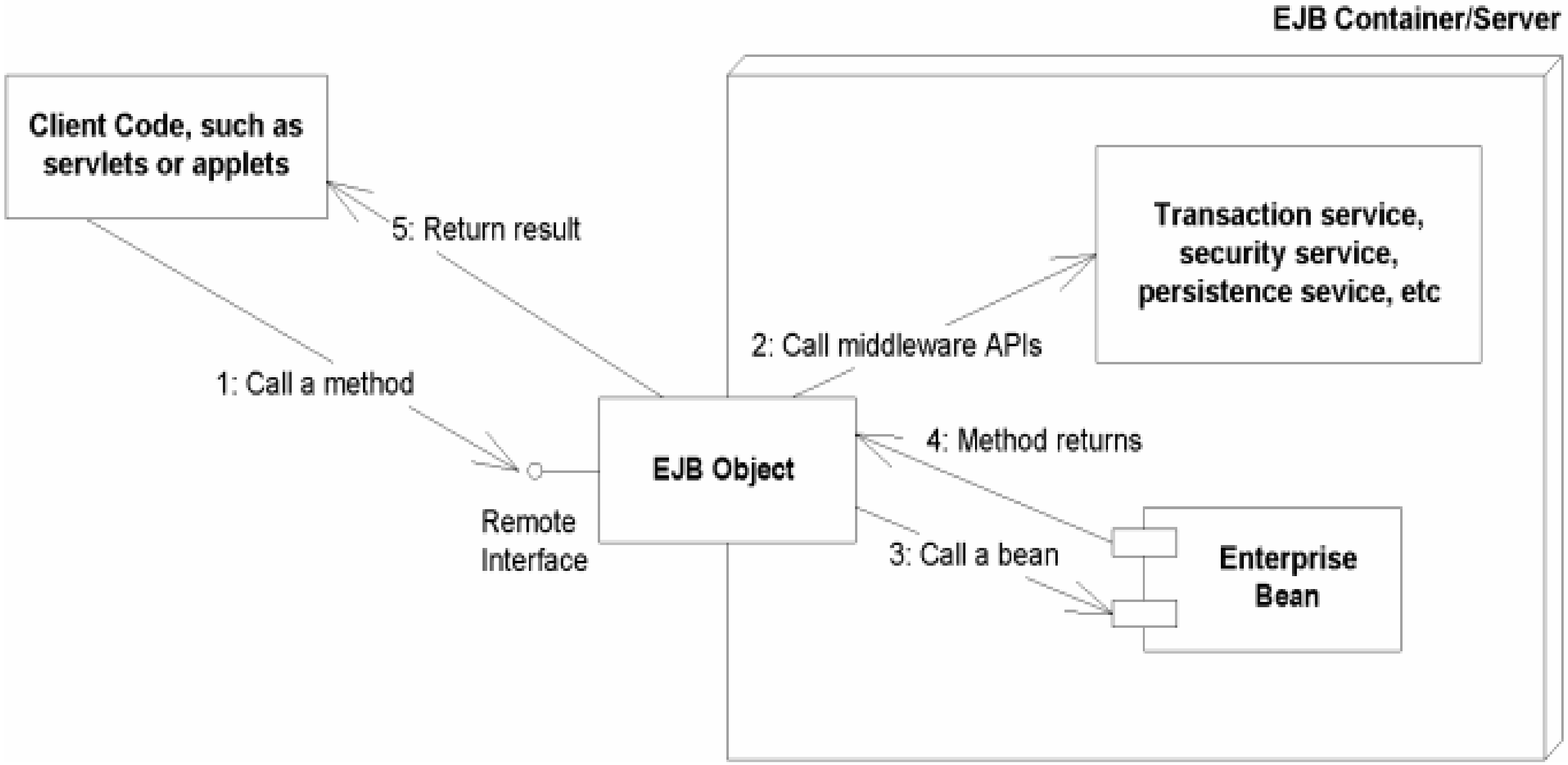


The request interceptor knows what to do because you describe your needs in a special descriptor file.

