SCA

Assembly of Business Systems using Service Component Architecture

Anish Karmarkar
Oracle Portland

Mike Edwards
IBM Hursley Park

Service Component Architecture (SCA):

A model for the creation of business systems using SOA by the composition and deployment of new and existing service components

- **SCA:Why?**
- **SCA: What?**
- **SCA: How and Where?**
- **Summary**

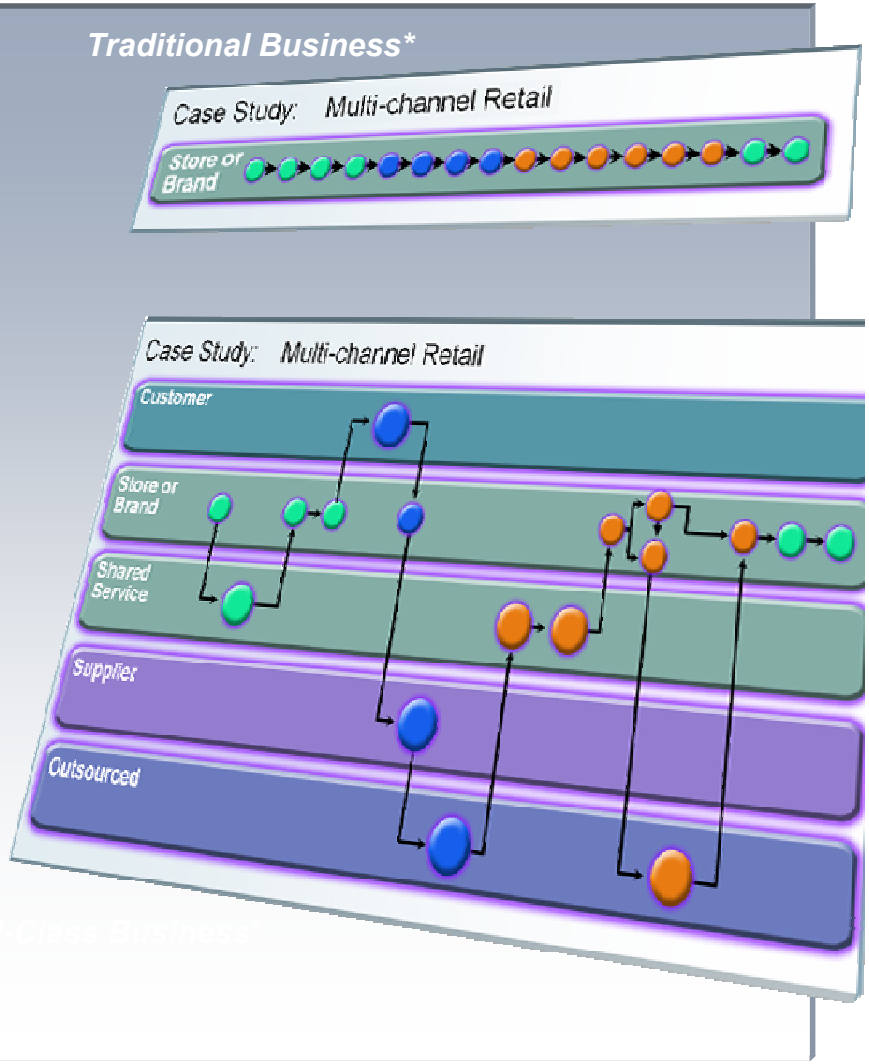
Outline

- **SCA: Why?**
  - **Business Drivers**
  - **What We Have Today**
  - **Where We Want To Get To**
  - **SOA Programming Model**
- **SCA: What?**
- **SCA: How and Where?**
- **Summary**

