Service Component Architecture (SCA) and Java™ Platform, Enterprise Edition: Integration Inside

Ron Barack
ron.barack@sap.com
SAP AG

Peter Peshev
peter.peshev@sap.com
SAP Labs Bulgaria, Ltd.

TS-5706

Java

Sun microsystems

Learn about the concepts of Service Component Architecture and see how it can extend the Java™ Platform, Enterprise Edition (Java EE).

GOAL

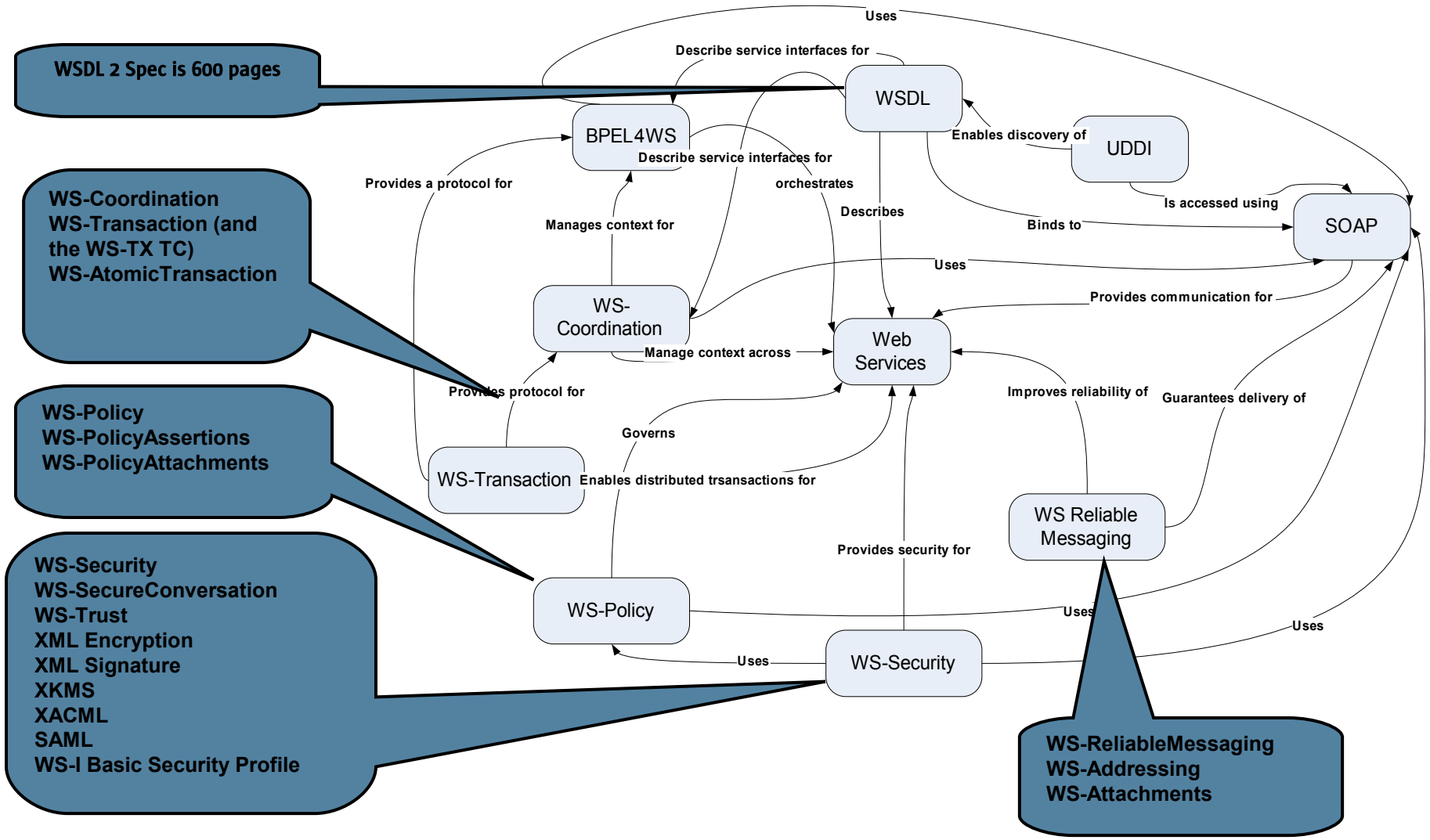
Agenda

- How Service Component Architecture (SCA) Relates to Java™ EE platform
- Building Applications with SCA
- Example: Using SCA for Cross-Technology Applications
- SCA Contributions and the Domain Composite
- More SCA Concepts

SCA Simplifies Service-Oriented Architecture (SOA) Development

- All complex plumbing code, protocols, technologies and configuration should be done by the infrastructure, not by developers
  - Programmers focus on service calls (only)
  - The framework integrates the technologies and makes sure that all policies (security, reliability, transactionality, QoS) are enforced
  - The framework converts between different data representations
- Configuration is done by simple XML file(s)
  - „Deployment Descriptors on steroids“
  - Regular text editor should be enough to create those
  - But tooling makes things even easier!
- Should simplify the development, and not introduce new complex programming model and another layer of complexity

Consider the Alternative



Open SOA – The Collaboration Behind SCA@Java EE

Where is SCA today ?

- 1.0 SCA specs are available under
  http://www.osoa.org/
- SCA \ Java EE integration spec has 0.9 version under
  http://www.osoa.org/
- 1.1 specs are developed under OASIS as part of the Open Composite Services Architecture (Open-CSA)
  http://www.oasis-opencsa.org/

The Java EE Platform Provides the Basics

- The Java EE platform provides an industry standard
  - Programming model for
    - named, remotable, demarcated (Tx/Sec) components: Enterprise JavaBeans™ (EJB™) technology
    - Persistence: Java Persistence API (JPA)
    - Web Applications (Servlets, JavaServer™ Pages (JSP™) technology, JavaServer Faces technology, etc.)
    - Web Service provision and consumption (Java API for XML Web Services Addresssing (JAX-WS))
    - Messaging (Java Message Service API (JMS)/ Message Driven Beans (MDBs))
    - Connection-oriented Backend access (J2EE™ Connector Architecture)
  - Assembly model that describes how
    - EJB technology uses EJB technology, Web Components use EJB technology, EJB technology use Resource Adapters…
  - Packaging model
    - Modules (.jar, .war, .rar) to Enterprise applications (.ear).
  - Declarative security and transaction demarcation

