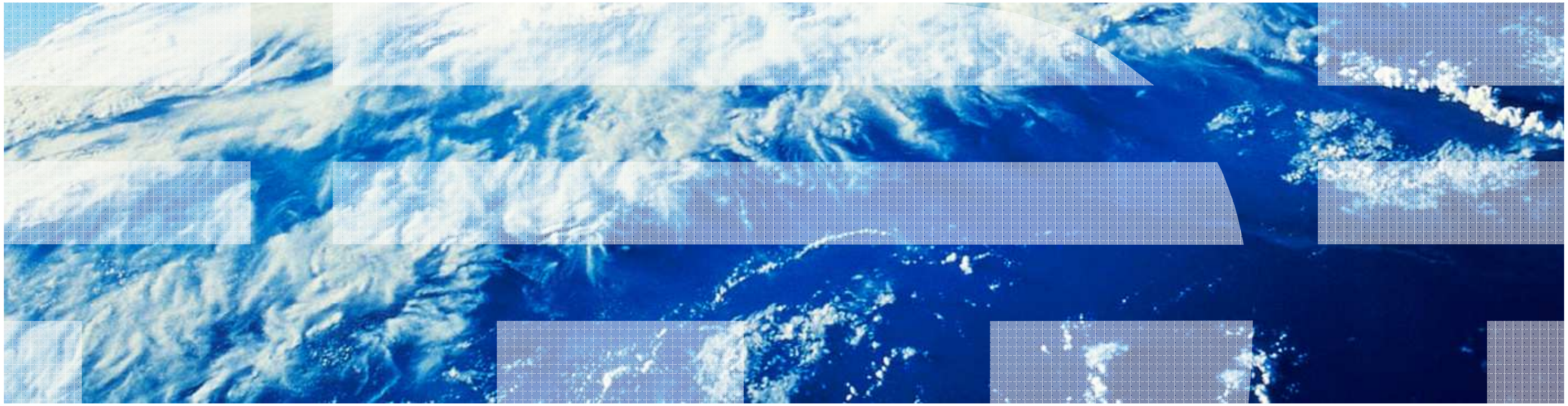
Tomáš Müller – IT Architekt

tomas_muller@cz.ibm.com

22/11/2010

FI MU:
SCA and SDO



Agenda

- A brief history of SOA
- Why SCA makes life simpler
- Composing and assembling SCA applications
- Code and other details
- Customer scenarios
- Service Data Objects
- Resources

Agenda

- **A brief history of SOA**
- Why SCA makes life simpler
- Composing and assembling SCA applications
- Code and other details
- Customer scenarios
- Service Data Objects
- Resources

A brief history of SOA

- When we started with Web services (SOAP over HTTP), we used XML to move data the idea was to send XML to a URL, invoking a service synchronously.
- Things have gotten more complicated since then:
  - Protocols other than HTTP
  - Document-style SOAP services instead of RPC
  - Asynchronous invocation with JMS
  - Encryption, conversations, reliable messaging, WS-*
  - Etc.

A component

- When dealing with a component (in an SOA or not), there are three important pieces of information:
  - The **interface** of the component
  - The **implementation** of the component
  - The **access method** to invoke the component
- We’ll consider how we use this information to invoke components.

The bad old days

- Originally, most components were hardwired into an application:
  - The application knew the details of the component's interface at build time.
  - The application accessed the component's implementation at build time.
  - The application knew the details of the component's access method at build time.
- This worked (and still does), but the application is relatively brittle.
  - If the implementation or access method changes, we have to modify our code, rebuild it, retest it and redeploy it.

The early days of Web services

- SOAP introduced a way to invoke a remote service with an XML envelope.
- The SOAP infrastructure built the envelope and sent it to a particular URL; the SOAP service's host invoked a service and sent XML back to us.
  - The application knew the details of the component's interface at build time.
  - ***The application did not access the component's implementation at build time; the component is invoked at run time by the SOAP infrastructure.***
  - The application knew the details of the component's access method at build time (usually SOAP/HTTP).

Next-generation SOA with SCA

- An SCA application is even more dynamic:
  - The application knows the details of the component's interface at build time.
  - ***The application does not access the component's implementation at build time; the component is invoked by the SCA invocation framework.***
  - ***The application does not know the details of the component's access method at build time; this is also handled by the SCA invocation framework.***

osoa.org

- SCA and SDO were developed by the Open Service Oriented Architecture group (**`osoa.org`**):

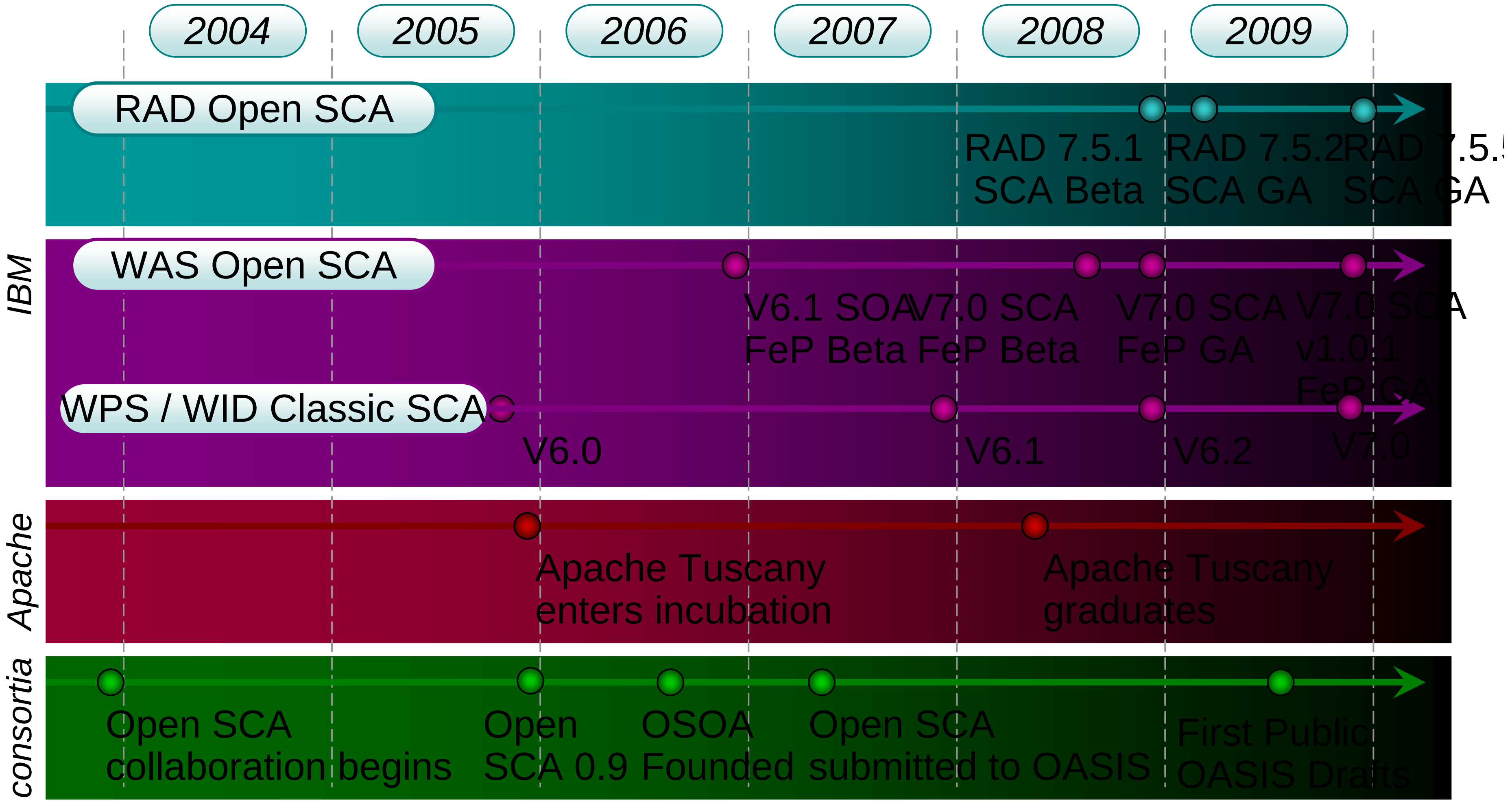
bea Think liquid.
CapeClear
IBM
INTERFACE21
IONA
ORACLE
PRIMETON
PROGRESS SOFTWARE
redhat
ROGUE WAVE SOFTWARE A QUOVADX DIVISION
SAP
software AG
Sun microsystems
SYBASE
TIBCO The Power of Now
xcalia Service Your Data
Zend The php Company

OASIS Open CSA

- The specifications work of **`osoa.org`** has been turned over to OASIS.
- The Open Composite Services Architecture group is being formed now.
  - See **`oasis-opencsa.org`** for more details.
  - *Yes, the **SCA** work has moved to a group named **CSA**.*

  - *http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications*

A Brief History of SCA



Agenda

- A brief history of SOA
- **Why SCA makes life simpler**
- Composing and assembling SCA applications
- Code and other details
- Customer scenarios
- Service Data Objects
- Resources

Why SCA matters

- SCA gives your developers a **single programming model for using services**.
- As your SOA gets more complicated, your developers have to learn more and more interfaces.
  - In Java alone, you might have EJBs, RMI, JCA, JAX-WS or JAX-RPC.
- Similarly, SDO gives your developers a **single programming model for using data sources**.