Business Drivers

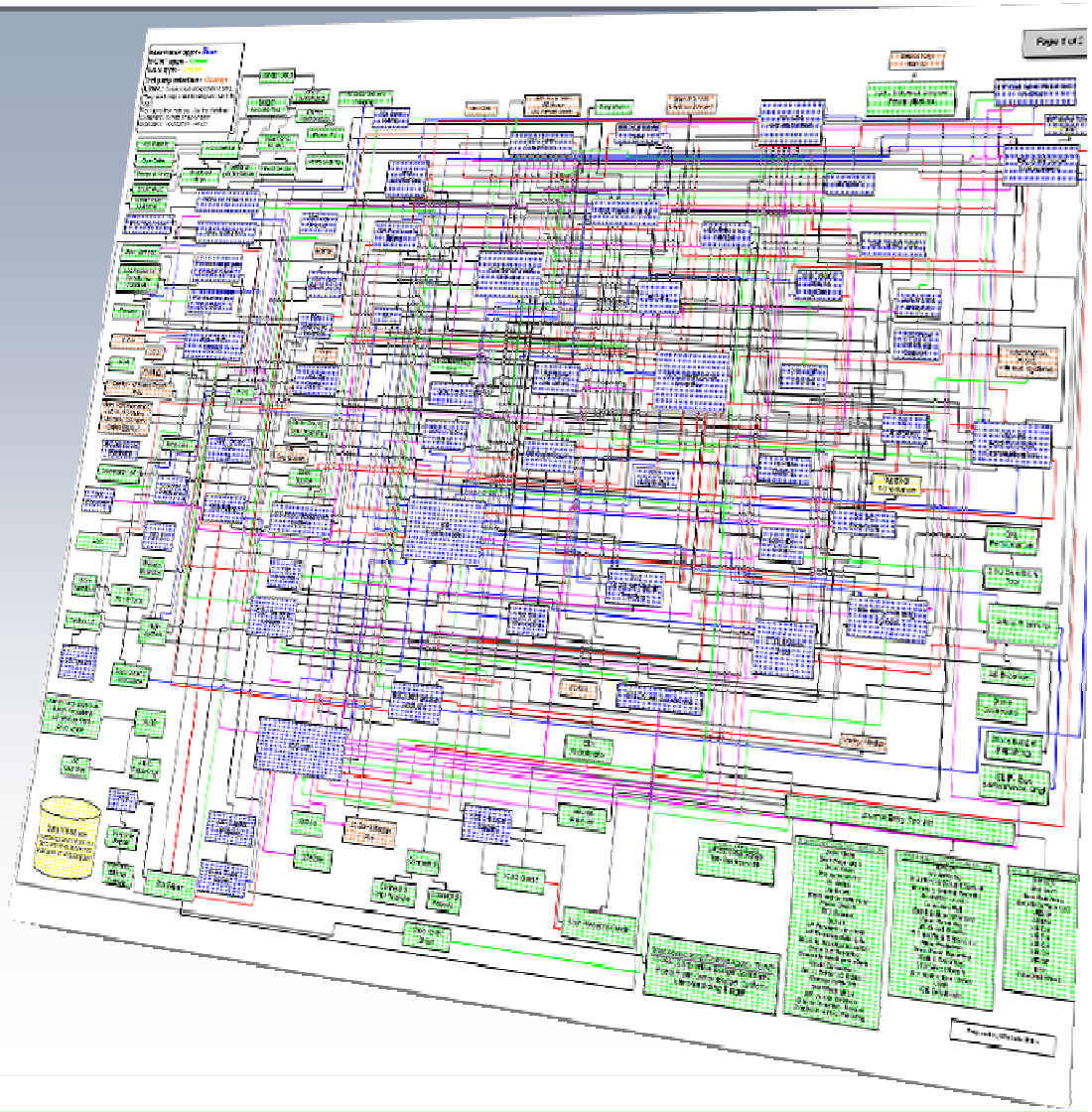
Flexible business requires flexible IT

- **Economics: globalization demands greater flexibility**
- **Global supply chain integration**
- **Business processes: daily changes vs. yearly changes**
- **Growth through flexibility is at the top of the CEO agenda**
- **Reusable assets can cut costs by up to 20%**
- **Crucial for flexibility and becoming an On Demand Business**