EJB in action



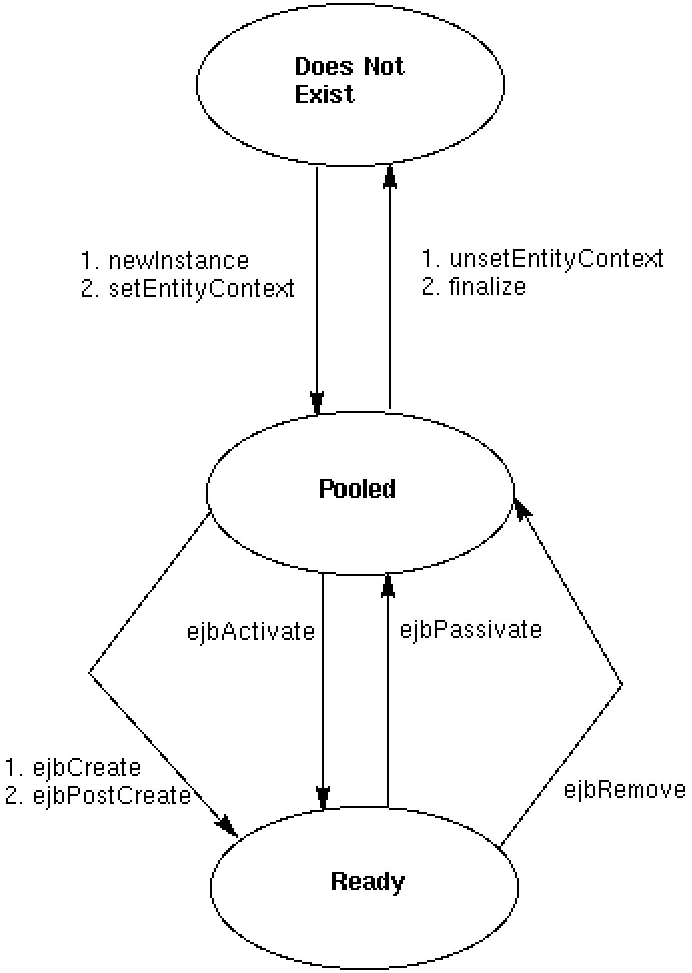
EJB in action



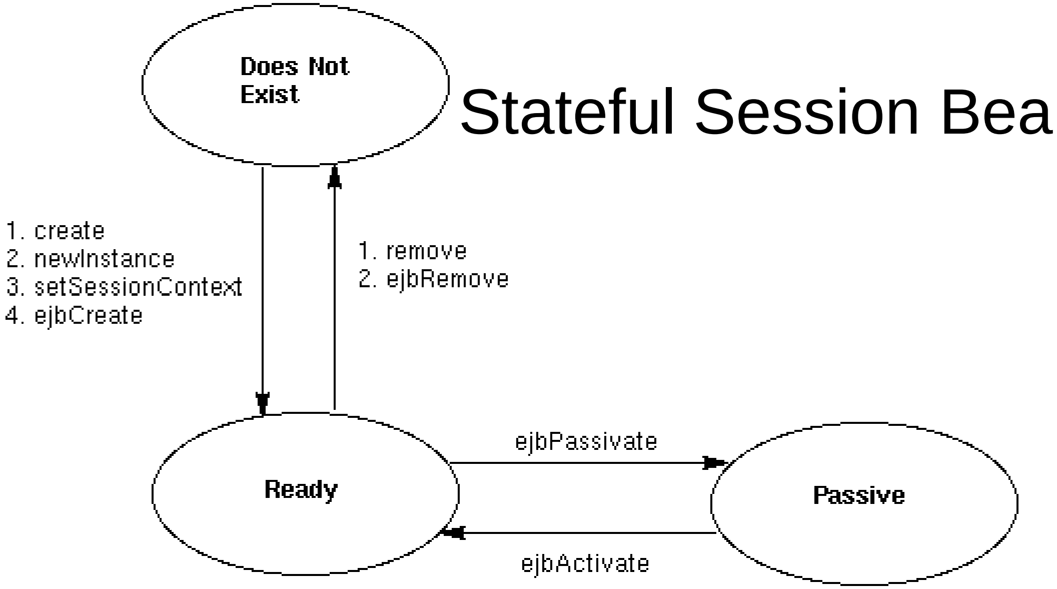
EJB categories

javax.ejb.SessionBean
javax.ejb.EntityBean
javax.ejb.MessageDrivenBean

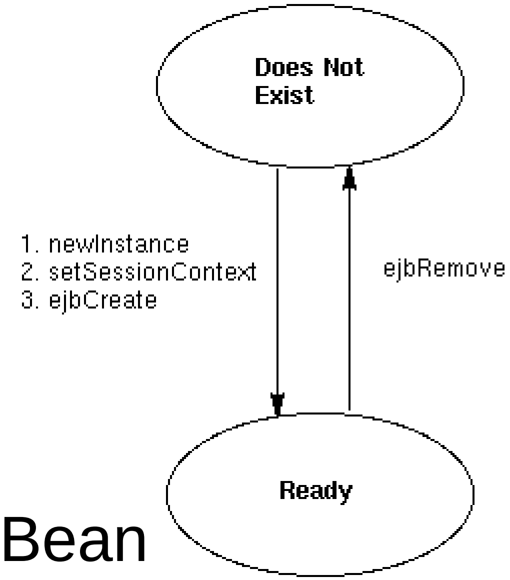
EJB’s life-cycles



Entity Bean



Stateful Session Bean



Stateless Session Bean

EJB development cycle

1. Business interface (Distant or Remote)
2. Management interface (Local or Home)
3. Bean
4. Descriptor
5. Deployment
6. Client

EJB interfaces and component

```
public interface Fibonacci extends javax.ejb.EJBObject {
   public int getFibonacciNumber(int n) throws java.rmi.RemoteException;
}
```

```
import javax.ejb;
import java.rmi;

public interface FibonacciHome extends EJBHome {
 public Fibonacci create() throws CreateException, RemoteException;
}
```

```
import java.rmi.*;
import javax.ejb.*;
public class FibonacciBean implements SessionBean {
  public void ejbPassivate(){}
  public void ejbActivate(){}
  public void ejbRemove(){}
  public void setSessionContext(SessionContext ctx){}
  public void ejbCreate() {}
  public int getFibonacciNumber(int n){//...}
}
```

EJB descriptor

```
<?xml version="1.0"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise
   JavaBeans 2.0//EN" "http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
  <enterprise-beans>
    <session>
         <ejb-name>Fibonacci</ejb-name>
         <home>exemple.fibonacci.FibonacciHome</home>
         <remote>exemple.fibonacci.Fibonacci</remote>
         <ejb-class> exemple.fibonacci.FibonacciBean</ejb-class>
         <session-type>Stateless</session-type>
         <transaction-type>Container</transaction-type>
      </session>
      ...
  </enterprise-beans>
</ejb-jar>
```