Evolutionary Approach

- Preserve investment in Java EE technology and Java EE platform knowledge
  - User Base
  - Applications
- The programming model is only part of the story!
  - Tools
  - Infrastructure for
    - Lifecycle management
    - Monitoring
    - Administration
- Therefore:
  - Evolutionary (not revolutionary) approach, starting with Java EE technology, and adding as much SCA as justified by the use-case
  - No new packaging to be introduced

SCA Extends Java EE Technology into the SOA Realm

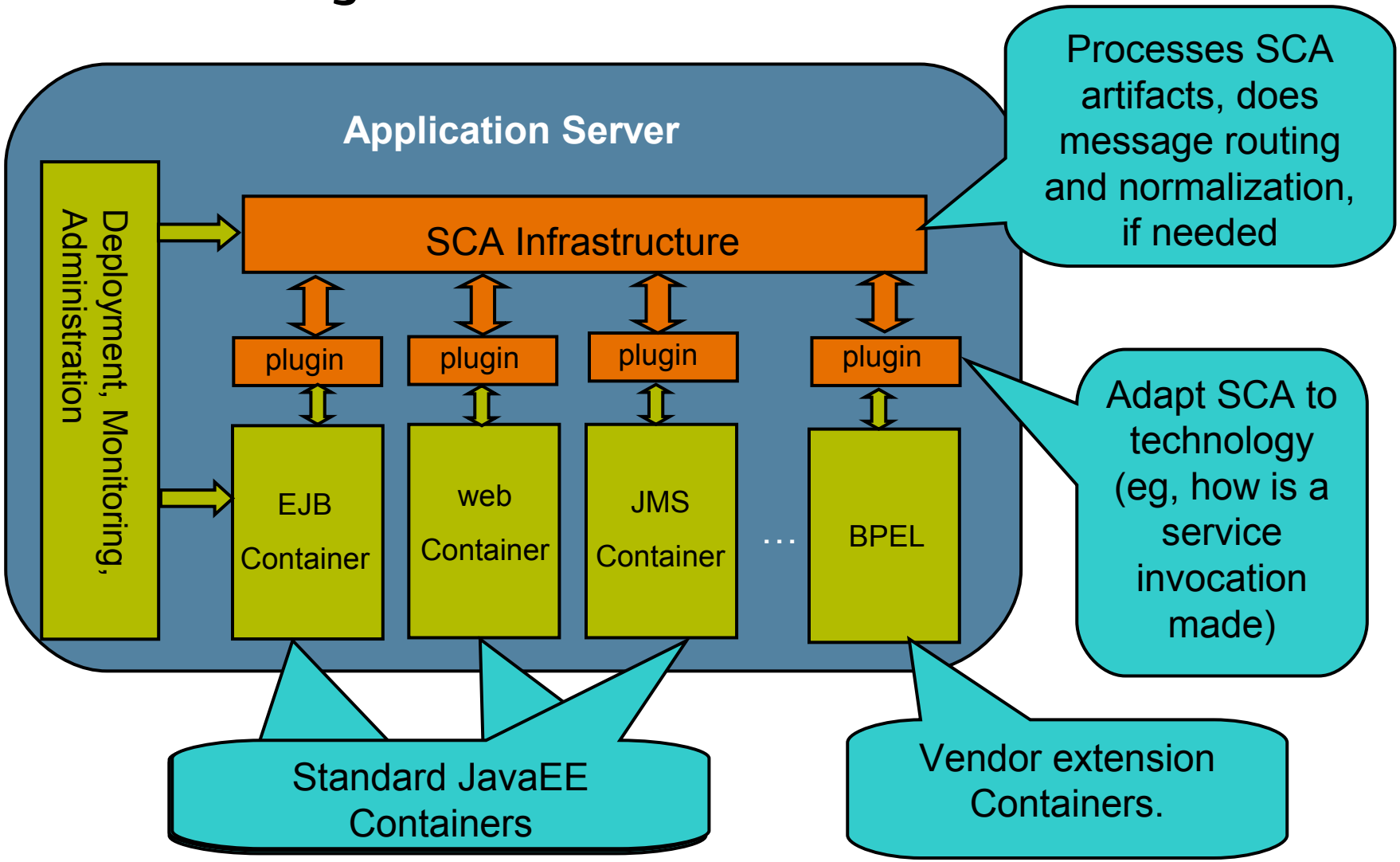
- Build applications that cross technology boundries:
  - Business Process Execution Language (BPEL), Spring, Scripting Languages, Enterprise Service Bus (ESB) style interactions, etc.
- Move protocol specifics to configuration via bindings
- Cross-Application Assembly
  - Domain assembly extends Enterprise App Assembly to the cross-app level
- SCA Wiring automatically converts between the data representations used by the individual components
  - Across Classloaders
  - Between Generic and TypeSafe Data Access
  - Between XML- and Object- Oriented Components

SCA-Enabled Java EE Platform Runtime

- *Not* a new component model
- *Not* a layer above (hiding) Java EE technology!
- The standard containers (EJB technology, Web, etc.) are enhanced, adding SCA aspects while still remaining Java EE platform complient
- Code and other artifacts mix Java EE technology with SCA
  - Annotations
  - Deployment Descriptors
- Packaging and Deployment of standard artifacts (Enterprise Application Archive (EAR), Web Application Archive (WAR), etc.)
- SCA deployment *is* Java EE deployment

SCA extends Java EE technology, does not hide or compete with it!

Possible Design of an SCA Enabled Runtime



Agenda

- **How Service Component Architecture (SCA) Relates to Java™ EE platform**
- Building Applications with SCA
- Example: Using SCA for Cross-Technology Applications
- SCA Contributions and the Domain Composite
- More SCA Concepts

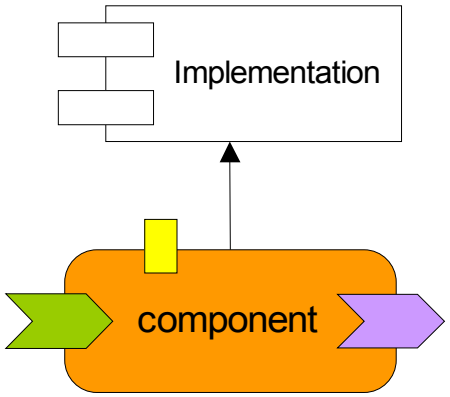
All the programmer has to do is…

```
@javax.ejb.Stateless
public class MyBeanImpl implements MyBean
{
    @Reference MyInterface myReference;
    public void doSomething() {
        …
        myReference.serviceCall(…);
        …
    }
}
```

