An Introduction to SCA & SDO: How to Build Business applications

Jeff Mischkinsky
Director, Oracle Fusion Middleware and Web Services Standards

OSOA Collaboration | 30th May 2007

Agenda

- SOA -The Business Drivers
- The SOA Vision - Model and Characteristics
- SCA/SDO - An Overview
- SCA - Details and Example
- Summary

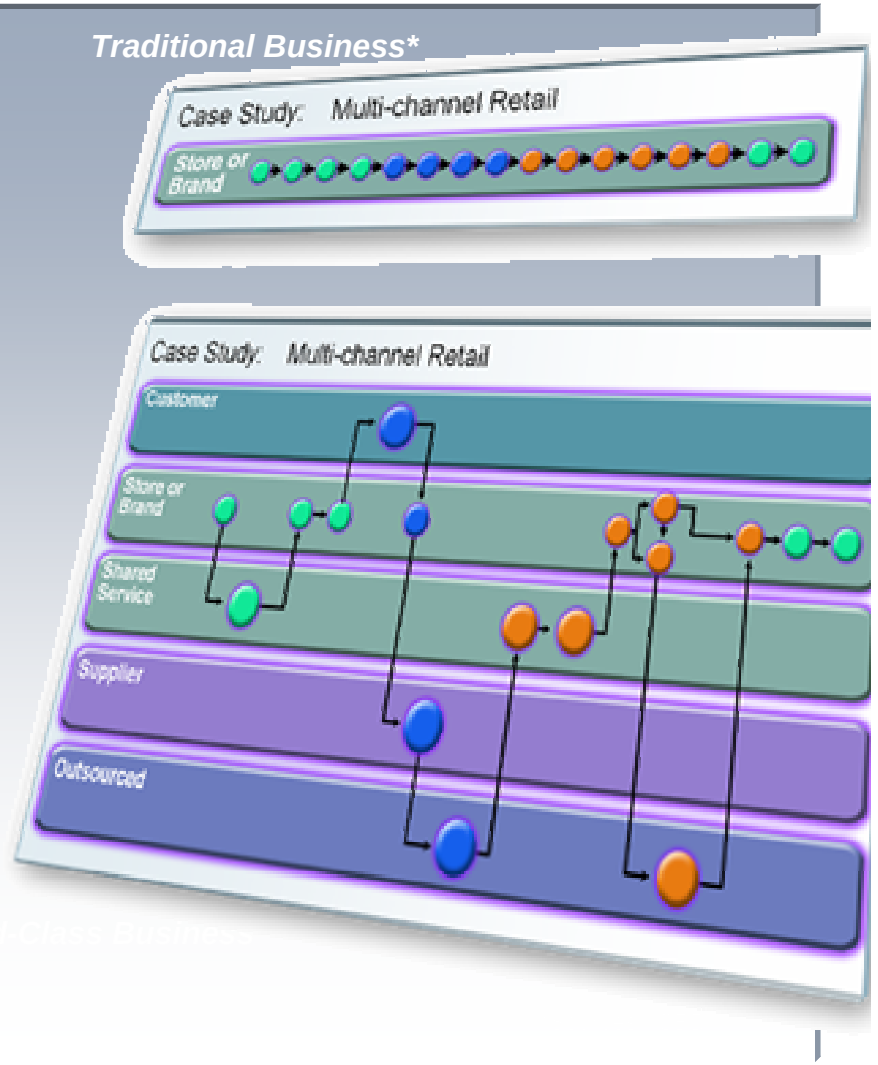
SOA - The Business Drivers

OSOA Collaboration | 30th May 2007

Business Drivers

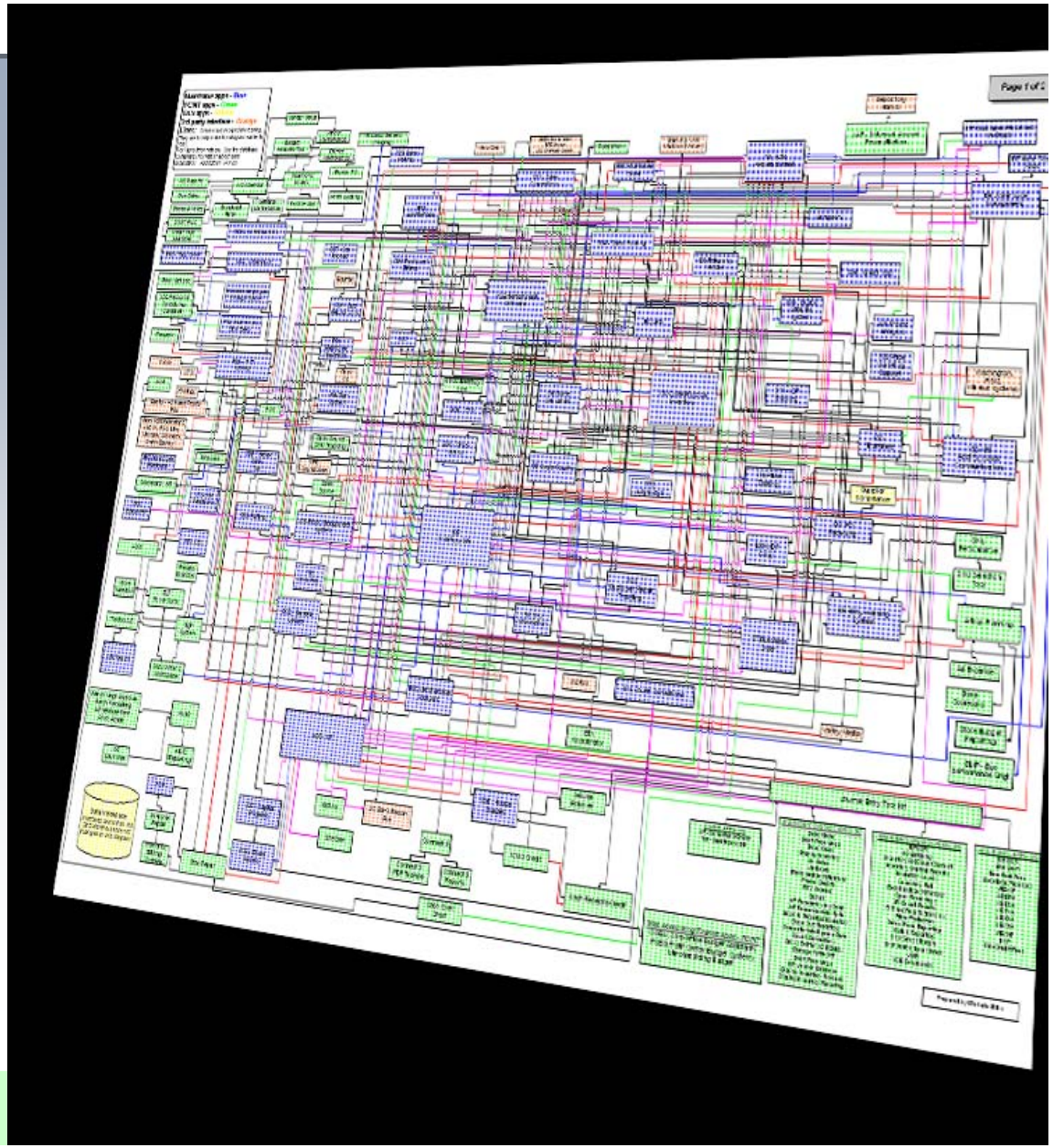
Flexible business requires flexible IT

- **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**

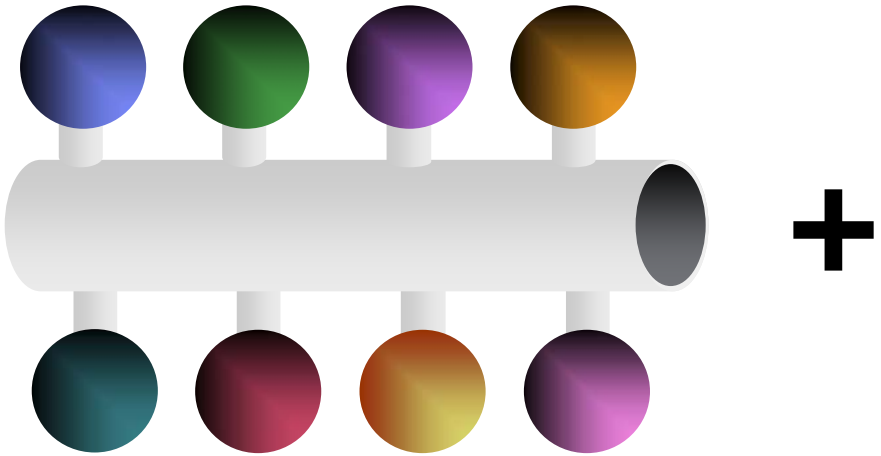


What We have Today

- 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 Vision:
Model and Characteristics

OSOA Collaboration | 30th May 2007

Service Oriented Architecture Characteristics

- Service model for business functions
  - Services are coarse-grained and network-facing
- Characteristics
  - Flexibility
    - Autonomous services are highly reusable
      Non predictive patterns of use
  - Productivity
    - High level of abstraction
  - Comprehension
    - Well-understood system architecture and behavior

The SOA Programming Model (1)

- SOA Programming Model derives 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

The SOA Programming Model (2)

- Core Elements:
  - ***Service Assembly***

    technology- and language- independent representation of composition of services