EJB deployment

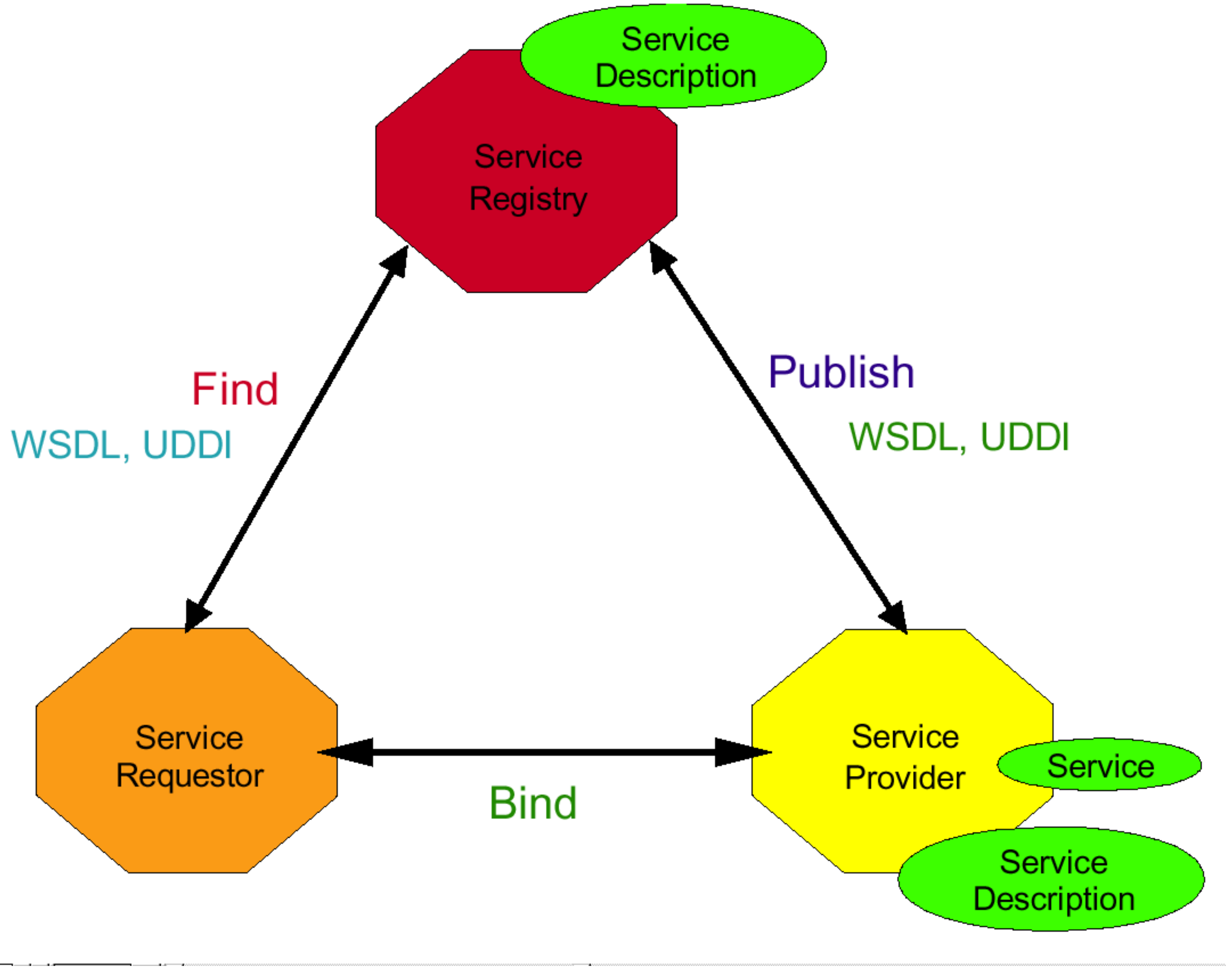
```
<?xml version="1.0"?>
<!DOCTYPE weblogic-ejb-jar PUBLIC "-//BEA Systems, Inc.//DTD WebLogic 6.0.0
   EJB//EN" "http://www.bea.com/servers/wls600/dtd/weblogic-ejb-jar.dtd" >
<weblogic-ejb-jar>
  <weblogic-enterprise-bean>
    <ejb-name>fibonacci</ejb-name>
    <stateless-session-descriptor>
      <pool><max-beans-in-free-pool>100</max-beans-in-free-pool></pool>
    </stateless-session-descriptor>
    <transaction-descriptor>
       <trans-timeout-seconds>300</trans-timeout-seconds>