- @EJB, @WebServiceRef tell you which technology is being used to implement the service, @Reference doesn't
- SCA specs – definition of XML artefacts that explain the configuration (protocols, technologies, properties, policies, some other stuff invisible to the developer)

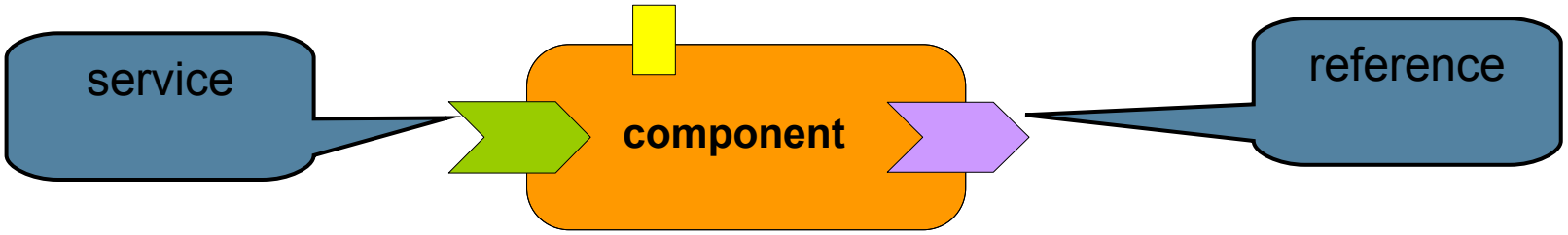
Components are pieces of Business Logic implemented using some Technology

- Components can be anything that can provide services or use the services offered by other components
- Java EE technologies include
  - EJB technologies
  - Web modules (war)-s
  - Java EE Applications
- Non-Java EE technologies include
  - BPEL
  - Java POJOs
  - Spring Beans
- SCA is Extensible – vendors can define their own implementation types

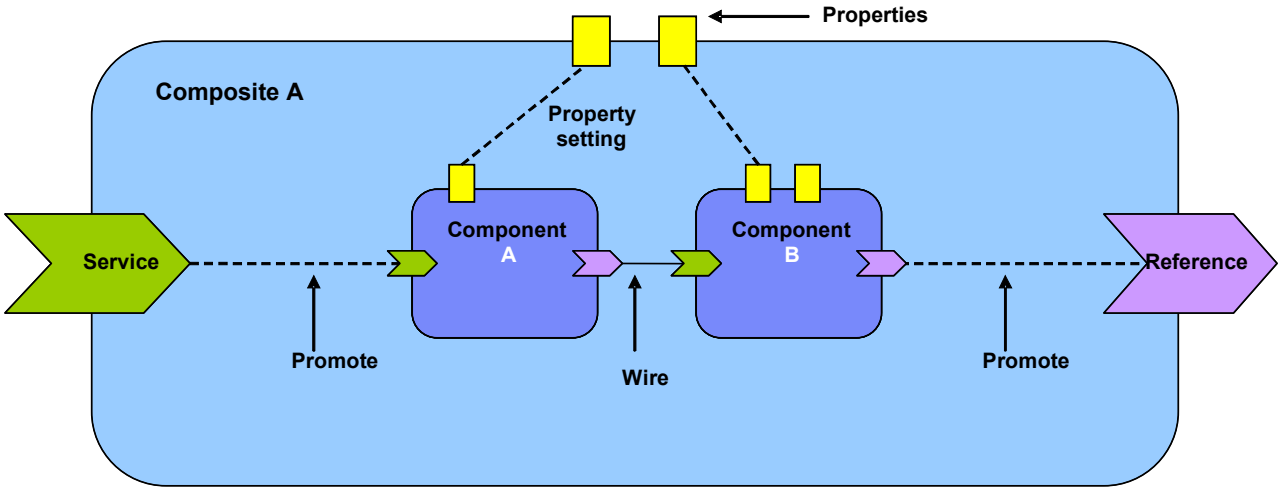


Components provide services to other components, and consume services, too

- SCA focuses on
  - what **services** a component offers, and who uses them, and what services the component uses
  - what **reference** the component uses, and who provides them



SCA Assembly – Deployment Descriptors on Steroids



An SCA Assembly is basically a bag of **components**, connected by **wires**

The implementation type doesn‘t really show up in SCA assembly diagrams.
It‘s just an implementation detail

SCA assembly defined using XML, SCDL (pronounced „skiddle“)

Anatomy of a Component Description

The component's name, used in identifying the component as a wire target

This line says that the component is implemented by an EJB. The ejb-link attribute identifies the particular EJB.

```
<sca:component name="MyComponent">
    <sca:implementation.ejb ejb-link="ejb.jar#BeanName"/>
    <sca:service name="PublicService">
      <sca:interface.java interface="com.acme.IService"/>
    </sca:service>
    <sca:reference name="ExtensionPoint"/>
</sca:component>
```

This element defines a service that the component is offering, and states which interface it supports.

This line names a reference that this component consumes, the assembler should hook this up to a service

Anatomy of a Wire

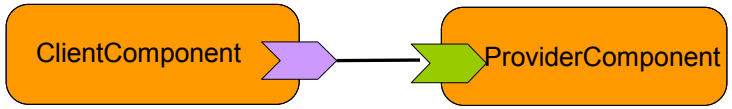
Wires connect services to references

This attribute identifies the name of the component and the reference that will be connected

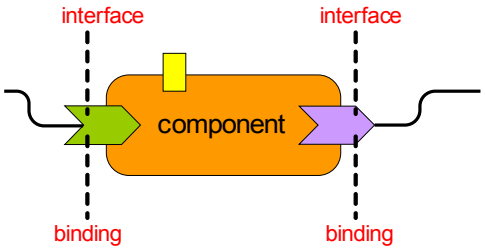
```
<sca:wire
      source="ClientComponent/ReferenceName"
      target="ProviderComponent/ServiceName"
          />
```

This attribute identifies the provider of the service

- This is only one option for defining wires
  - The target could be specified as part of a <reference> tag
  - „autowire“ finds appropriate targets based on interface matching