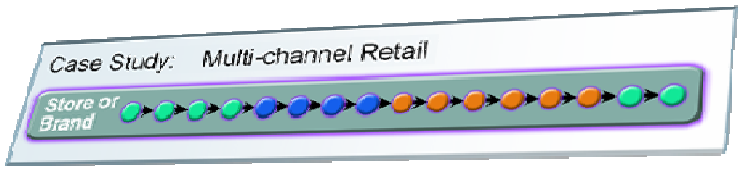


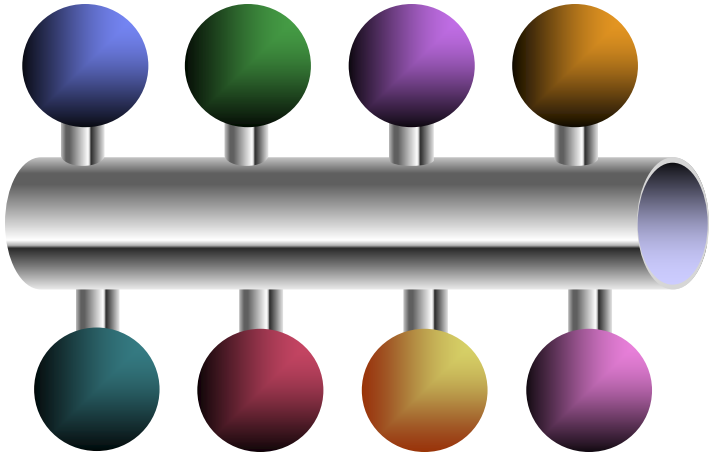
What We have Today

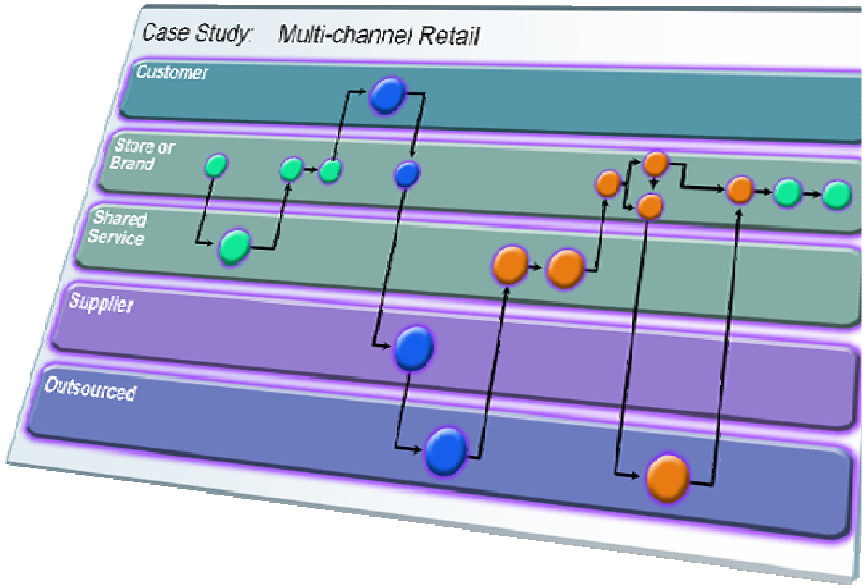
- Complexity
- Rigid, brittle architectures
- Inability to evolve



What we want to get to







- Well-defined interfaces with business-level semantics
- Standardized communication protocols
- Flexible recombination of services to enhance software flexibility

Service-Oriented Architecture is one of the key technologies to enable flexibility and reduce complexity

The SOA Programming Model

- SOA Programming Model derives its technical strategy and vision from the basic concept of a *service*:

  "A service is an abstraction that encapsulates a software function."

  "*Developers* build services, use services and develop solutions that aggregate services."

  "*Composition* of services into integrated *solutions* is a key activity"

  Core Elements

  - **Service Assembly**
    - technology- and language- independent representation of the composition of services into business solutions
  - **Service Component**
    - technology- and language-independent representation of a service which can be composed with other services

Outline

- **SCA: Why?**
- **SCA: What?**
  - **SCA: Simplified Programming Model for SOA**
  - **SCA: High Level View**
  - **SCA Elements**
  - **Assembly Model Concepts**
    - **Component, Service, Composite, Interaction Model, System**
  - **SCA Client and Implementations**
  - **SCA Bindings**
  - **SCA Policy**
- **SCA: How and Where?**
- **Summary**

SCA: Simplified Programming Model for SOA

- What is SCA:
  - ***executable*** model for assembly of service components into business solutions
  - simplified ***component programming model*** for implementation of services:
    - Business services implemented in any of a variety of technologies
      e.g. EJBs, Java POJOs, BPEL process, COBOL, C++, PHP …
- Key Benefits of SCA:
  - ***Loose Coupling***: Components integrate with other components without needing to know how other components are implemented
    - **Loose coupling - KEY requirement for SOA**
  - ***Flexibility***: Components can easily be replaced by other components
    - **Flexibility - KEY requirement for SOA**
  - ***Services*** can be *easily* invoked either synchronously or asynchronously
  - ***Composition*** of solutions: clearly described
    - **Composition of services - KEY requirement for SOA**
  - ***Productivity***: Easier to integrate components to form composite application
- SCA simplifies development experience for **all** developers, integrators and application deployers

SCA: What is it NOT

- Does not model individual ***workflows***
  - use BPEL or other workflow languages
- Is not ***Web services***
  - SCA can use / may use Web services, but can also build solutions with no Web services content
- Is not tied to a specific runtime environment
  - distributed, hetergeneous, large, small
- Does not force use of specific programming languages and technologies
  - aims to encompass many languages, technologies