    </transaction-descriptor>
    <jndi-name>fibonnaciApp</jndi-name>
  </weblogic-enterprise-bean>
</weblogic-ejb-jar>
```

JSP client

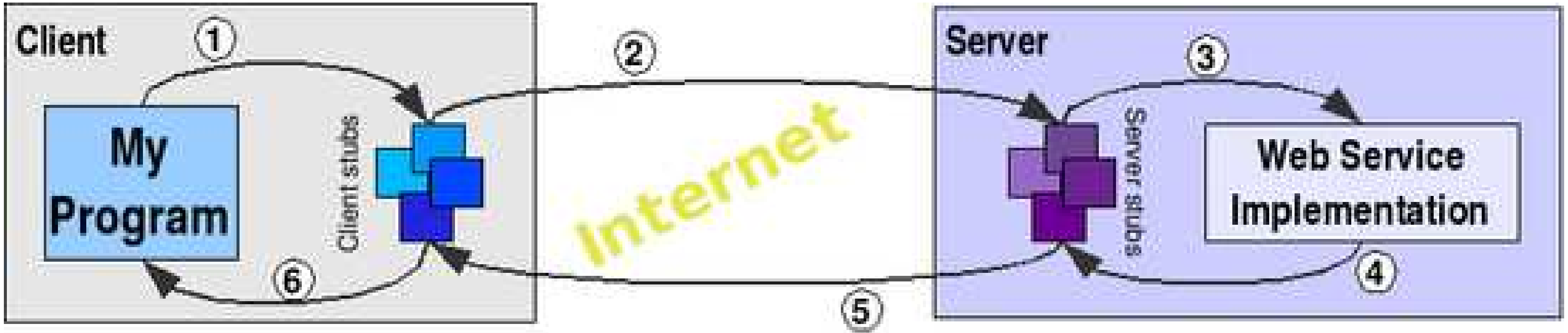
```
<html><head><title> <%= pagetitle %> </title></head>
<%!String pagetitle = « JSP et EJB";%>
<%try {
        ctx = getInitialContext();
        homeFib = (FibonnacciHome) ctx.lookup("fibonnaciApp");
        uneSuite=homeFib.create();
        out.println("fib(7)="+uneSuite.getFibonacciNumber(7));
    }catch(Exception e){e.printStackTrace();
    }
%>
</body></html>
```