Bindings provide non-SCA service technologies to SCA

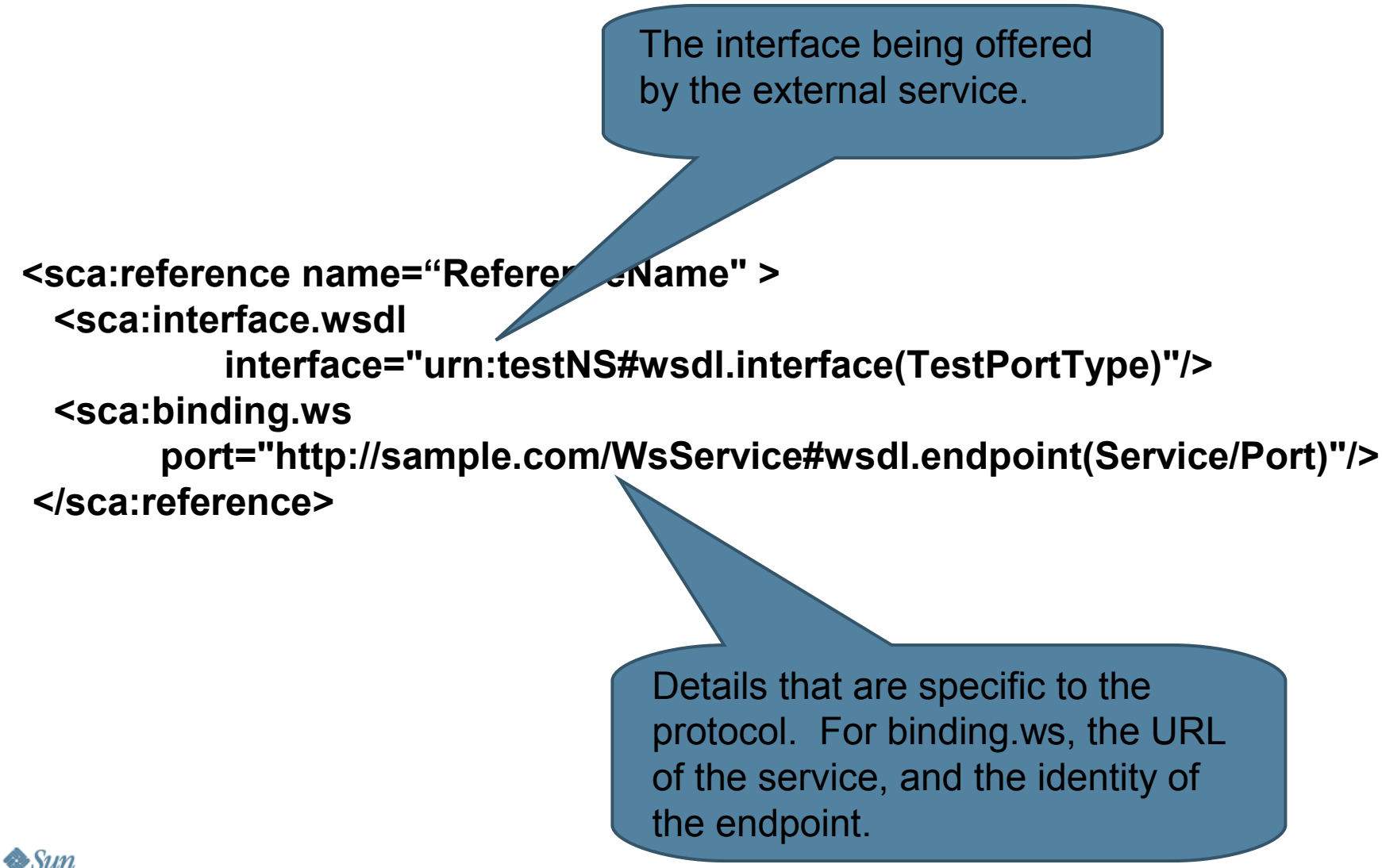


- Exposure of services according to some protocol
- Consumption of services provided by some protocol

  Examples:
  - binding.ws – expose/consume web service
  - binding.jms – receive/send JMS message
  - binding.jca – expose/integrate EIS

- SCA is Extensible – vendors can define their own bindings

Anatomy of an External Binding



Other examples

Details for the destination and connection factory

Instructs the environment to create the JMS resources

```
<sca:reference name="…ference1" >
  <sca:binding.jms>
    <sca:destination name=”MyQueue” create="always"/>
    <sca:connectionFactory name=”MyQCF” create="always"/>
  </sca:binding.jms>
 </sca:reference>
<sca:service name=“ExternalService" >
  <nmsp:binding.xyz   attrib=“abc”/>
</sca:service>
```

Some vendor specific technology

Agenda

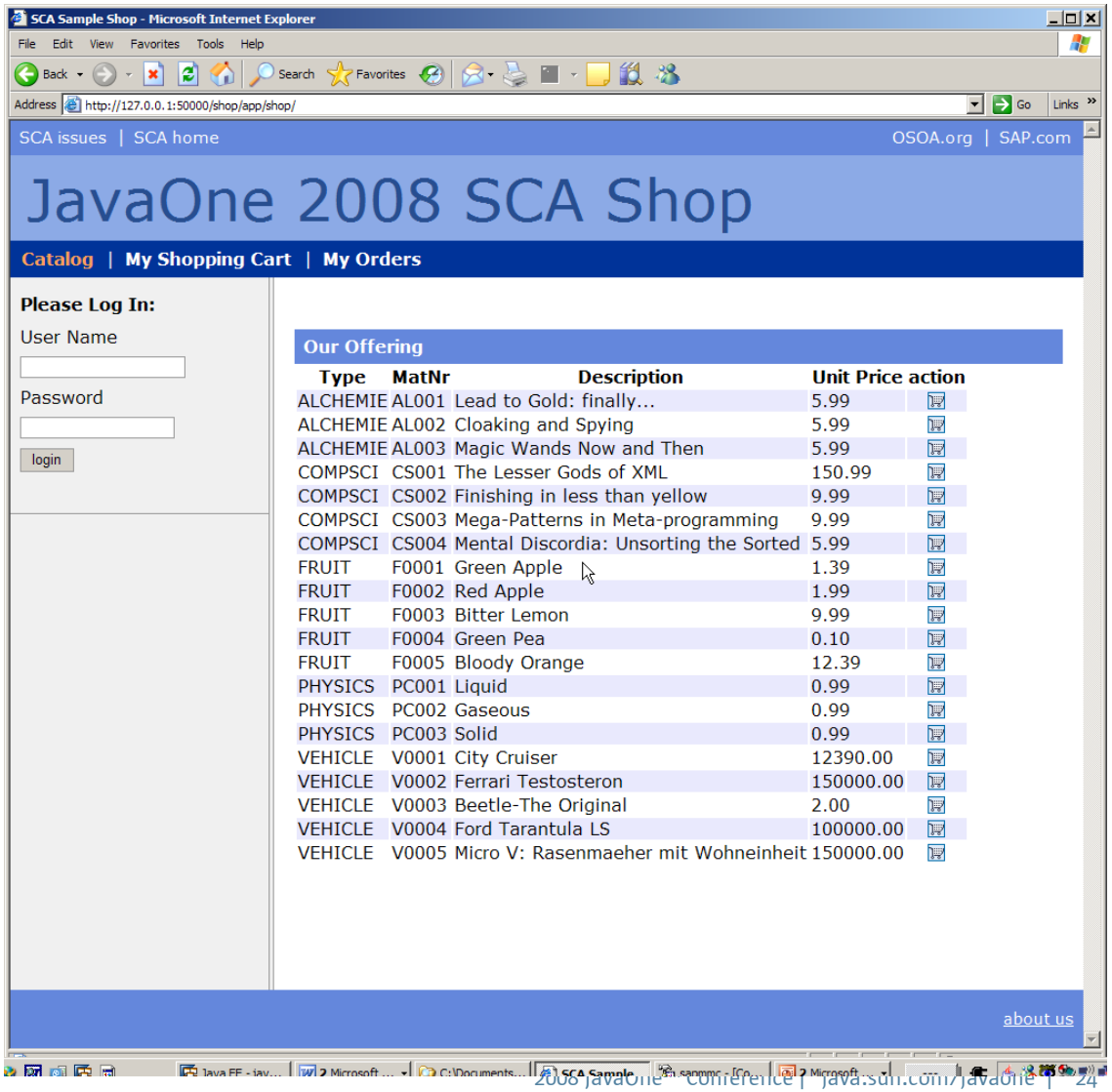
- **How Service Component Architecture (SCA) Relates to Java™ EE platform**
- Building Applications with SCA
- Example: Using SCA for Cross-Technology Applications
- SCA Contributions and the Domain Composite
- More SCA Concepts