  - ***Service Components***

    technology- and language-independent representation of composable service implementation
  - ***Service Data Objects***

    technology- and language-Independent representation of service data entity

SCA/SDO:
An Overview

OSOA Collaboration | 30th May 2007

Service Component Architecture (SCA): A Simplified Programming/Deployment Model for SOA

- A model for building components, assembling them into applications, and deploying them to various runtime environments
  - Components can be built from ***new or existing code using SOA principles***
  - ***vendor-neutral*** – supported across the industry
  - ***language-neutral*** – components written using any language
  - ***technology-neutral*** – use any communication protocols and infrastructure to link components

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 …

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

Key benefits of SCA

- ***Loose Coupling***: Components integrate without needing to know how other components are implemented
- ***Flexibility***: Components can easily be replaced by other components
- ***Services*** can be *easily* invoked either synchronously or asynchronously
- ***Composition*** of solutions: clearly described
- ***Productivity***: Easier to integrate components to form composite application
- ***Heterogeneity***:
  - multiple implementation languages / different frameworks
  - multiple communication mechanisms
- ***Declarative*** application of infrastructure services
  - WSDL is the contract language
- ***Simplification*** of development experience for **all** developers, integrators and application deployers

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…
- ***Interaction Model*** for connected and disconnected services
  - Synchronous, Asynchronous and Conversational services relationships

Service Data Objects (SDO): Simplified Data Handling for SOA

- A simplified programming model for access to business data
  - Uniform model for data formats
  - Uniform model for data access
- Independent of form of data (source and target)
  - Data stored in wide variety of formats
  - RDBMS
  - XML formats
  - Unstructured
- Interaction style designed for SOA
  - Disconnected, optimistic-update policy
    - Read, manipulate, update

SCA:
Details and Example

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…

Basic Assembly Elements

- ***Component***
  - ***configured*** instance of ***implementation***
  - ***provides*** and ***consumes services***
  - sets implementation ***properties***
- ***Composite***
  - ***combines*** collections of ***components***
  - ***wires*** references to services
  - selects ***bindings***, endpoints
  - applies ***policies***

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

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 Policies and Infrastructure Capabilities