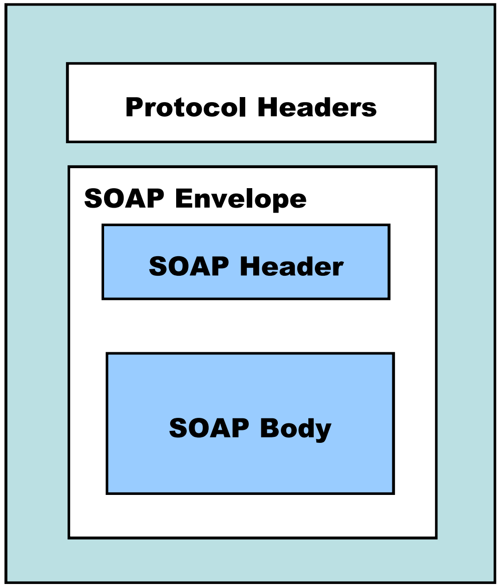
WEB SERVICE

WS architecture



WS in action



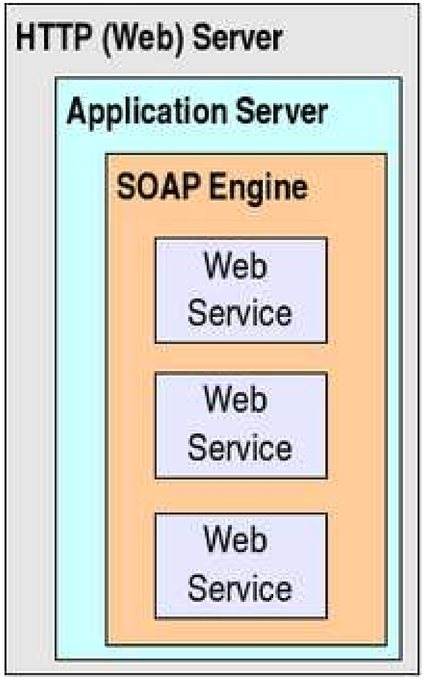


Protocol Headers

SOAP Envelope

SOAP Header

SOAP Body



HTTP (Web) Server

Application Server

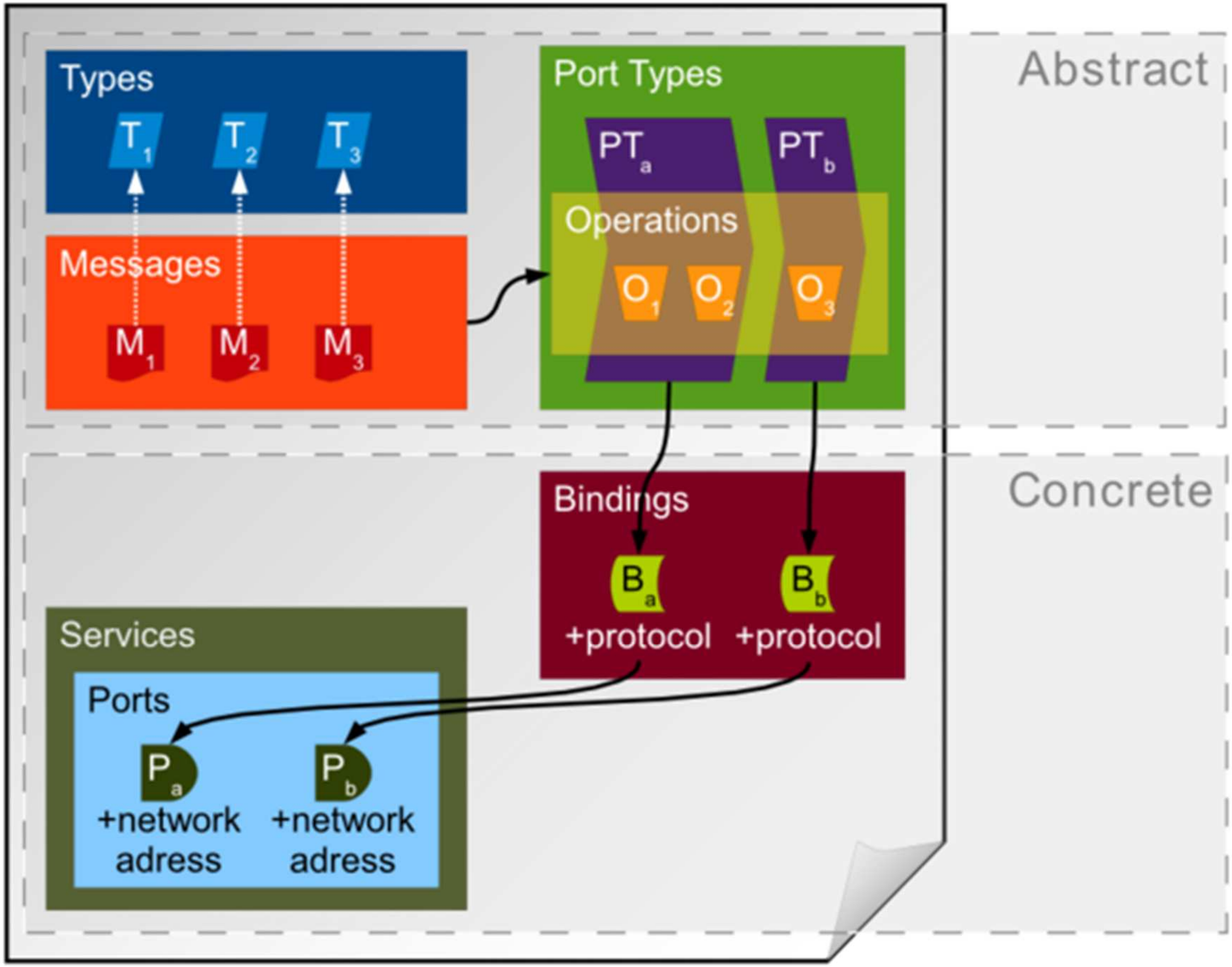
SOAP Engine

Web Service

Web Service

Web Service

WSDL



Abstract
Types
T1
T2
T3
Messages
M1
M2
M3
Port Types
PTa
PTb
Operations
O1
O2
O3
Concrete
Bindings
Ba
+protocol
Bb
+protocol
Services
Ports
Pa
+network adress
Pb
+network adress