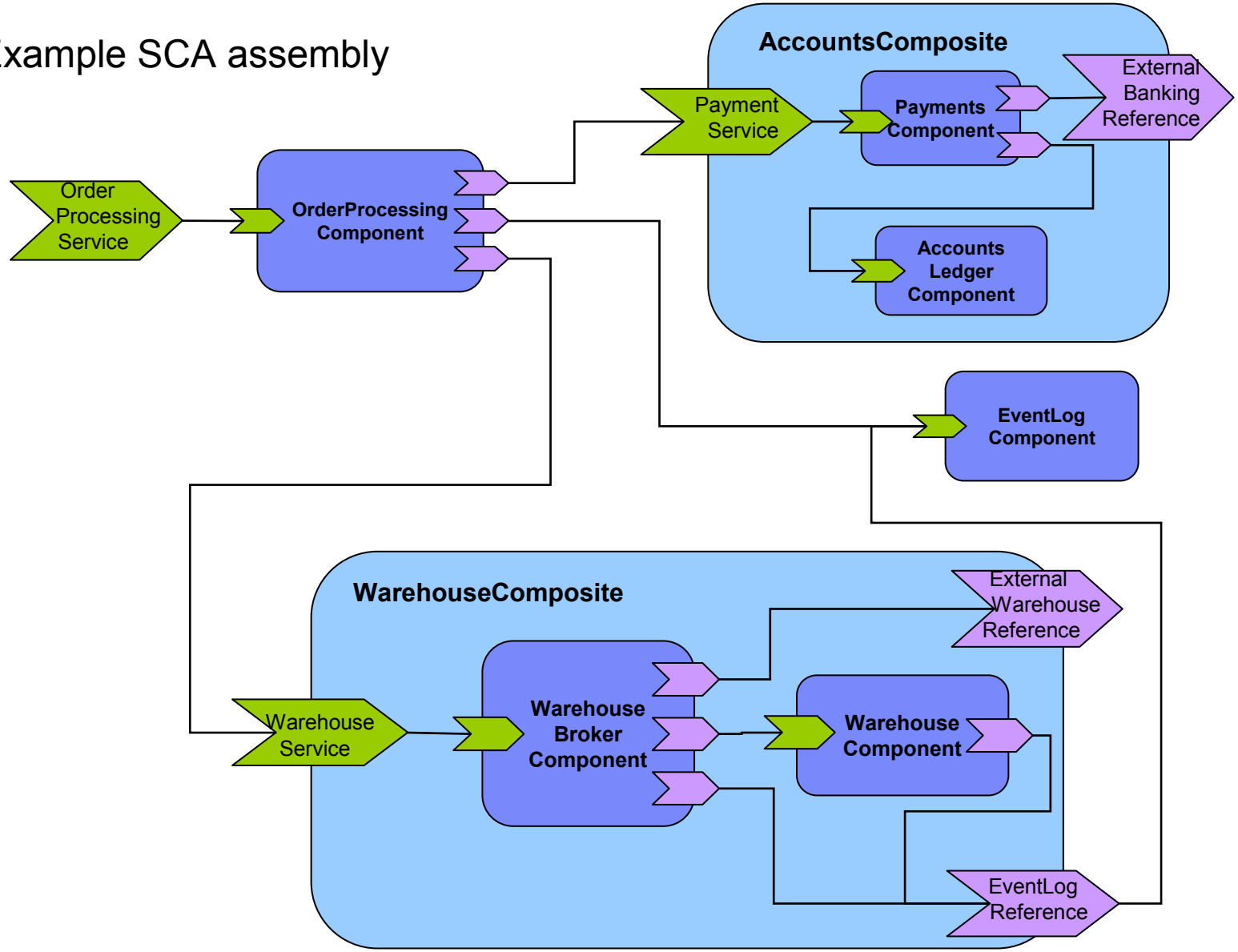
SCA – High Level View

- ***Unified declarative model*** describing service assemblies
  - dependency resolution and configuration
  - declarative policies for infrastructure services
    - Security, Transactions, Reliable messaging
- ***Business-level model*** for implementing services
  - service components with service interfaces
  - no technical APIs like JDBC™, JCA™, JMS™, …
- ***Binding model*** for multiple access methods and infrastructure services
  - WSDL, SOAP over HTTP, JMS™/messaging, Java™ RMI/IIOP…

SCA Elements

- ***Assembly*** Model
  - how to define structure of composite applications
- ***Client & Implementation*** specifications
  - how to write business services in particular languages
  - Java, C++, BPEL, PHP….
- ***Binding*** specifications
  - how to use access methods
  - Web services, JMS, RMI-IIOP, REST…
- ***Policy Framework***
  - how to add infrastructure services to solutions
  - Security, Transactions, Reliable messaging…