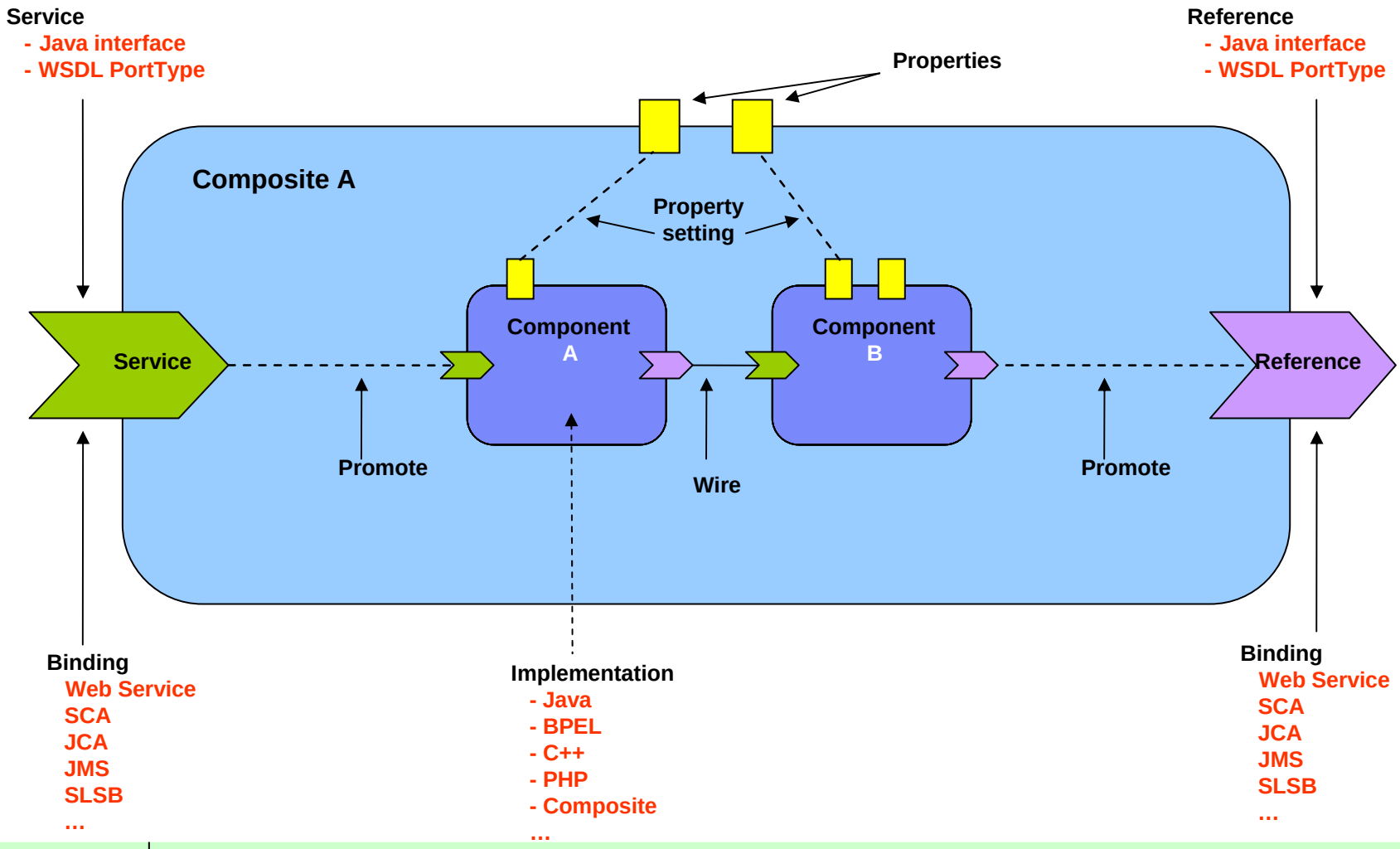
- ***Infrastructure*** has many configurable capabilities
  - Security: Authentication and Authorization
  - Security: Privacy, Encryption, Non-Repudiation
  - Transactions, Reliable messaging, etc.
  - Complex sets of configurations across multiple domains of concern