Example Scenario

We have a plain old webshop mainly using Java EE software components

But the Java EE platform has no standard representing the business process/ workflow

Therefore, we want to integrate a BPEL component into the application



SCA Provides Integration without WebServices

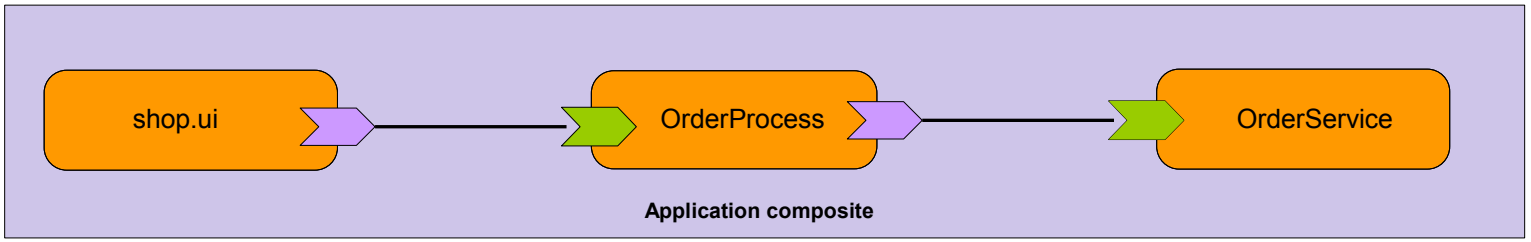
- Integrations with a BPEL process could be done using Web Services. Why use SCA?
  - Web Services impose performance costs, e.g., associated with the generation and parsing of strings
  - Web Services Transactions are very cumbersome.
  - The might be security considerations.
  - Conceptually, the call to the BPEL process is a call to a local component that just happens to be implemented using another technology
    - WebServices are for remote calls across platforms
    - The whole application should have a single lifecycle

Java EE Components as Implementation Types

- Java EE software uses SCA's programming model to consume SCA-exposed services
- Use session beans as Service Component Implementations that may be consumed by other SCA components
- Deploy SCA Components as a part of a Java EE application environment

SCA is the Scale-Out model for the Java EE platform

Example: Creating a Composite




```
<sca:component name="shop.ui>
    <sca:implementation.web war="shop.web.war"/>
    <sca:reference name="orderProcess" target="OrderProcess"/>
</sca:component>
<sca:component name="OrderProcess">
    <sca:implementation.bpel process="shop.bpel" version="2.0"/>
    <sca:reference name="orderServicePL" target="OrderService">
    <sca:service name="OrderProcessRole"/>
</sca:component>
<sca:component name="OrderService">
    <sca:implementation.ejb ejb-link="shop.ejb.jar#OrderService"/>
    <sca:service name="IOrderService"/>
</sca:component>
```

SCA Wires Convert between Data Representations

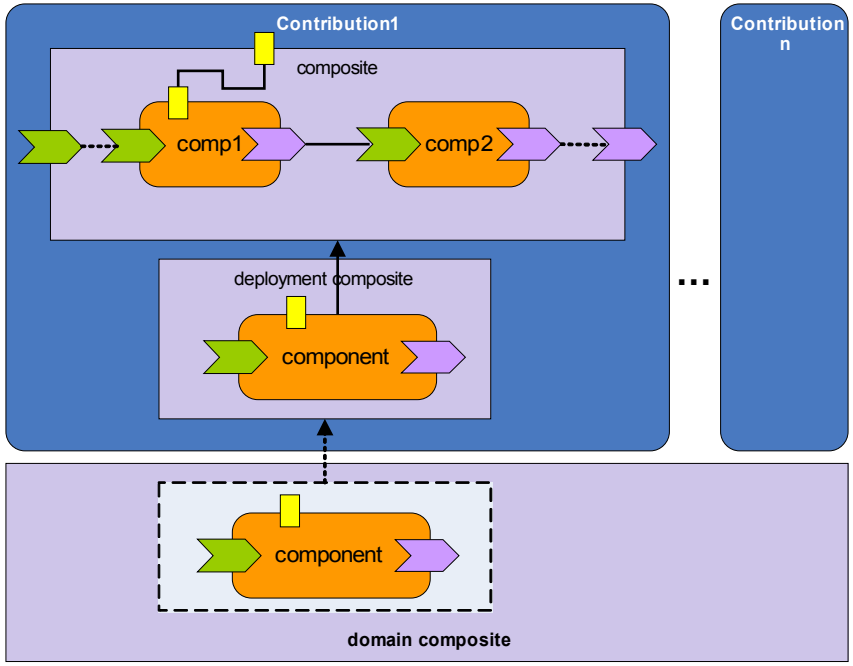
- In this example, OrderService is programmed using standard JPA, and this is reflected in the API it offers. In particular, `IOrderService` accepts and returns POJOs
- BPEL interfaces are defined using WSDL
- SCA wiring must convert between the data representations used on the each side of the wire. Furthermore, the coupling between the objects and their XML representation must be much looser in SOA architectures than is forseen by JAX-B
- Ask Ron about **SDO 3.0** after the Session!