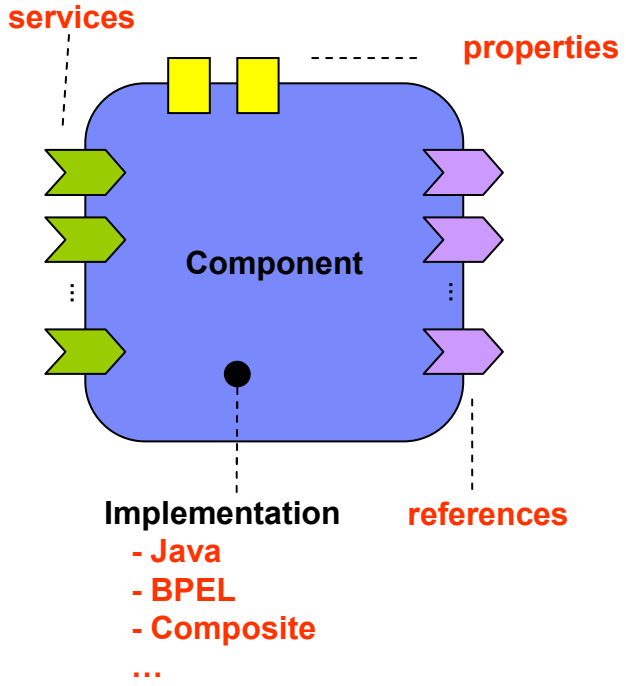
Example SCA assembly



Assembly Model Concepts

- Component
- Implementation
- Composite
- Service
- Reference
- Wire
- System

Component

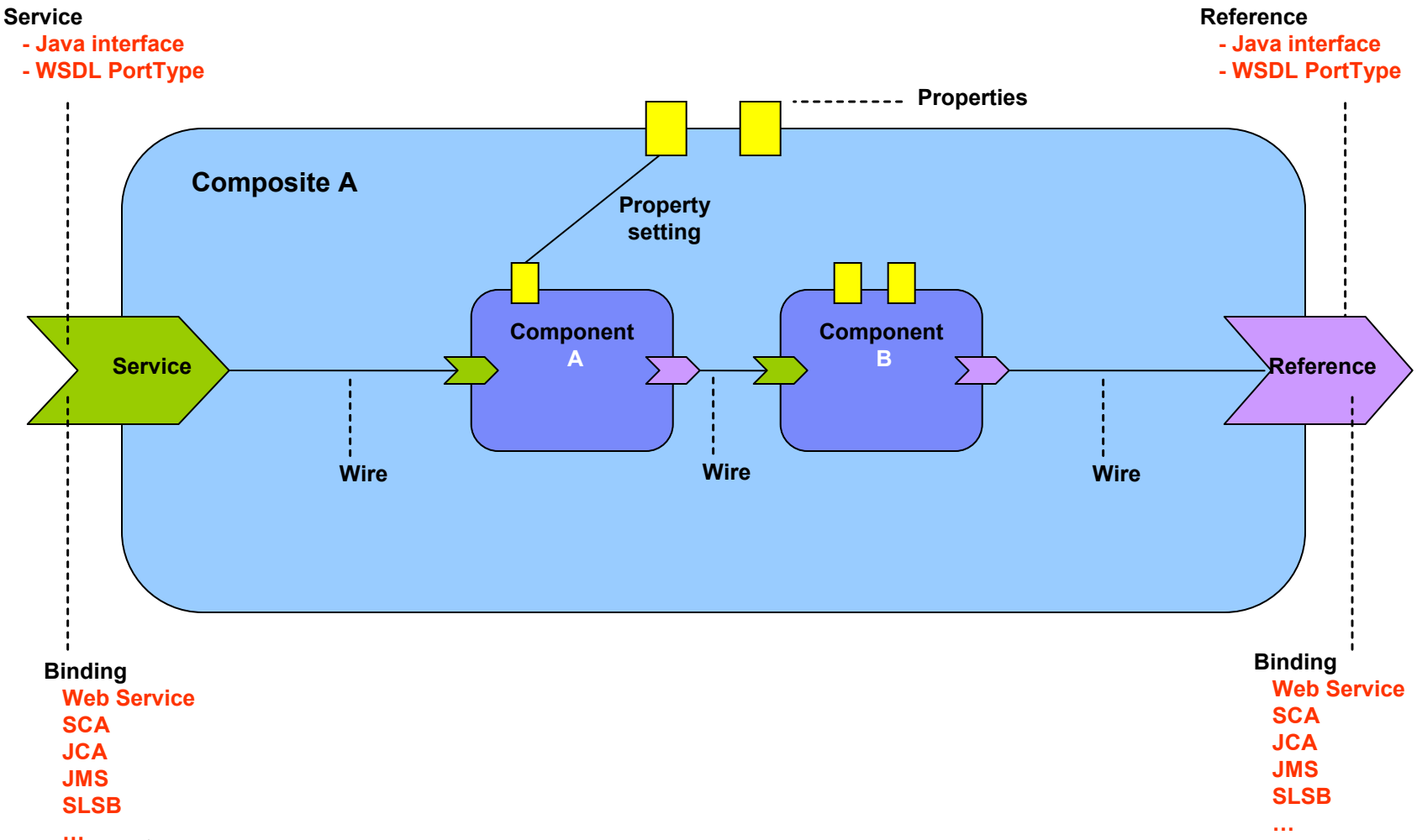


- ***Configured*** instance of ***implementation*** within a Composite
  - more than one component can use same implementation
- ***Provides*** and ***consumes services***
- Sets implementation ***properties***
- Sets service ***references*** by ***wiring*** them to services
  - wiring to services provided by other components or by references of the composite

Service Implementations

- Basic elements of business function
- Support for *different implementation technologies*
  - e.g. Java™, Spring, BPEL, C++, PHP, XSLT…
    - implementation type *extensibility*
  - *composite* can also be used as an *implementation*
- Provides business function via one or more *services*
- Uses other services through service *references*
- Service and references typed by *interfaces*
- *Scoped*
  - Runtime managed state and message routing

Composite


Service
- Java interface
- WSDL PortType
Reference
- Java interface
- WSDL PortType
Properties
Composite A
Property setting
Component A
Component B
Service
Reference
Wire
Wire
Wire
Binding
Web Service
SCA
JCA
JMS
SLSB
...
Binding
Web Service
SCA
JCA
JMS
SLSB
...