Why SCA matters

You're committed to SOA, but…	**SCA solves these problems:**
It's not convenient to convert everything to a Web service.	You can integrate many kinds of components, not just Web services.
You want to minimize the learning curve for your developers.	Your developers don't have to learn the details of each component, they just connect them without learning a new API.
As you have more components and data sources, you'll want to rewire your applications more often.	When you integrate an SCA component or an SDO data source, you can replace the component/data source without changing your code.
Your developers don't understand how to exploit the power of an SOA.	Your developers focus on reusable business logic. SCA provides the SOA model and hides the middleware complexity from them.

What SCA *is*

- An **executable** model for assembling services
- A simplified **component programming model** for implementing services
  - Write ‘em as BPEL processes, Java POJOs, EJBs, COBOL apps, PHP scripts, C++ apps…
- We won’t focus on this today, but an SCA composite definition includes all of the services that our composite depends upon.
  - Dependency management is much simpler.

What SCA *isn’t*

- A workflow model
  - Use BPEL for that
- Web services
  - Many SCA implementations will use Web services, but you can create SCA solutions with no Web services content
- Tied to a specific programming language, protocol, technology, runtime, etc.

The SCA specs

- There are four parts to the specs:
  - The **Assembly Model**

    How to define composite applications
  - The **Client and Implementation** specifications

    Java, C++, BPEL
  - **Binding** specifications

    How to use access methods – Web services, JMS, RMI-IIOP, REST…
  - **Policy Framework**

    How to add security, transactions, conversations, reliable messaging, etc. *declaratively*

Agenda

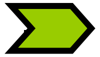
- A brief history of SOA
- Why SCA makes life simpler
- **Composing and assembling SCA applications**
- Code and other details
- Customer scenarios
- Service Data Objects
- Resources

Composition and assembly

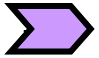
- How do you package a service so it can be integrated with other services?
- SCA has a consistent model:
  - A simple service in SCA is called a ***component***.
  - Components can be grouped into ***composites***.
  - Components and composites are hooked together with ***wires***.
- We’ll use SCA diagrams to illustrate these concepts.
- All of the definitions and configuration are done in XML.

Symbols in SCA assembly diagrams

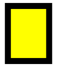
- Here are the symbols used in SCA assembly diagrams:



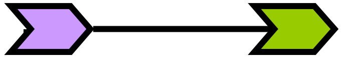
A **green chevron** represents a **service**. This is an entry point to the SCA component or composite.



A **purple chevron** represents a **reference**. This points to a service provided by something else.



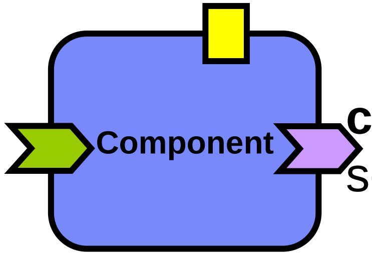
A **yellow rectangle** represents a **property**. This is a value you can set when you invoke the component or composite.



A **line** represents a **wire**. This is the connection between a service reference and the service itself.

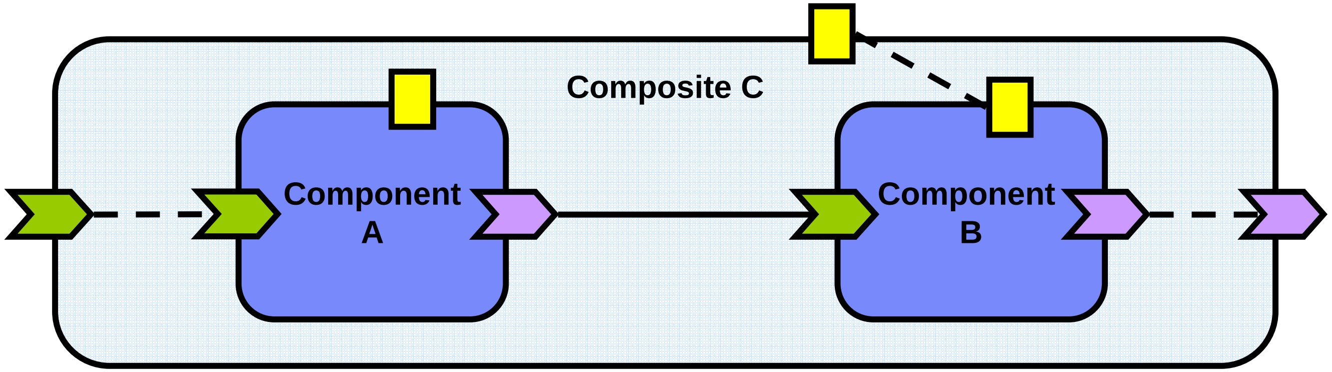
SCA symbols

- More symbols:

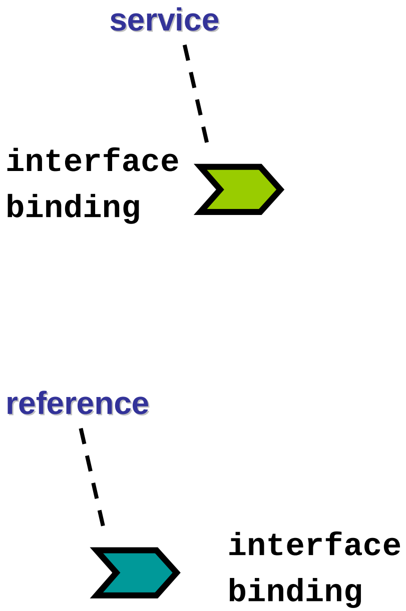


A **rounded rectangle** represents a **component**. A component can have services, references and properties.

A **large rounded rectangle** represents a **composite**. A composite contains one or more components. Like a component, it can have services, references and properties. A composite can also contain a composite.

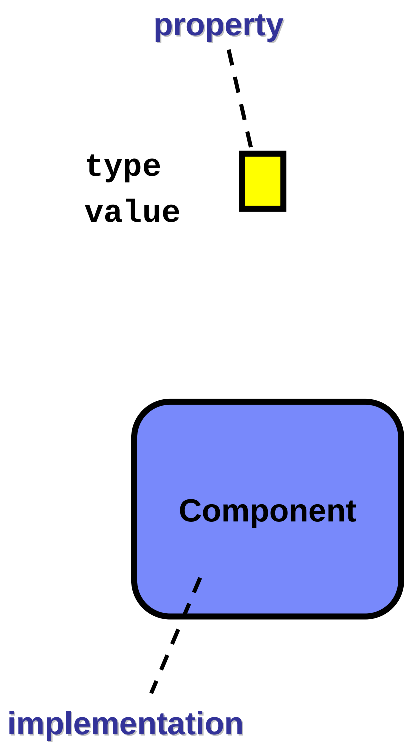


Services and references



- A service or a reference has an **interface** and a **binding**.
- The interface might be a Java interface, a WSDL port type, a BPEL partner link, a C++ class, etc.
- The binding defines the access method. It might be SOAP/HTTP, JMS, JSON, RMI-IIOP, SCA, etc.

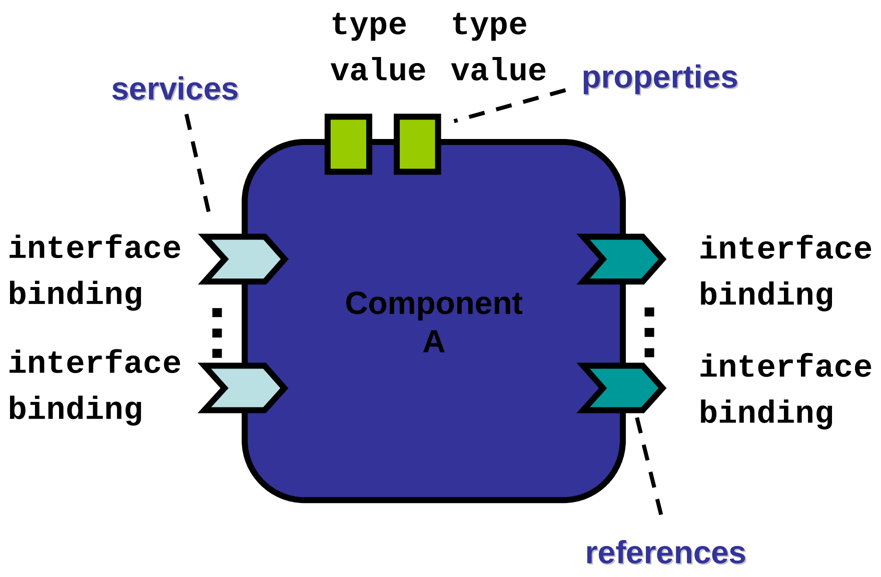
Properties and implementations



- A **property** has a type and a value.
- A component has an **implementation**; that’s the code that actually provides the service.
- The implementation might be BPEL, Java, C++, Spring, etc.