Agenda

- **How Service Component Architecture (SCA) Relates to Java™ EE platform**
- Building Applications with SCA
- Example: Using SCA for Cross-Technology Applications
- SCA Contributions and the Domain Composite
- More SCA Concepts

Deployment and Contributions

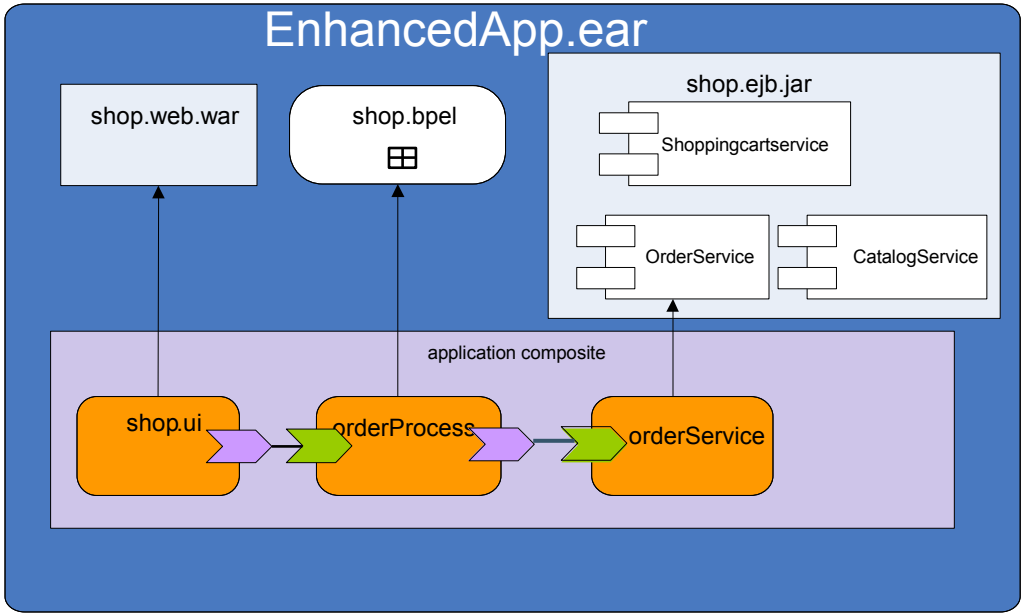
- Runtime artifacts are deployed as part of **Contributions**
- Deployment
  - Activates any external bindings,
  - Exposes components, wires into the virtual **Domain** Composite
- Anything a contribution exposes on the domain can be seen and wired to by the domain components from other contributions.



- The SCA@Java EE spec defines 2 modes of deployment
  - Modified Java EE archives can be used as SCA contributions
  - Java EE archives can be included in larger SCA contributions.

Deploying the example application

- We deploy a modified EAR, containing the normal Java EE platform artifacts together with our SCA composite and other SCA related files, plus a module containing the BPEL process



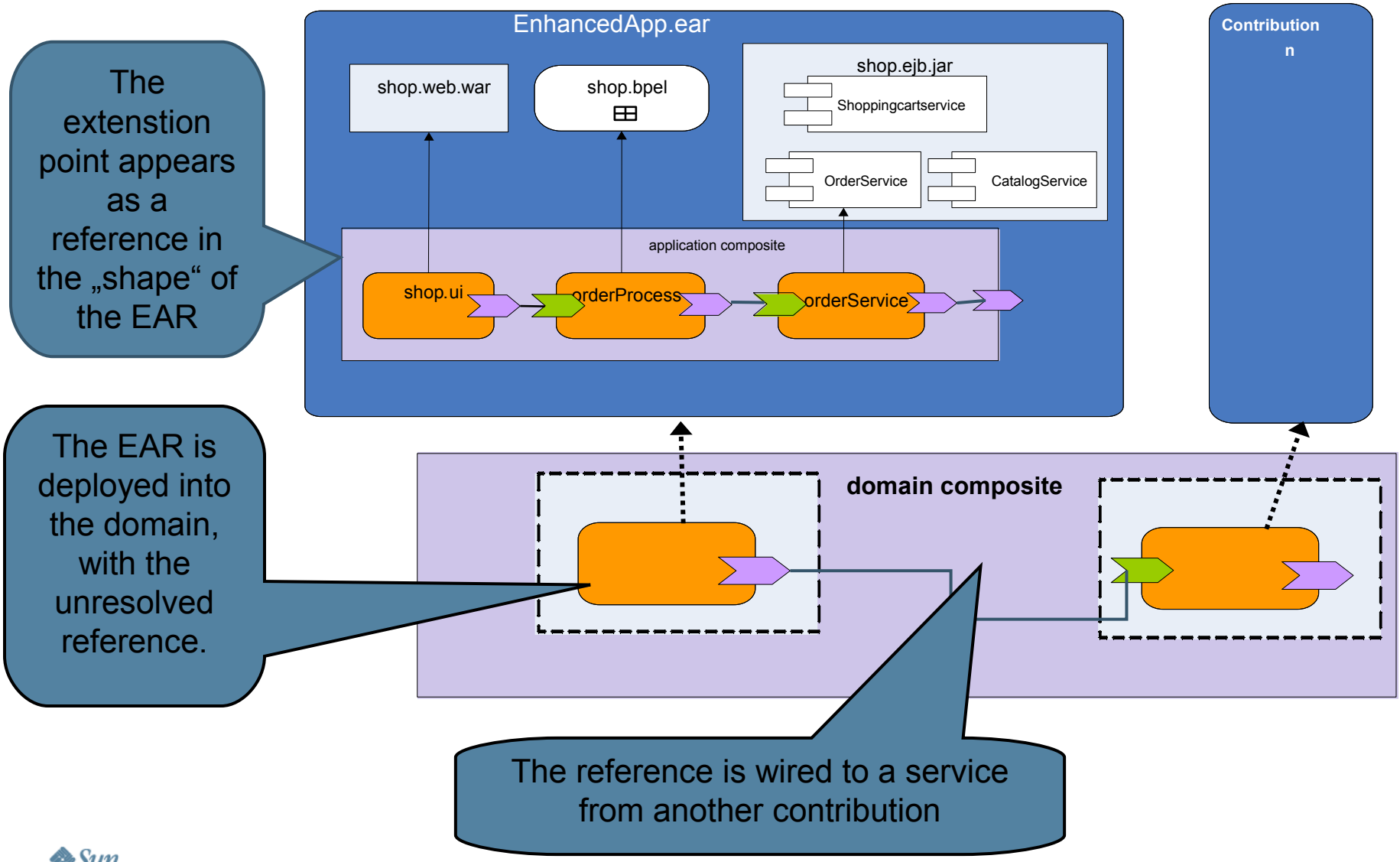
- The advantage here is that we can leverage the tooling, monitoring and application lifecycle management capability already present on the Java EE platform server