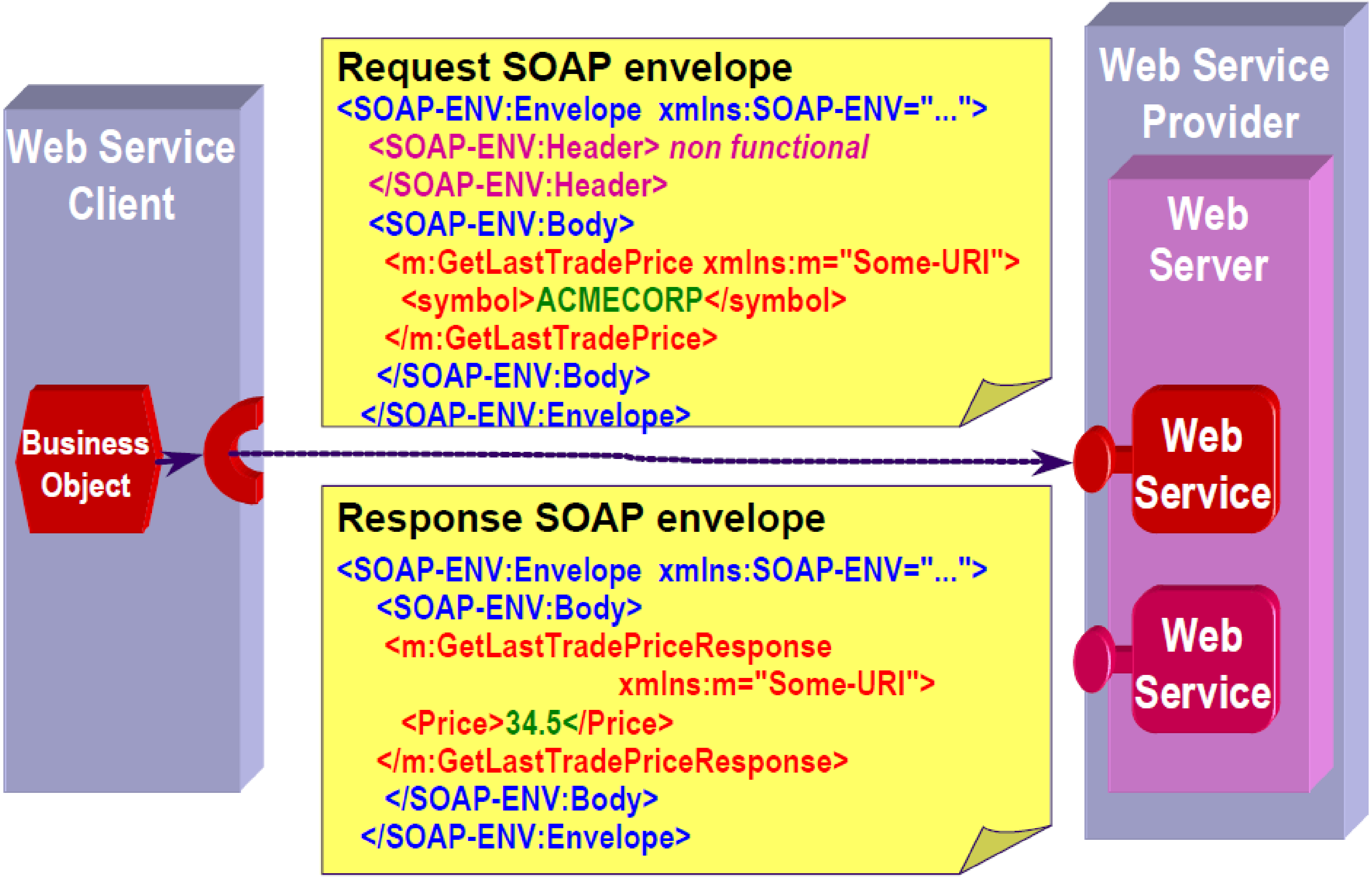
WSDL

```
<?xml version="1.0" ?>
<definitions name="CarnetAdresse" targetNamespace="http://exemple.com/carnetAdresse.wsdl"
      xmlns:per="http://example.com/carnetAdresse.wsdl"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns="http://schemas.xmlsoap.org/wsdl/">
        <types>
        <xsd:schema targetNamespace="http://exemple.com/personne.xsd"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema">
        <xsd:complexType name=“Personne”>
                <xsd:element name=“Nom”  type=“xsd:string”>
                <xsd:element name=“Prenom”  type=“xsd:string”>
                <xsd:element name=“Age”  type=“xsd:float”>
        </complexType>
        </types>
        <message name="GetPersonne">
             <part name="body" type="Personne"/>
        </message>
        <message name="AddPersonne">
                <part name="Nom" type="xsd:string"/>
                <part name="Prenom" type="xsd:string"/>
                <part name="Age" type="xsd:float"/>
        </message>
        <portType name="CarnetAdresse">
             <operation name="AddPersonne">
                <input message="AddPersonneRequest"/>
             </operation>
             <operation name="GetPersonneParNom">
                <input       message="GetPersonneParNom Request"/>
                <output       message=" GetPersonneParNom Response"/>
             </operation>
        </portType>
```

WSDL

```
    <binding type="CarnetAdresseBinding">
        <soap:binding transport=http://schemas.xmlsoap.org/soap/http style="rpc" />
        <operation name="GetPersonneParNomOpe">
        <soap:operation  soapAction="http://exemple.com/ GetPersonneParNom " />
           <input>
                         <soap:body use="encoded"
                           encodingStyle="schemas.xmlsoap.org/soap/encoding"/>
            </input>
            <output>
                         <soap:body use="encoded"
                           encodingStyle="schemas.xmlsoap.org/soap/encoding"/>
            </output>
        </operation>
    </binding>
    <service name=“CarnetAdresseService">
           <port name=" CarnetAdressePort " binding="per:PersonnesBinding">
           <soap:address location="http://example.com/GetPersonneParNom"/>
             </port>
    </service>
</definitions>
```