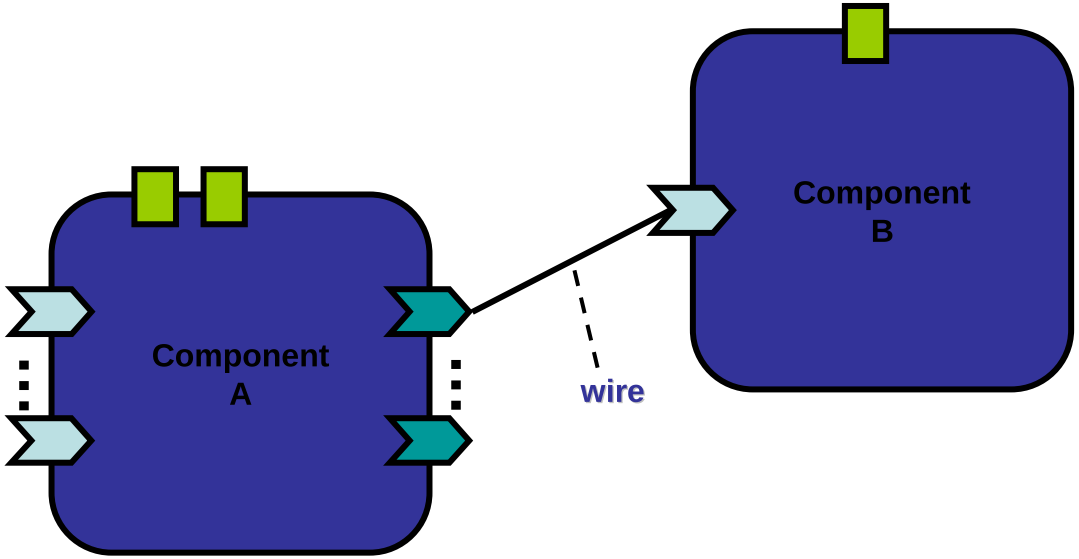
A component

- This diagram is a component with services, references and properties.



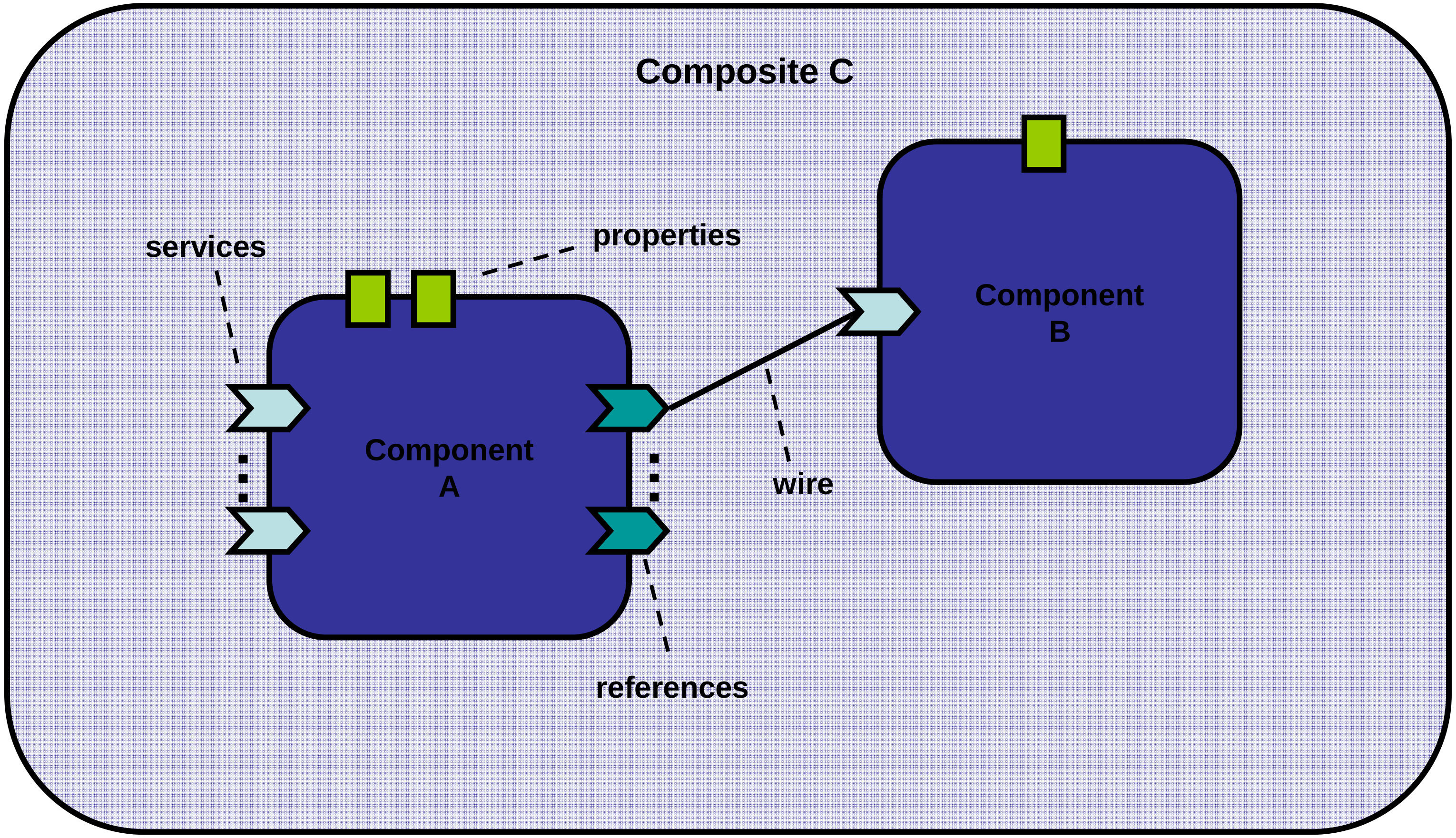
Wiring

- Here are two components wired together:

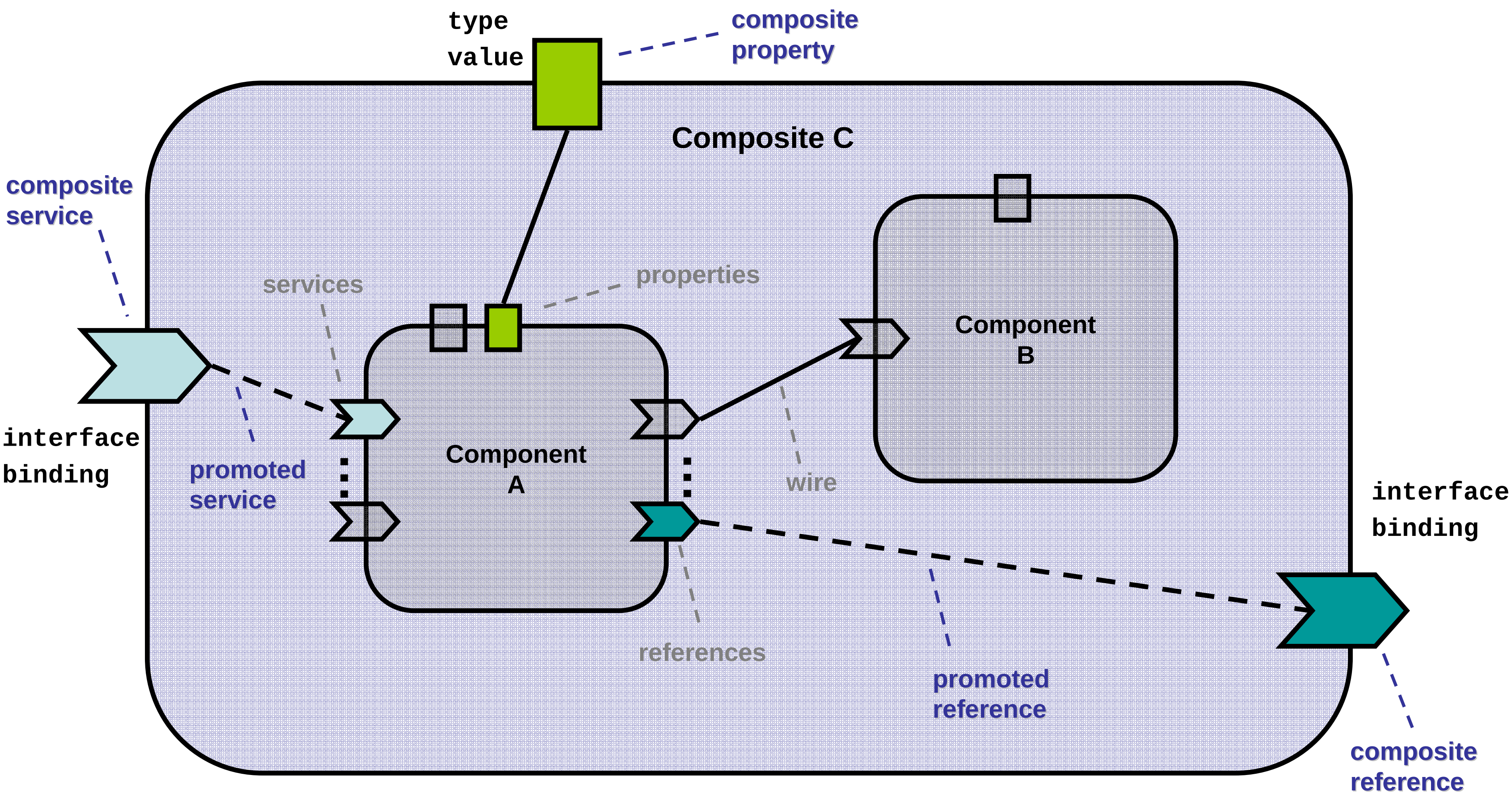


A composite

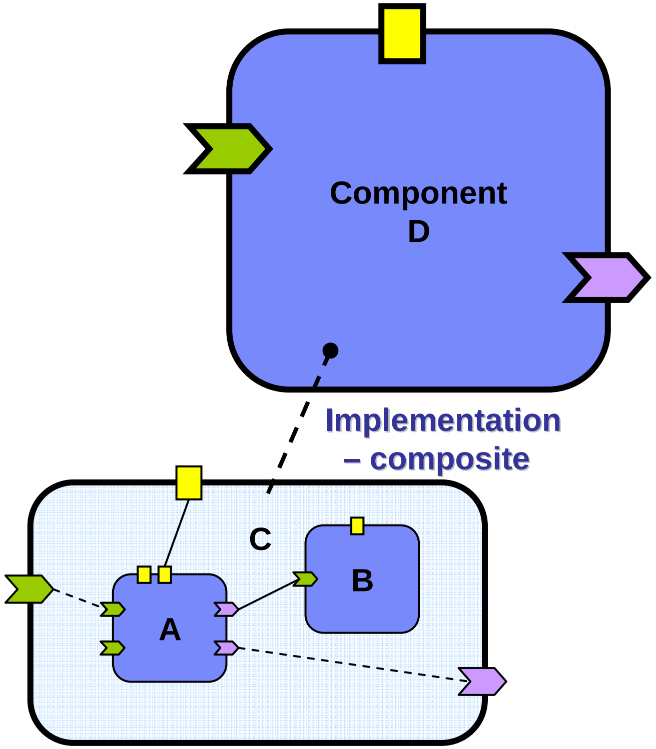
- Here are two components grouped together in a composite:



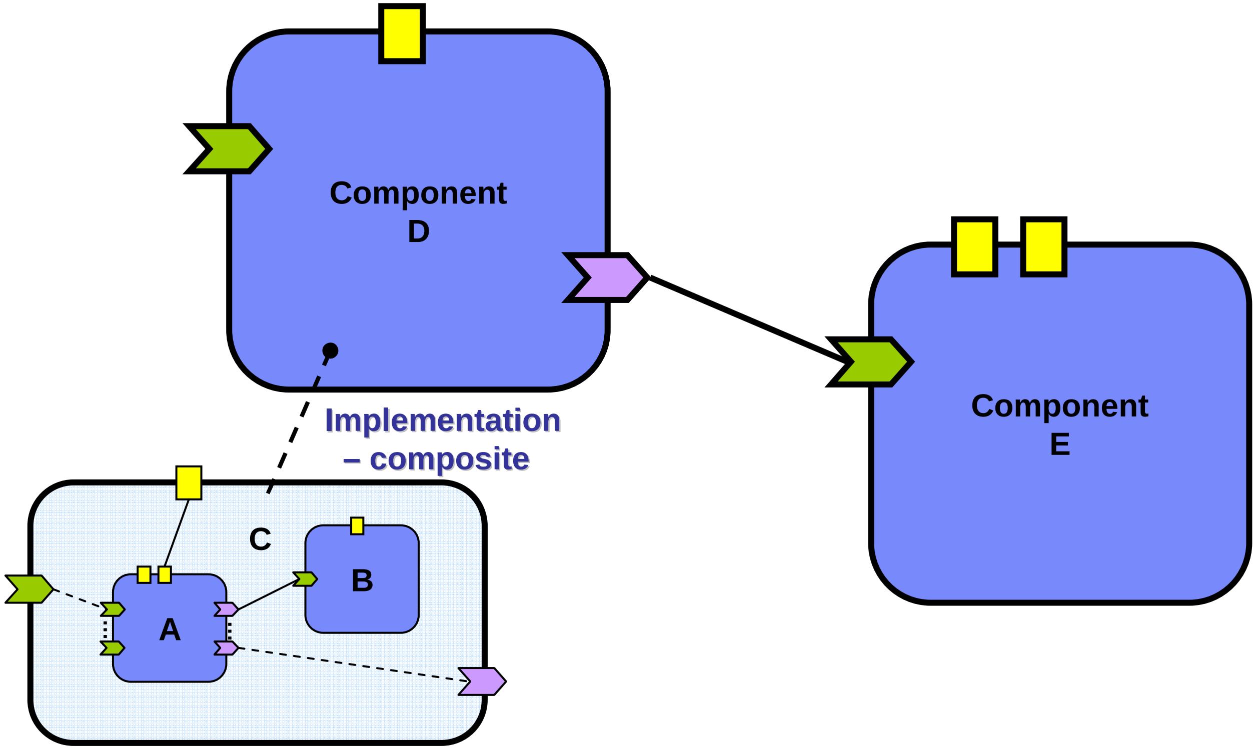
Promotion


type
value
composite
property
Composite C
composite
service
services
properties
Component
B
interface
binding
promoted
service
Component
A
wire
interface
binding
references
promoted
reference
composite
reference

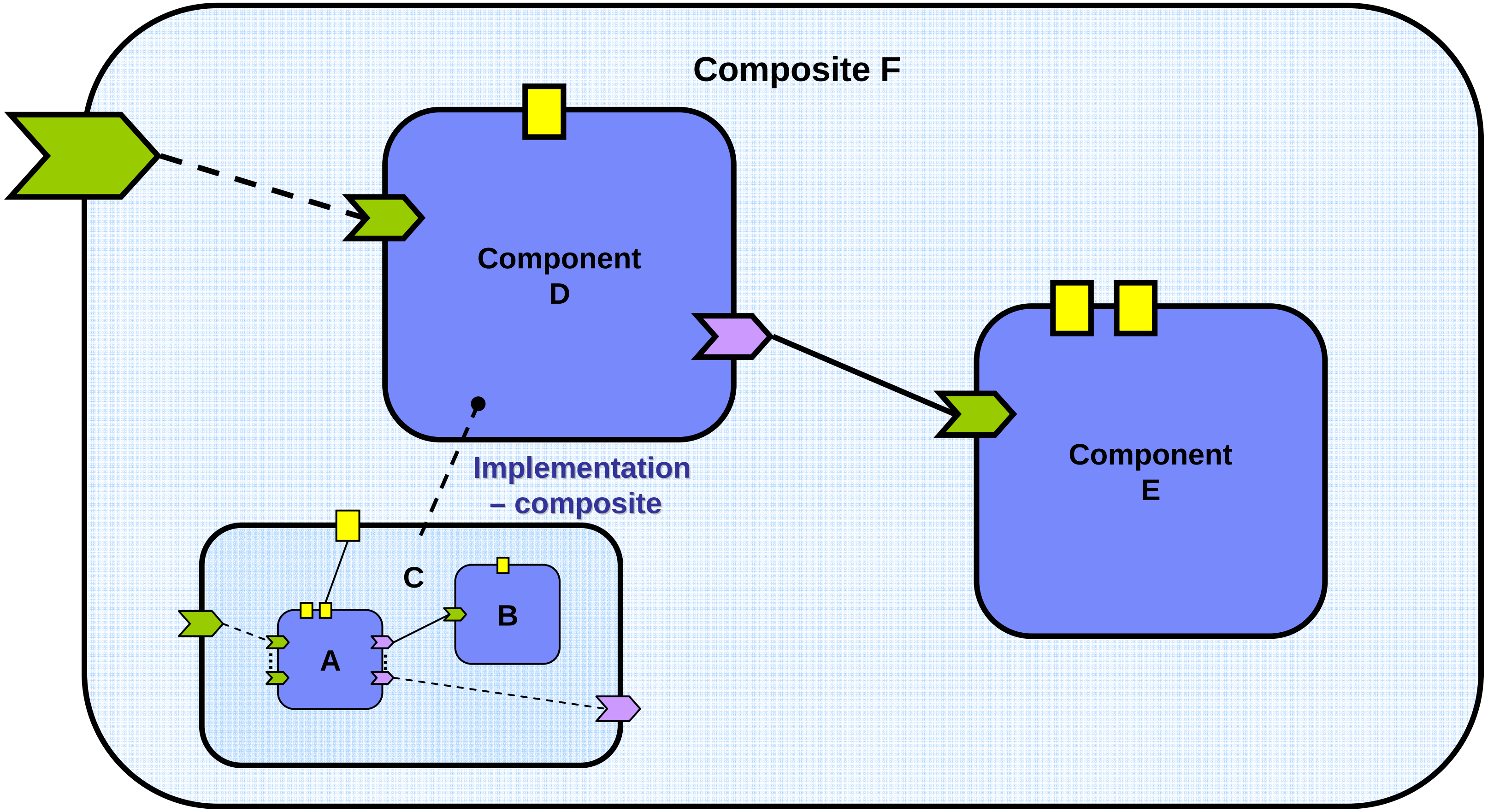
A composite implementation



A composite using another component



A composite that includes a composite

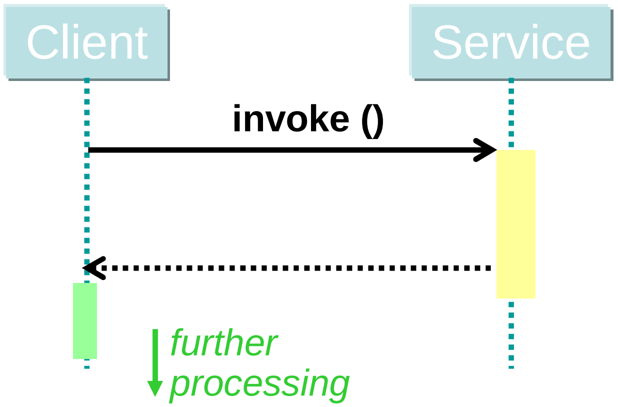


Synchronous / Asynchronous Model

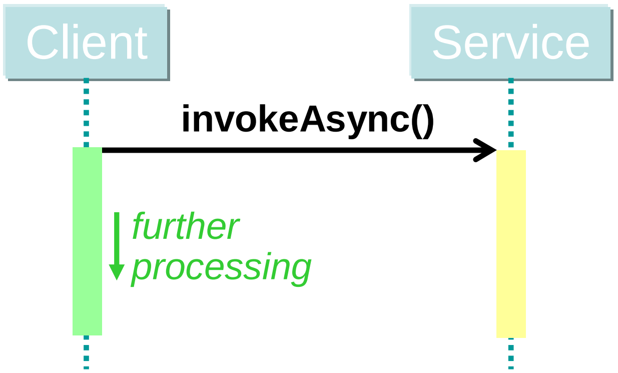
- SCA provides the ability for services to be called synchronously or asynchronously
  - Synchronous Model
    - Blocking