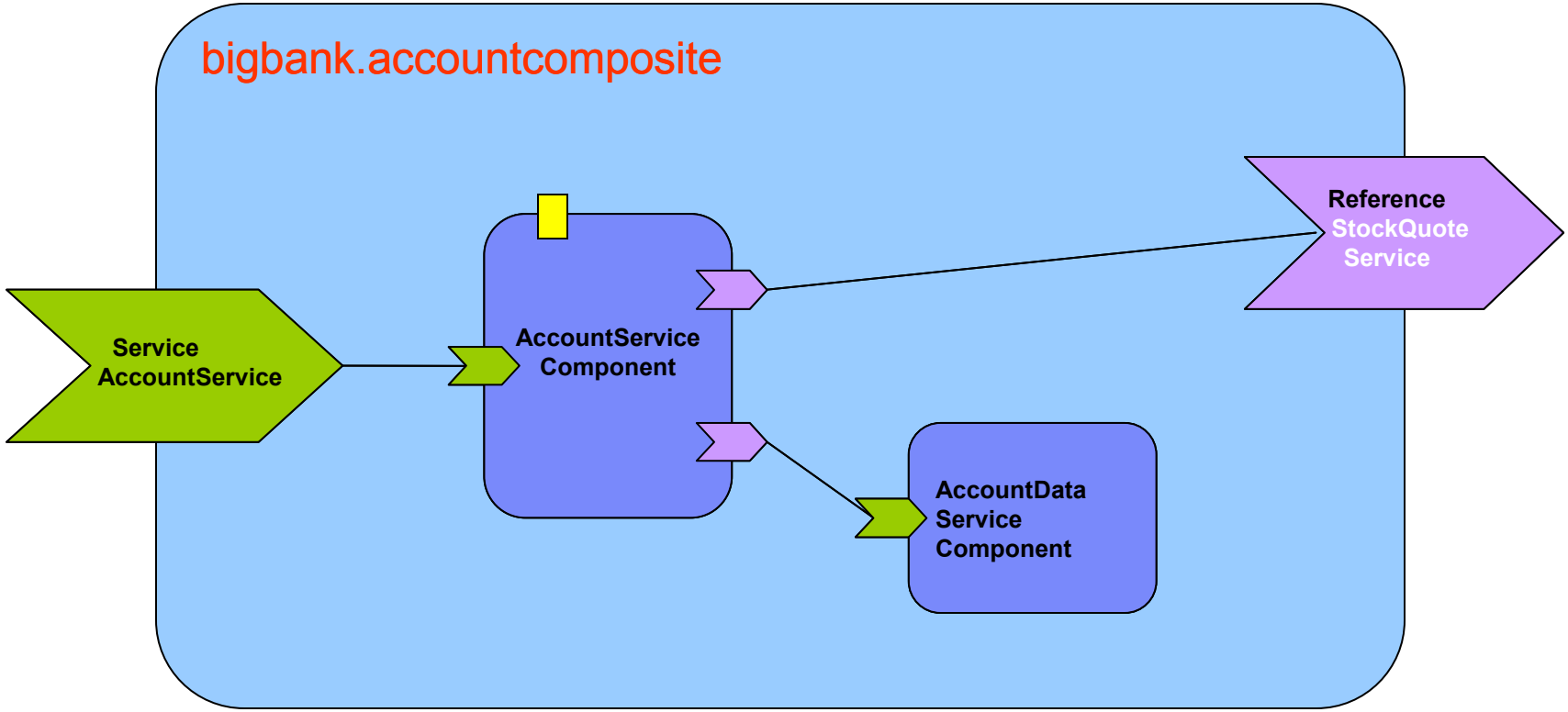
Composite

- ***Assembly*** of ***service components*** developed and deployed together
- Contains
  - public ***services***
  - service implementations organized as ***components***
  - required services as ***references***
  - ***wires*** connect components, services, and references
  - ***properties***
- May be used as ***implementation*** of components at next higher layer

SCA Interaction Model

- ***Synchronous*** & ***Asynchronous*** service relationships
- ***Conversational*** services
  - stateful service interactions
- Asynchronous support
  - "non-blocking" invocation
  - asynchronous client to synchronous service
  - callbacks