Use-Case: Allow the customer to provide his own catalog service

- Image that the shop application described up to now has been packaged by one software provider
- The software provider has supplied his own implementation of the catalog service, implemented as an EJB technology
- At the site where the shop is installed, however, there is already a catalog service running (perhaps implemented using another technology, e.g., RoR)
- The software provider can provide for this by defining an extension point

```
@javax.ejb.Stateless
public class MyBeanImpl implements MyBean {
    @Reference
    @EJB ICatalogService _catalog;
    public void doSomething() {
        …
        _catalog.serviceCall(…);
        …
    }
}
```

Wiring an Extension Point (1)



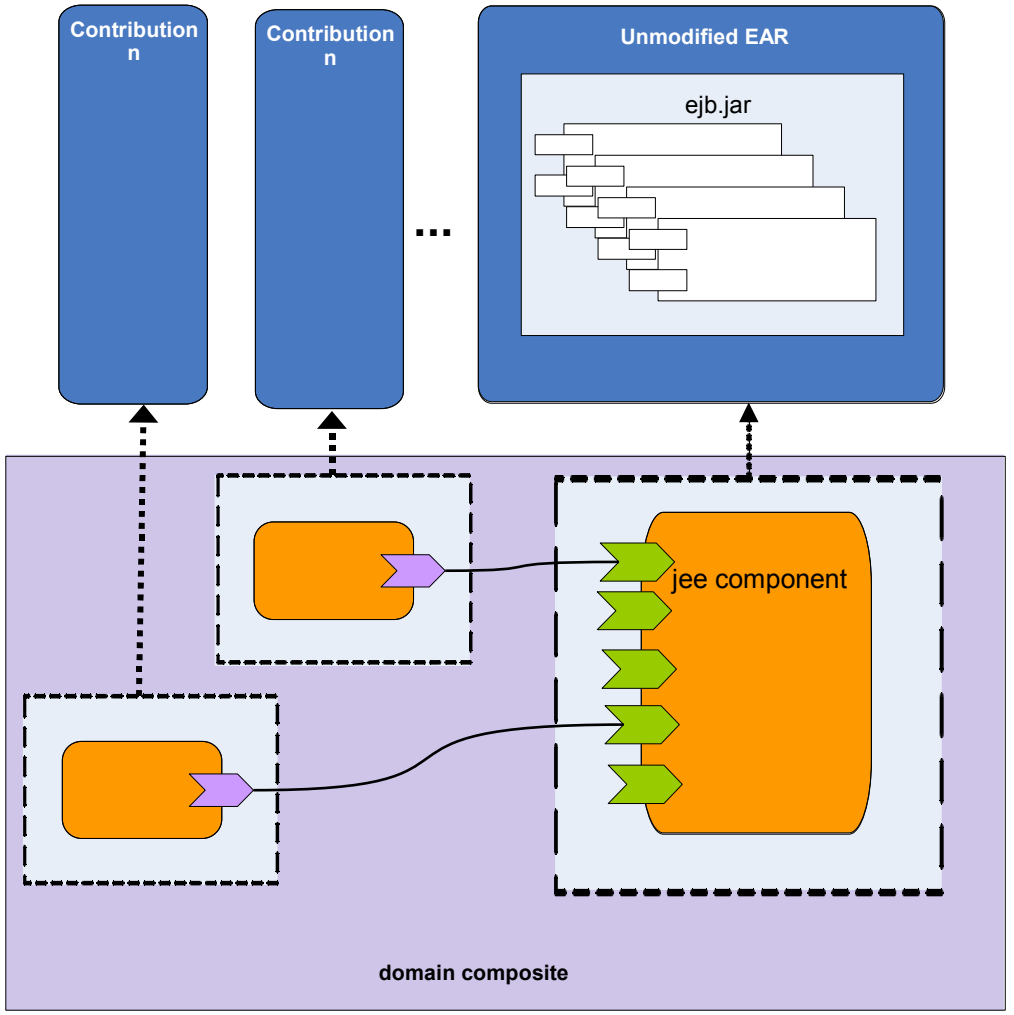
Java EE Applications as Implementation Types

- The whole Java EE application can be classified as one component
- SCA can be used as deployment plan –the deployer can provide target for the @EJB technology references as SCDL-s
  - Thus unmodified Java EE applications can be integrated into a SOA landscape.