      Client waits for a response
  - Asynchronous Model
    - Non-Blocking

      Client doesn't wait for a response



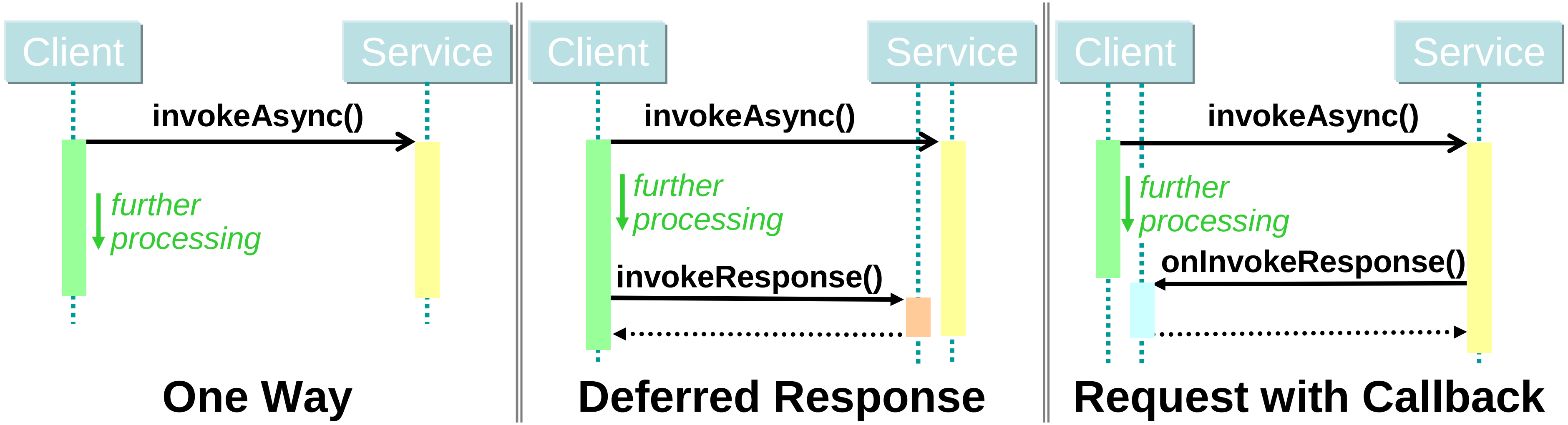
Synchronous Model



Asynchronous Model

Asynchronous Model

- There 3 types of asynchronous invocation models



One Way

Deferred Response

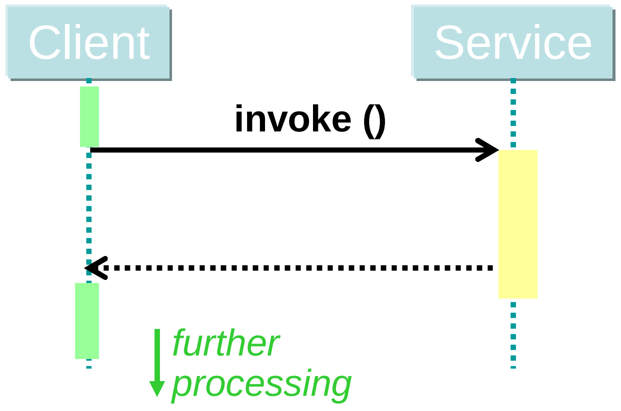
Request with Callback

Asynchronous Model

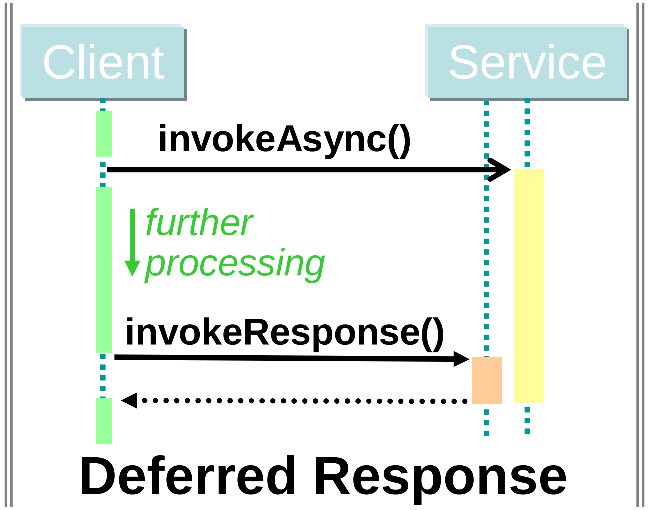
- Synchronous vs. Pseudo Synchronous

- Synchronous Model
  - Blocking

    Client waits for a response

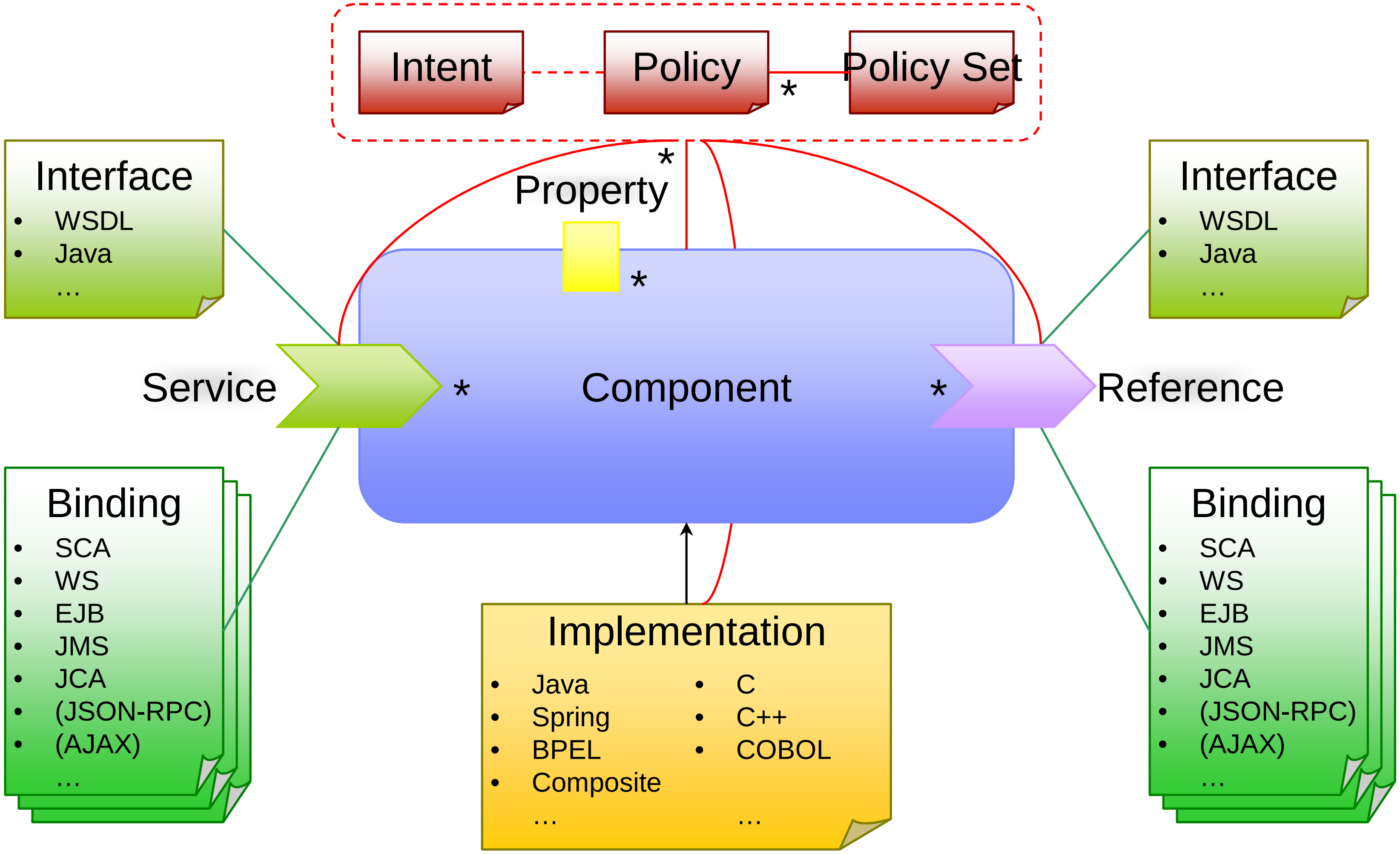


Synchronous Model



Deferred Response

SCA Summary


Intent
Policy
*
Policy Set
Property
*
*
Interface
- WSDL
- Java
…
Service
*
Component
*
Reference
Interface
- WSDL
- Java
…
Binding
- SCA
- WS
- EJB
- JMS
- JCA
- (JSON-RPC)
- (AJAX)
…
Implementation
- Java
- Spring
- BPEL
- Composite
…
- C
- C++
- COBOL
…
Binding
- SCA
- WS
- EJB
- JMS
- JCA
- (JSON-RPC)
- (AJAX)
…