Example


bigbank.accountcomposite
Service
AccountService
AccountService
Component
Reference
StockQuote
Service
AccountData
Service
Component

sca file for bigbank.accountcomposite

```
<?xml version="1.0" encoding="ASCII"?>
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
           name="bigbank.accountcomposite" >

   <service name="AccountService">
      <interface.java interface="services.account.AccountService"/>
      <binding.ws port="http://www.bigbank.com/AccountService#
          wsdl.endpoint(AccountService/AccountServiceSOAP)"/>
      <reference>AccountServiceComponent</reference>
   </service>

   <component name="AccountServiceComponent">
      <implementation.java class="services.account.AccountServiceImpl"/>
      <property name="currency">EURO</property>
      <reference name="accountDataService" target="AccountDataServiceComponent"/>
      <reference name="stockQuoteService" target="StockQuoteService"/>
   </component>

   <component name="AccountDataServiceComponent">
      <implementation.java class="services.accountdata.AccountDataServiceImpl"/>
   </component>

   <reference name="StockQuoteService">
      <interface.java interface="services.stockquote.StockQuoteService"/>
      <binding.ws port="http://www.quickstockquote.com/StockQuoteService#
          wsdl.endpoint(StockQuoteService/StockQuoteServiceSOAP)"/>
   </reference>
</composite>
```

System

- Composites deployed, configured into ***SCA system***
  - SCA runtime – potentially *distributed*
- System contains ***components, services, references, wires***
  - configured using ***composites***
- Composites make deployment simpler
  - individual composites created, deployed independently
  - may contain only wires, components or externally provided services or references

SCA Client and Implementation Specifications

- Specify how service components and service clients are built
- Specific to a particular language or framework or language- or framework-specific APIs
- Extensible
- Currently defined C&I specifications:
  - BPEL
  - Java
  - Spring Framework
  - EJB
  - JAX-WS
  - C++
  - (PHP)

SCA Bindings

- Specific to particular:
  - Access Method / Protocol / Transport
  - Serialization
  - Framework
- Apply to services and references
- Typically added during deployment
- Currently defined bindings:
  - Web services binding
  - JMS binding
  - JCA binding
  - EJB (RMI-IIOP) binding

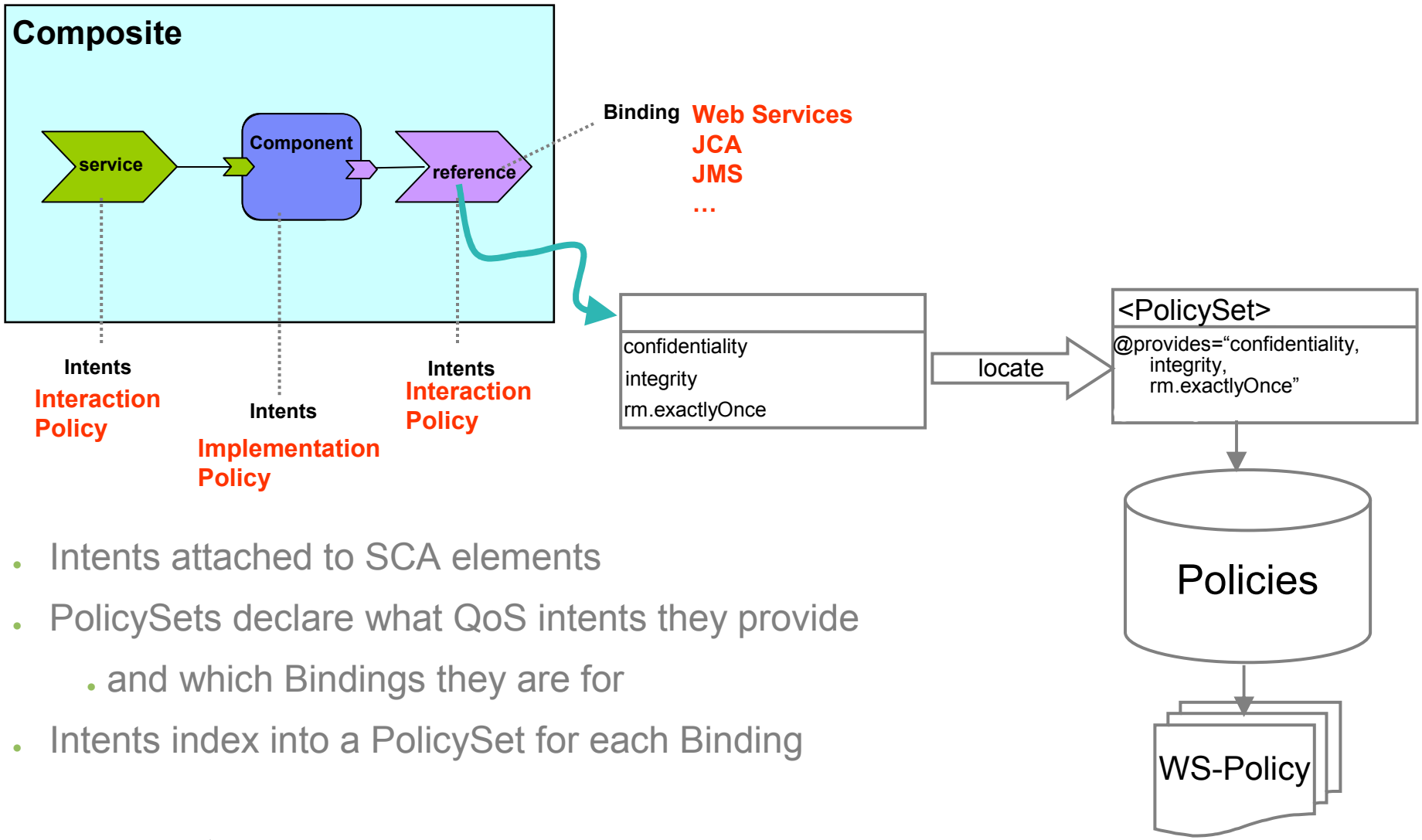
Policies and Infrastructure Capabilities