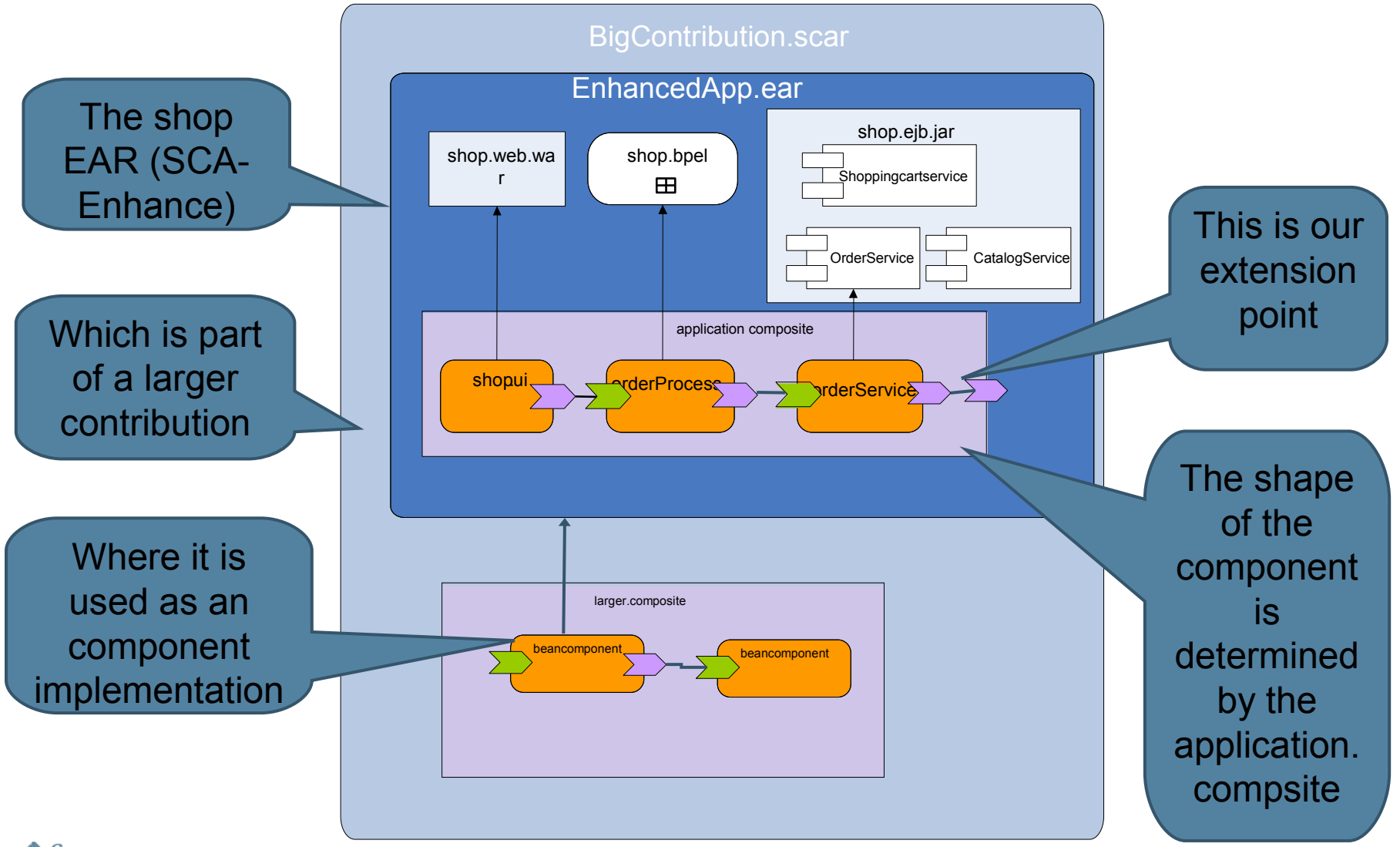
SCA is the level of assembly above the Java EE platform

Exposure of an Unmodified EAR

- An unmodified EAR may be deployed (through implementation.jee)
- By default all EJB technology business interfaces are exposed as SCA services
- Implementations MAY choose to expose remote EJB technology references



Wiring Extension Points (2)



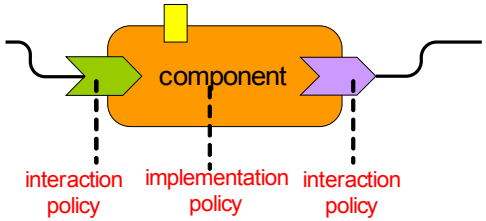
Agenda

- **How Service Component Architecture (SCA) Relates to Java™ EE platform**
- Building Applications with SCA
- Example: Using SCA for Cross-Technology Applications
- SCA Contributions and the Domain Composite
- More SCA Concepts

Security, Reliability, Transactions are declared through Policy Framework

- **Interaction Policies**
  - Declaration of service capabilities
  - Declaration of reference requirements
  - For example: security, reliability
- **Implementation Policies**
  - Declaration of constraints between component and runtime
    (for example transaction demarcation)
- Simplification of complex policy languages by supporting abstract policies called **intents**:

```
@Authentication("message")
public String hello(String message) {…}
```



All the programmer has to do is…

```
@javax.ejb.Stateless
public class MyBeanImpl implements MyBean
{
    @Reference @OneWay @Requires(Confidentiality,
ExactlyOnce) MyInterface myReference;
    public void doSomething() {
        …
        myReference.serviceCall(…);
        …
    }
}
```

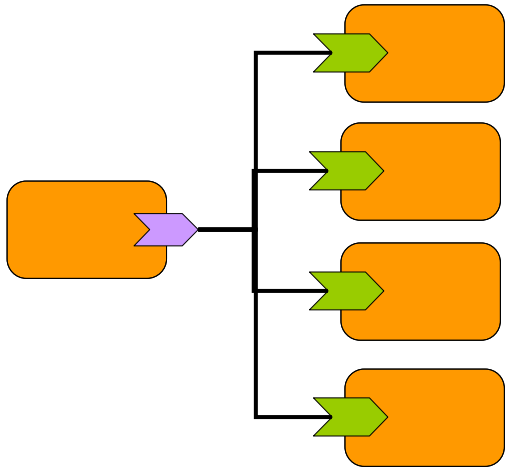
- @OneWay – the call will be asynchronous
- @Requires(Confidentiality, ExactlyOnce)– intents provided by the developers requesting quality of service. Encryption and guaranteed delivery.

Interface Options

- Conversational interfaces
  - The runtime will route the call to the same instance
  - ConversationId will be supplied for technologies that wish to manage their own state
- Callbacks
  - Interfaces may be bi-directional
  - The service can make callbacks to the client via @Callback
  - Callbacks can be changed dynamically via API

Cardinality of References

```
@Reference MyInterface[] myReferences;
public void doSomething() {
    …
    for (MyInterface ref: myReferences) {
        ref.serviceCall(…);
    }
    …
}
```



- References can have cardinality
  - 0..1 – may be wired
  - 1..1 – must be wired
  - 0..n – may be wired multiple times
  - 1..n – must be wired at least once
- For cardinality > 1, client code sees it as an array (or list)
- Example: ask each component for price and select which should be used to place order.

Dynamic Resolution of References

- Java EE can resolve all references at deploytime
- SCA allows contributions to expose a reference on the domain
- The target of the reference may change as other contributions are deployed to or undeployed from the domain.
- Runtimes MAY re-inject references
- The safe way to get current value is to use the API

Other SCA Sessions

- TS-5870

  The Best of Both Worlds with Java™ Business Integration and Service Component Architecture

- TS-5850

  SCA: Flexible and Agile Composition of Distributed Service-Oriented Architecture Applications

- TS-5918

  Open-Source Service-Oriented Architecture with Service Component Architecture and Apache Tuscany

- PAN-5188

  Open Standards for SOA and Java™ Technology

- SAP Booth 3:30

  View a demo of the EJB \ BPEL application and the tooling

THANK YOU

Ron Barack & Peter Peshev

Service Component Architecture (SCA) and Java™ Platform, Enterprise Edition: Integration Inside

TS-5706