Agenda

- A brief history of SOA
- Why SCA makes life simpler
- Composing and assembling SCA applications
- **Code and other details**
- Customer scenarios
- Service Data Objects
- Resources

An SCA client

- Here’s an SCA client that uses **SOAP/HTTP**:

```
public class CalculatorClient {
  public static void main… {
    SCADomain scaDomain =
      SCADomain.newInstance("calc.composite");
    CalculatorService calcServ =
    scaDomain.getService(CalculatorService.class,
        "CalculatorServiceComponent");
    System…println(calcServ.add(3,2));
    scaDomain.close();
  }
}
```

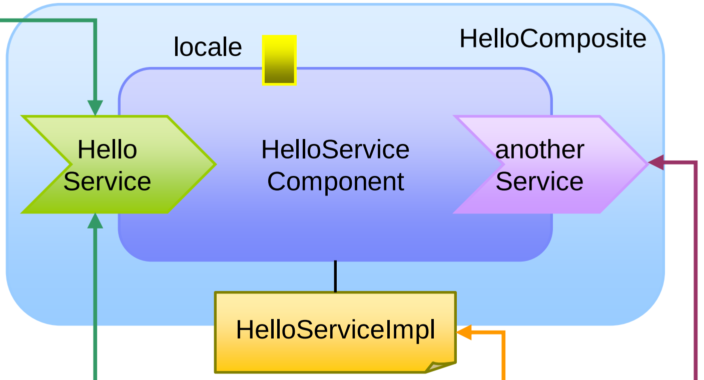
An SCA client

- Here’s an SCA client that uses **RMI**:

```
public class CalculatorClient {
  public static void main… {
    SCADomain scaDomain =
      SCADomain.newInstance("calc.composite");
    CalculatorService calcServ =
    scaDomain.getService(CalculatorService.class,
        "CalculatorServiceComponent");
    System…println(calcServ.add(3,2));
    scaDomain.close();
  }
}
```

SCA Annotated Java

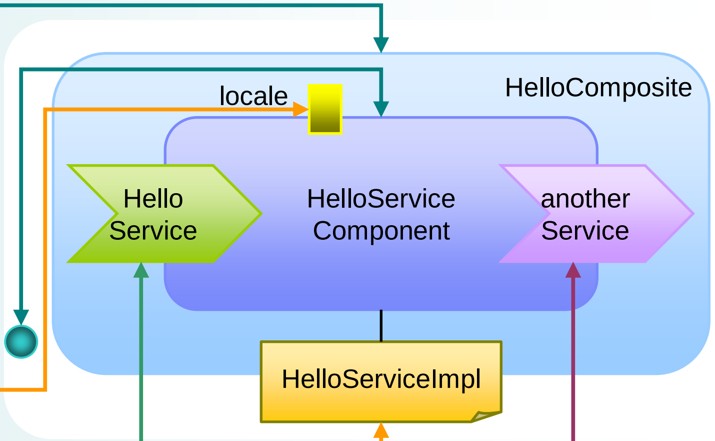
```
package service;
@Remotable
public interface HelloService
{
    String hello ( String message );
}
```



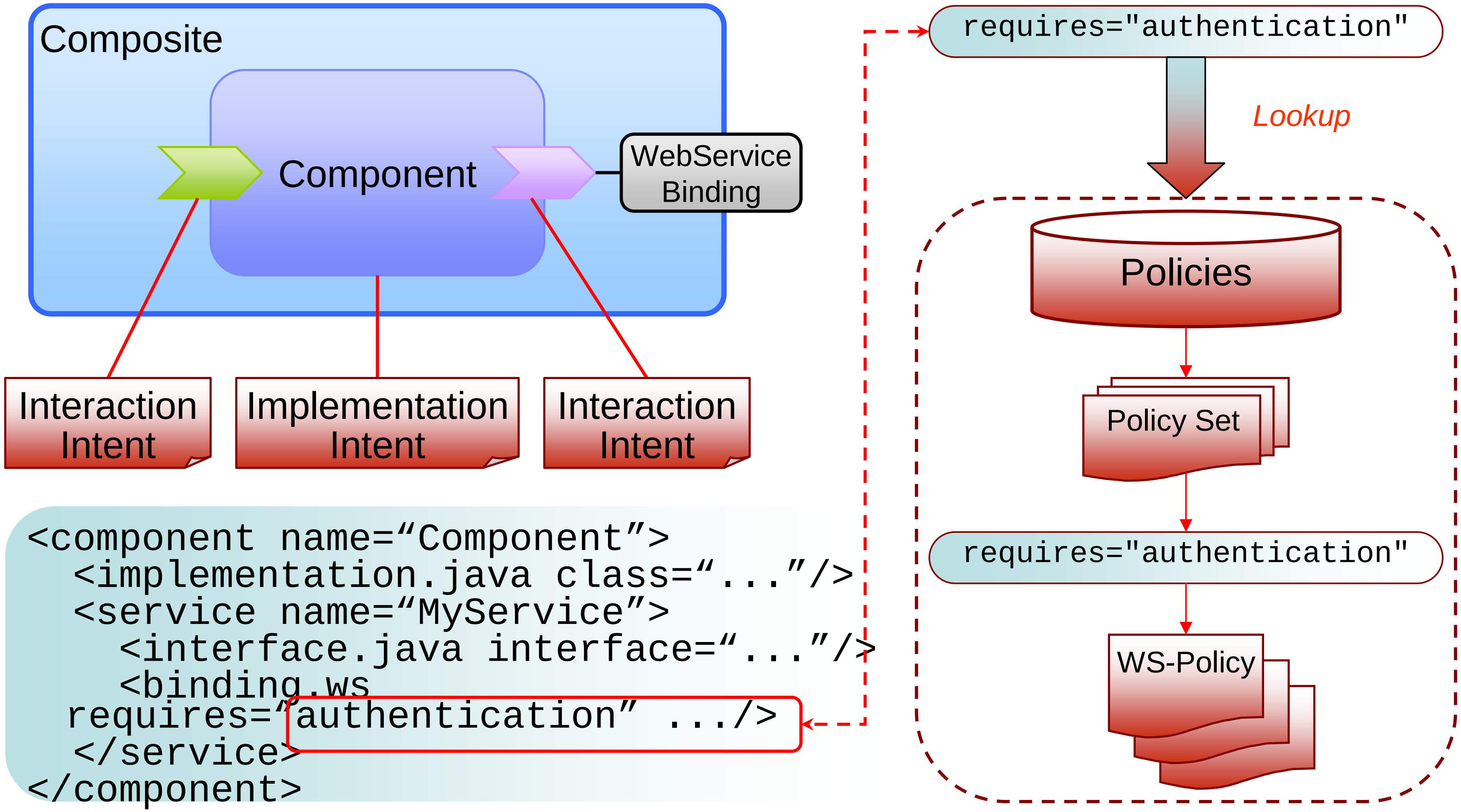
```
package service;
@Service ( HelloService.class )
public class HelloServiceImpl   implements HelloService
{
    @Reference public AnotherService anotherService;

    String hello ( String message ) {
        return anotherService.howdy(message);
    }
}
```

SCA Composite XML

```
<?xml version="1.0" encoding="ASCII"?>
<composite name=“HelloComposite"
 xmlns="http://www.osoa.org/xmlns/sca/1.0"
 targetNamespace="http://foo.com">
    <component name=“HelloServiceComponent">
        <property name=“locale">
            ...
        </property>
        <service name=“HelloService“>
            ...
        </service>
        <reference name=“anotherService“>
            ...
        </reference>
        <implementation.java class="services.HelloServiceImpl"/>
    </component>
</composite>
```



Qualities of Service – Intents and Policies



```
<component name=“Component”>
  <implementation.java class=“...”/>
  <service name=“MyService”>
    <interface.java interface=“...”/>
    <binding.ws
   requires=“authentication” .../>
  </service>
</component>
```

Bindings

- In SCA, a ***binding*** specifies how to access a service.
  - Current bindings include WSDL, JMS, JCA and EJBs.
  - More bindings are coming all the time at **`osoa.org`**.
  - Like all of SCA, the binding specification is open, so you can create your own.
- Add asynchronous support, conversational support, etc. declaratively:
  - **`<interface.java interface="..." callbackInterface="...InvoiceCallback"/>`**
  - **`sca:requires="conversational"`**
  - **`sca:endsConversation="true"`**

Policies

- Previous standards efforts, WSDL in particular, didn’t include how to define general ***policies*** for services.
  - Nowadays is situation better due to WS-Policy, WS-PolicyAttachment

- SCA gives you a single declarative way to establish policies.
  - “This component must provide this level of QoS.”
  - “All traffic on this wire must be digitally signed.”

What’s in the WSDL file?

- The data structures
  - *Defined with XML Schema*
- The interface
  - *There’s a method called* ***`getStockQuote`****, it takes a string as input and returns a string as output*
- The binding(s)
  - *SOAP over HTTP*
- The endpoint(s)
  - ***`http://xyz.com:8080/myService`***
- Ideally the bindings and endpoints are in a separate file.
  - Unfortunately, that doesn’t always happen.

What can I do with the WSDL file?