- ***Infrastructure*** has many configurable capabilities
  - Security: Authentication and Authorization
  - Security: Privacy, Encryption, Non-Repudiation
  - Transactions, Reliable messaging, etc.
  - Complex sets of configurations across multiple domains of concern
- SCA abstracts out complexity with a ***declarative model***
  - no implementation code impact
  - simplify usage via declarative policy intents
  - simple to apply, modify
  - complex details held in PolicySets

Policies, Profiles and Quality of Service

- Framework consists of:
  - SCA policy ***intent***
    - Each represent a single abstract QoS intent
    - may be ***qualified***
  - SCA ***policy sets***
    - Represent a collection of concrete policies to realize an abstract QoS intent
  - WS-Policy
    - A syntax for concrete policies in policy sets
    - others possible…

Attaching Profiles and mapping to PolicySets



- Intents attached to SCA elements
- PolicySets declare what QoS intents they provide
  - and which Bindings they are for
- Intents index into a PolicySet for each Binding

Interaction and Implementation Policies

- ***Interaction policies*** affect the contract between a service requestor and a service provider
  - Things that affect the interaction between them, such as message contexts, wire formats, etc.
  - eg. authentication, confidentiality, integrity
  - eg rm.atLeastOnce, rm.ordered
- ***Implementation policies*** affect the contract between a component and its container
  - Things that affect how the container should manage the component environment, such as transaction monitoring, access control, etc.
  - eg tx.transaction

Outline

- **SCA:Why?**
- **SCA: What?**
- **SCA: How and Where?**
  - **Open SOA (OSOA) Collaboration**
  - **OSOA & Evolution of Specifications**
  - **Future Work**
  - **Open Source Projects and Other Implementations**
  - **Useful Information and Pointers**
- **Summary**

The Open SOA (OSOA) Collaboration

- SCA specs being evolved by group of collaborators
  - BEA, CapeClear, IBM, Interface21, IONA, Oracle, Primeton Technologies, Progress Software, Red Hat, SAP, Rogue Wave, Software AG, Sun Microsystems, Sybase, TIBCO, XCalia, Zend Technologies
- OSOA is not a standards body
- Innovate rapidly and deliver the specification set to the community
  - Eventual submission to standards body
- Royalty Free
- Public website for specifications, white papers, news, etc
  - http://www.osoa.org
  - comment and feedback welcome
  - OSOA Supporters group

OSOA & Evolution of SCA Specifications

- Working towards SCA 1.0
  - Target delivery date of February 2007
- Will contain:
  - SCA Assembly Specification
  - SCA Policy Framework
  - SCA Client and Implementation for BPEL
  - SCA C&I for C++
  - SCA C&I for Java
  - SCA C&I for Spring Framework
  - SCA C&I for EJB*
  - SCA C&I for JAX-WS*
  - SCA Web Service Binding
  - SCA EJB Binding
  - SCA JMS Binding
  - SCA JCA Binding*

* = later publication date

Future Work

- Work will continue in the OSOA collaboration
  - SCA Eventing Model
  - SCA Client and Implementation Model for PHP
    - other scripting languages being investigated
  - SCA Client and Implementation Model for COBOL
  - other implementation languages & frameworks may follow

Open Source Projects and Other Implementaions

- Apache Tuscany Incubator Project
  - Aims to provide SOA programming runtime based on SCA and SDO
  - currently has “incubator” status within Apache
  - Java™ & C++ implementations today
  - Aim to support several runtimes and protocols in future
  - Associated PHP implementation on PECL site
- Eclipse SOA Tools Project
  - Eclipse-based tooling for SOA applications and systems
  - Based on SCA as model for solutions built using SOA
  - Target range of systems including SCA runtimes such as Tuscany
- Several other OSOA collaboration vendor implementations
  - Oracle Fabric, IBM WebSphere, RogueWave, TIBCO….

Useful Information And Pointers

- contact:
  - anish.karmarkar@oracle.com
  - mike_edwards@uk.ibm.com
- SCA, SDO specifications and related material
  - http://www.osoa.org
- Apache Tuscany Incubator project
  - http://incubator.apache.org/tuscany
- Eclipse SOA Tools Project
  - http://www.eclipse.org/stp/

Summary

- SCA models systems built using a Service Oriented Architecture
  - supports Service Implementation, Service Assembly
  - open to many kinds of service implementation
  - open to many types of service access
  - declarative intent & policy approach to application of Security & Transaction