SOAP



Request SOAP envelope

```
<SOAP-ENV:Envelope  xmlns:SOAP-ENV="...">
   <SOAP-ENV:Header> non functional
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>ACMECORP</symbol>
    </m:GetLastTradePrice>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

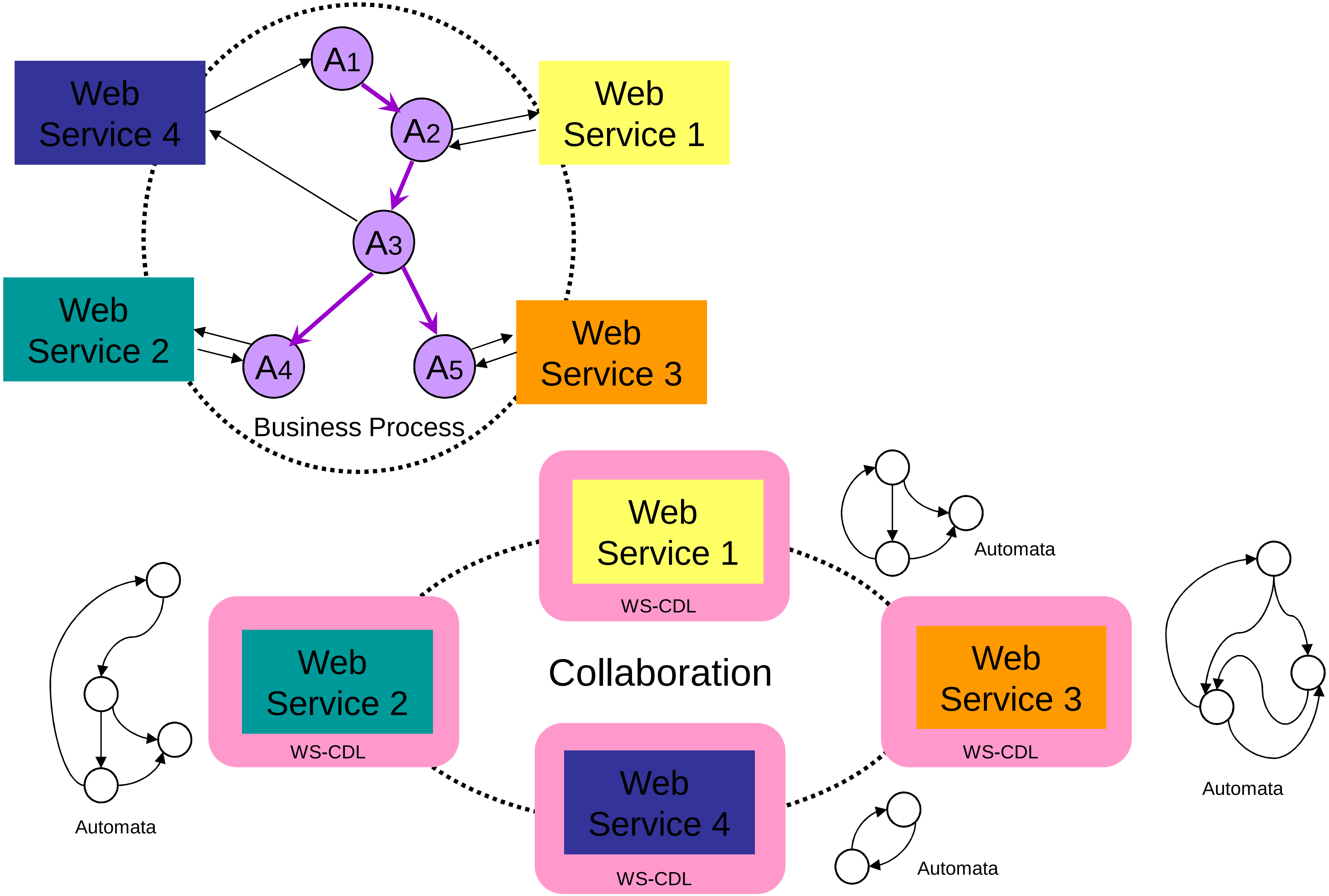
Response SOAP envelope

```
<SOAP-ENV:Envelope  xmlns:SOAP-ENV="...">
    <SOAP-ENV:Body>
     <m:GetLastTradePriceResponse
                        xmlns:m="Some-URI">
      <Price>34.5</Price>
    </m:GetLastTradePriceResponse>
     </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

WS development cycle

- Write your service implementation
- Make all your classes available to the toolkit (Apache Axis, Jonas…)
- Deploy your service *(usually done just once)*
- *R*estart the whole servlet engine
- Test it with a client request

Orchestration versus Choregraphy


Web Service 4
A1
A2
Web Service 1
A3
Web Service 2
A4
A5
Web Service 3
Business Process
Web Service 1
WS-CDL
Automata
Web Service 2
WS-CDL
Collaboration
Web Service 3
WS-CDL
Automata
Automata
Web Service 4
WS-CDL
Automata