- Send a SOAP envelope to a particular service over a particular protocol.
  - That's it.
- *A Service-Oriented Architecture needs a far more sophisticated way of working with services.*

Doing more sophisticated things

- “Everyone using this service must be authenticated.”
  - **The WSDL file won’t tell you that.**
- “Every request sent to this service must be digitally signed.”
  - **The WSDL file won’t tell you that.**
- “Any message sent to this service is guaranteed to be delivered.”
  - **The WSDL file won’t tell you that.**
- “Particular service ensures specific QoS.”
  - **The WSDL file won’t tell you that.**
- “Every message sent to this service must be encrypted.”
  - **The WSDL file won’t tell you that.**
- “Every request to this service …”
  - ***SCA tells you everything you need to know.***

The problem with the missing stuff

- Without this information, **nothing works.**
  - That’s a big problem.

- **SCA solves this problem in an elegant way.**
  - The details we just mentioned are handled by the SCA runtime.
  - Those details can be changed without any changes to the client application or the service.

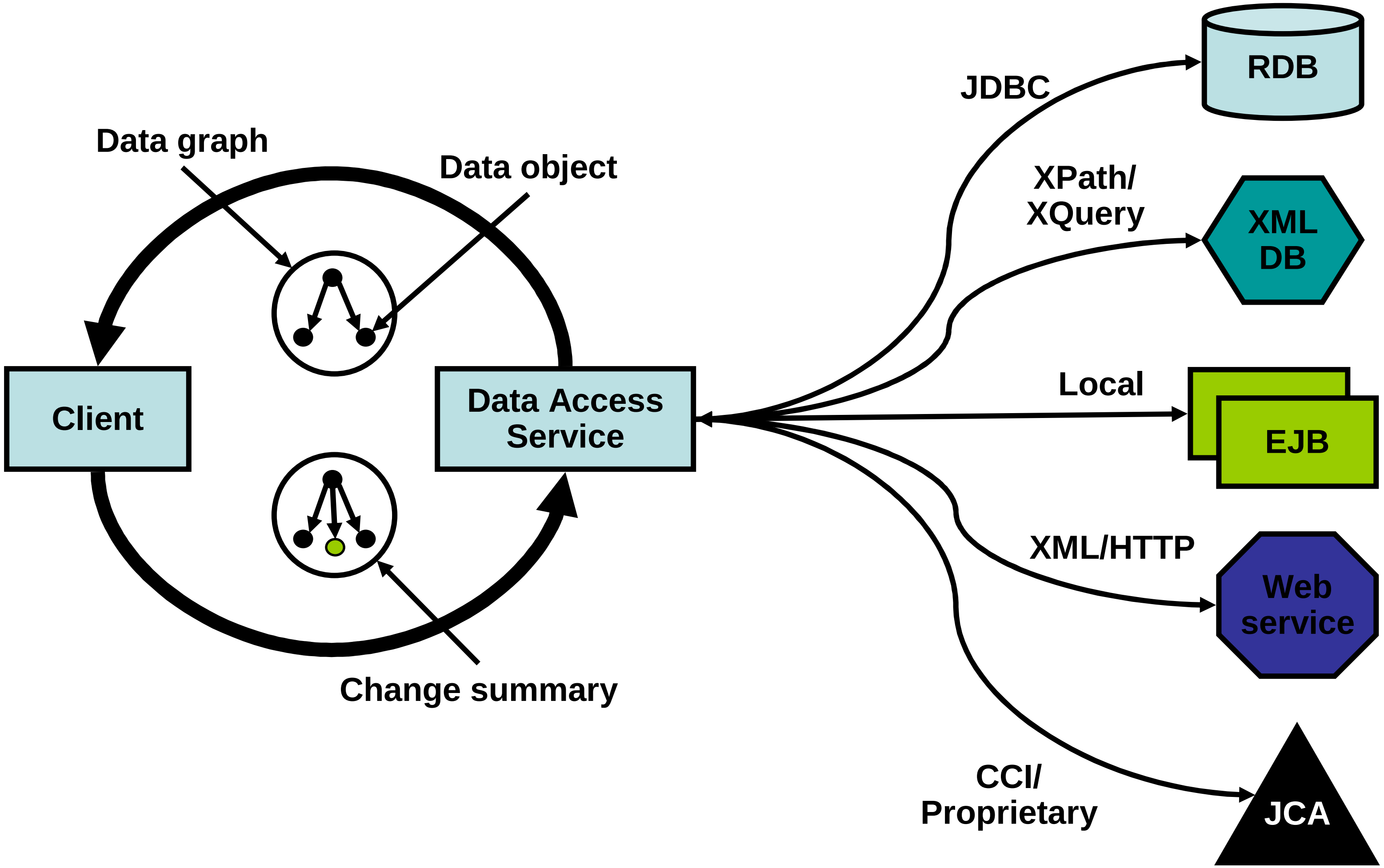
Agenda

- A brief history of SOA
- Why SCA makes life simpler
- Composing and assembling SCA applications
- Code and other details
- **Service Data Objects**
- Resources

Service Data Objects

- SDO gives you a single API to a wide variety of data sources.
- You and I as developers focus on CRUD operations, we don’t know or care what the data source actually is.
  - Relational database
  - XML database or XML file
  - EJB
  - Web service
  - JCA

A disconnected interface to many kinds of data sources

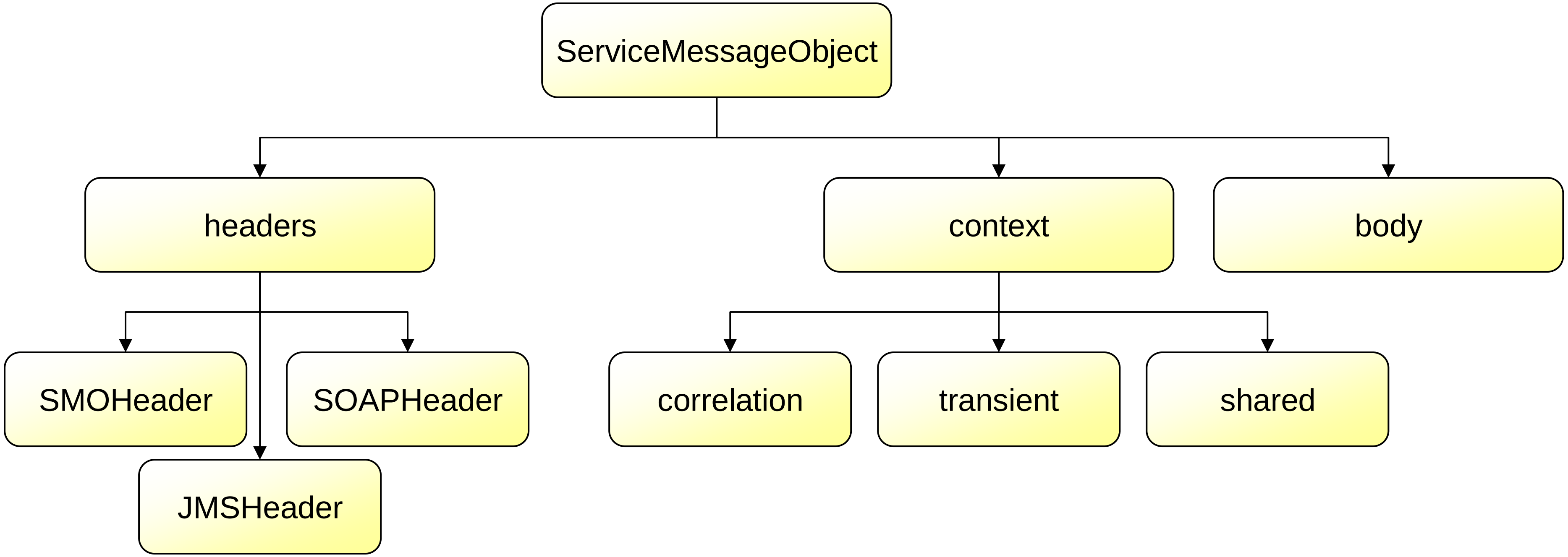


Accessing data in SDO

- I have some data.
- I use the data wherever and however it’s stored (RDBMS, XML file, LDAP, etc.)
- I use the most convenient language for CRUD operations on the data (SQL, XQuery, modified XPath, etc.)

Service Message Object

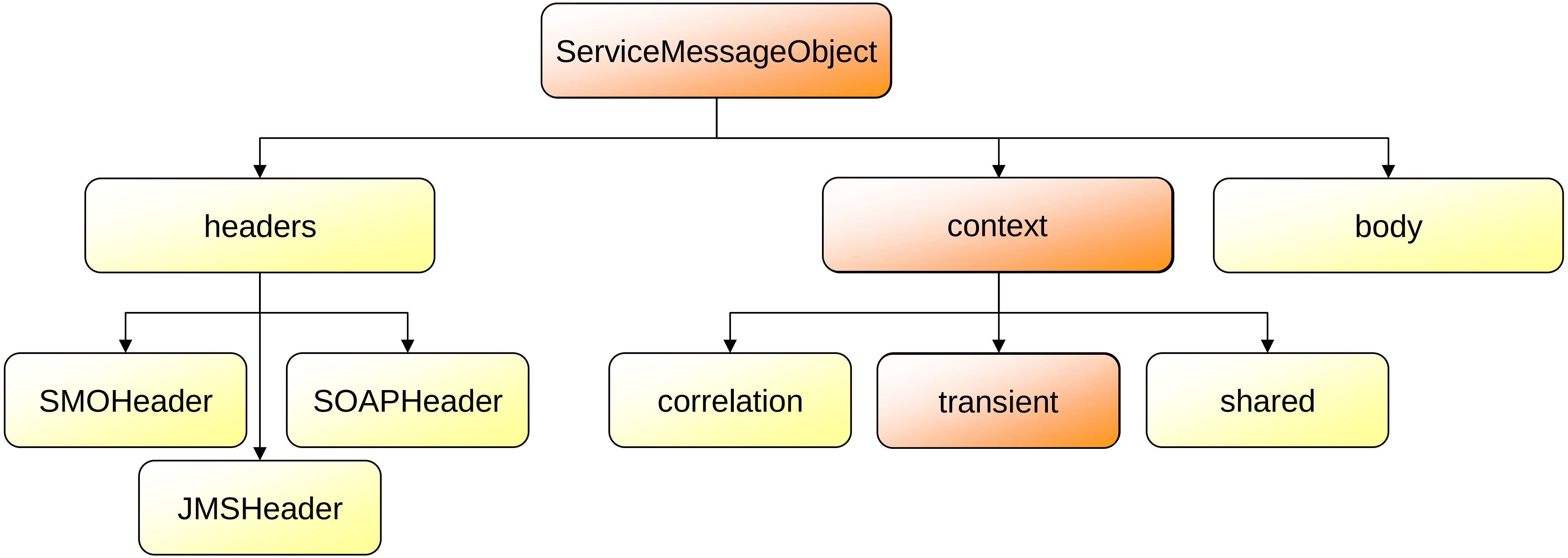
- Mediation primitives process messages as SMOs
- The SMO is an extension of the Business Object (BO) structure
- It contains: context, message headers, fault details, an array of properties and payload information



Service Message Object API

- Accessible by API (com.ibm.websphere.sibx.smo.*)

```
ServiceMessageObject smo = (ServiceMessageObject)a_type;
DataObject context = smo.getContext()
DataObject transient = context.getTransientContext();
```



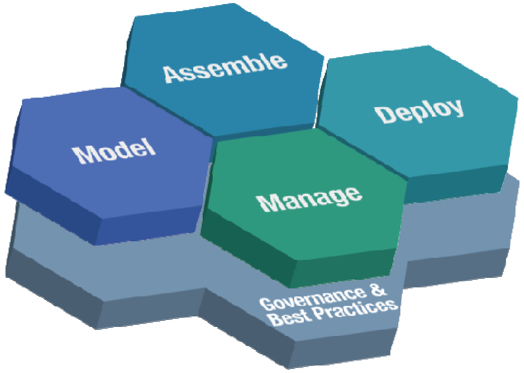
SMO – some Java APIs

```
import com.ibm.websphere.sibx.smobo.ContextType;
import com.ibm.websphere.sibx.smobo.HeadersType;
import commonj.sdo.DataObject;

public DataObject execute(DataObject a_type) {
 ContextType context = (ContextType) a_type.get("/context");
 HeadersType headers = (HeadersType) a_type.get("/headers");
 DataObject body = (DataObject) a_type.get("/body");
 return a_type;
}
```

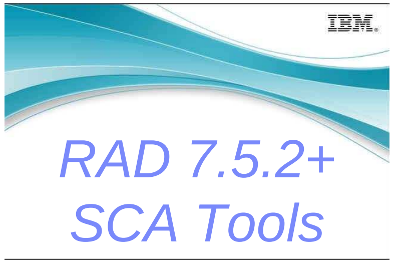
Open Service Component Architecture (Open SCA)

An open, emerging standard programming model for assembling flexible SOA business solutions from diverse, reusable service enabled IT assets

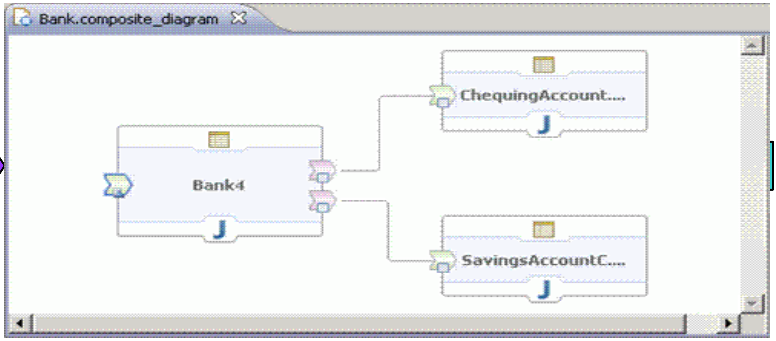
Develop interfaces and implementations. Compose and Wire. Bindings and Intents.



Define, install and run contributions on WebSphere Application Server.



RAD 7.5.2+ SCA Tools



WAS 7 SCA Feature Pack

Agenda

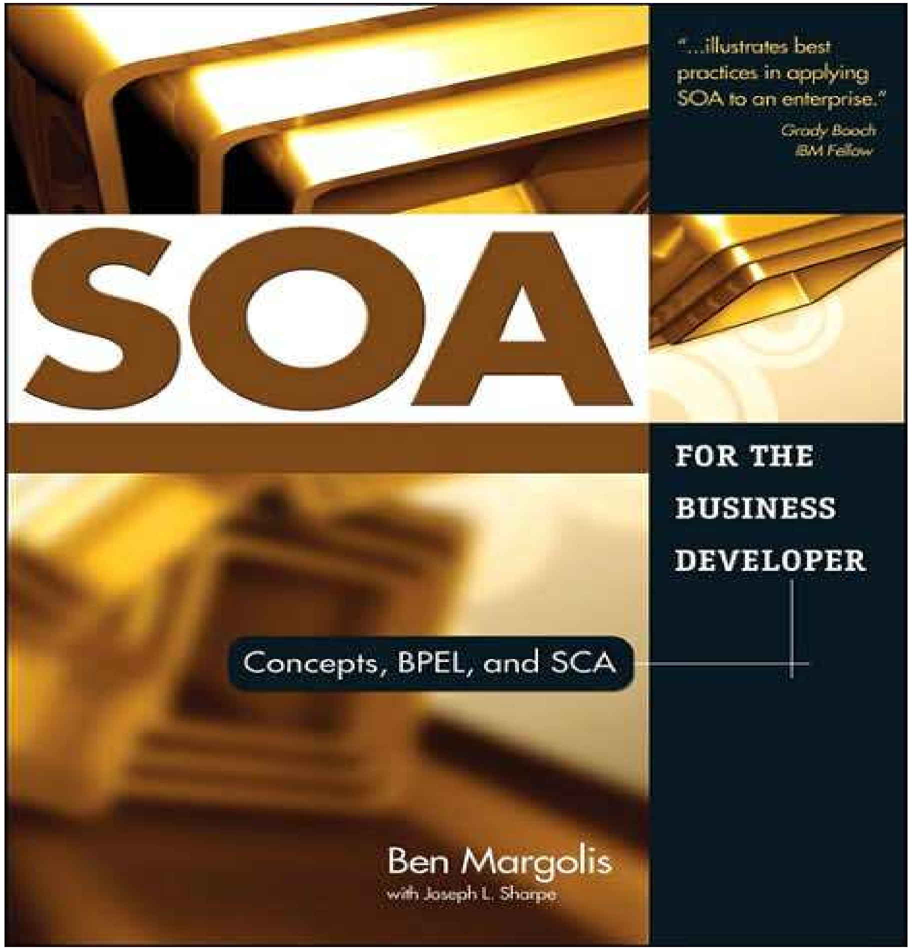
- A brief history of SOA
- Why SCA makes life simpler
- Composing and assembling SCA applications
- Code and other details
- Service Data Objects
- **Resources**

Resources

- The home of everything related to SCA and SDO is **`osoa.org`**.
  - From here you can find the specs, white papers and tutorials.
- The OSOA's work is moving to OASIS. For more information on the Open CSA project, visit **`oasis-opencsa.org`**.
- If you'd like to get involved in the standardization effort, a call for participation has recently been issued.
- The **`oasis-opencsa.org/specifications`** page is another great way to find the specifications and other technical resources.

Books

- Get Ben Margolis' book **SOA for the Business Developer: Concepts, BPEL and SCA**.
- As of August 2007, this is *the best book on SCA*. It covers both the architecture and the technical details.
- Many of the best minds in the SCA world had a hand in this book.
- ISBN 1-58347-065-4


"...illustrates best practices in applying SOA to an enterprise."
Grady Booch
IBM Fellow
SOA
FOR THE BUSINESS DEVELOPER
Concepts, BPEL, and SCA
Ben Margolis
with Joseph L. Sharpe

developerWorks articles

- **An overview of Service Component Architecture** (2 parts):
  - **`ibm.com/developerworks/webservices/library/ ws-soa-scadev1/ and .../ws-soa-scadev2/`**
- **Building SOA solutions with SCA** (4 parts):
  - **`ibm.com/developerworks/websphere/techjournal/0510_brent/0510_brent.html`**
- **Java SCA invocation styles:**
  - **`ibm.com/developerworks/webservices/library/ ws-soa-scajava/`**
- **Using PHP’s SCA and SDO extensions:**
  - **`ibm.com/developerworks/webservices/library/ ws-soa-scasdo/`**

developerWorks **articles**

- **Introduction to Service Data Objects:**
  - `ibm.com/developerworks/java/library/j-sdo/`
- **Build a Web service client with JSF and SDO** (Flash demo):
  - `ibm.com/developerworks/offers/lp/demos/ summary/jsfsdo.html/`
- **SDO 2.0: Create and read an XML document based on an XML Schema:**
  - `ibm.com/developerworks/webservices/library/ ws-sdoxmlschema/`
- More articles on SCA and SDO are coming all the time.