- 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

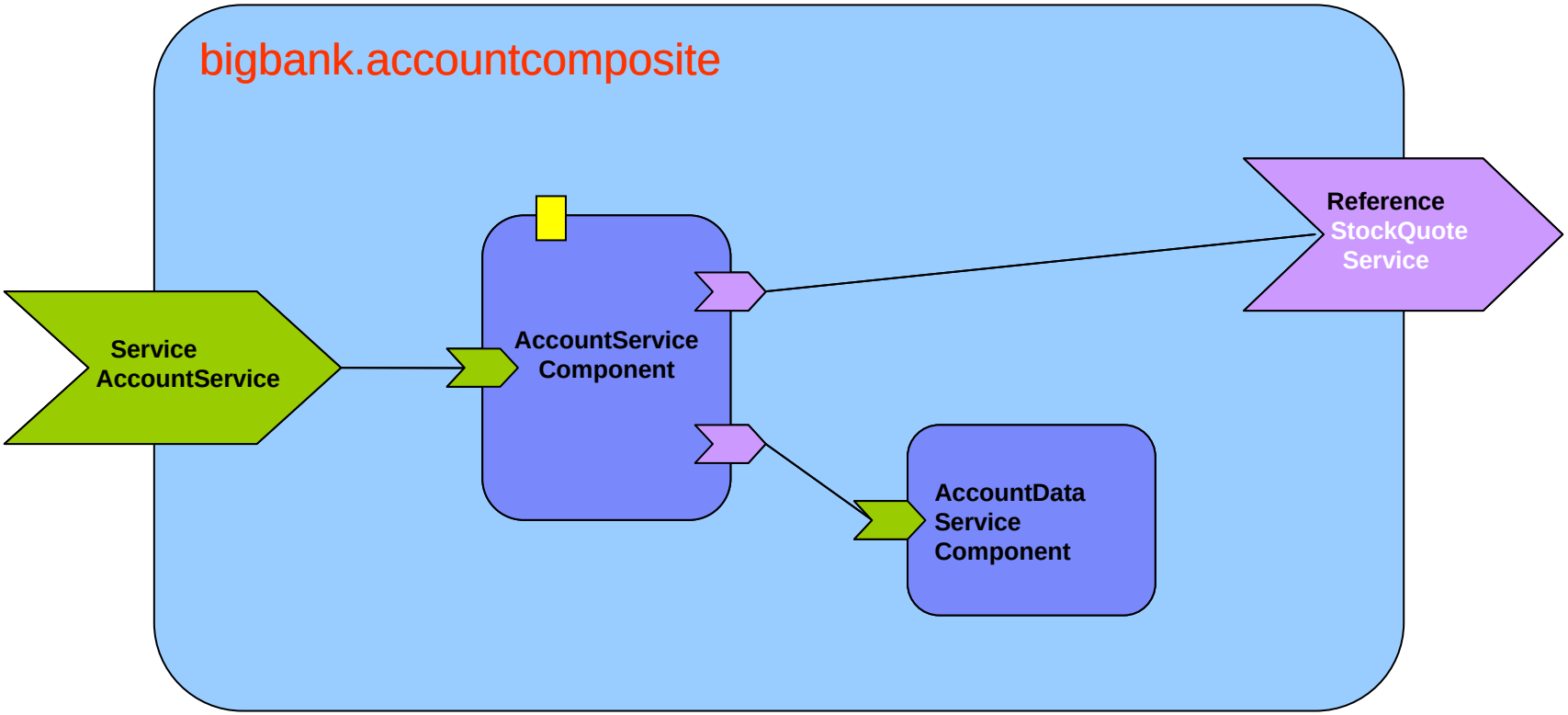
SCA Policy Framework

- Framework consists of:
  - SCA policy ***intent***
    - Each represent a single abstract QoS intent
    - E.g. reliable messaging
  - 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…

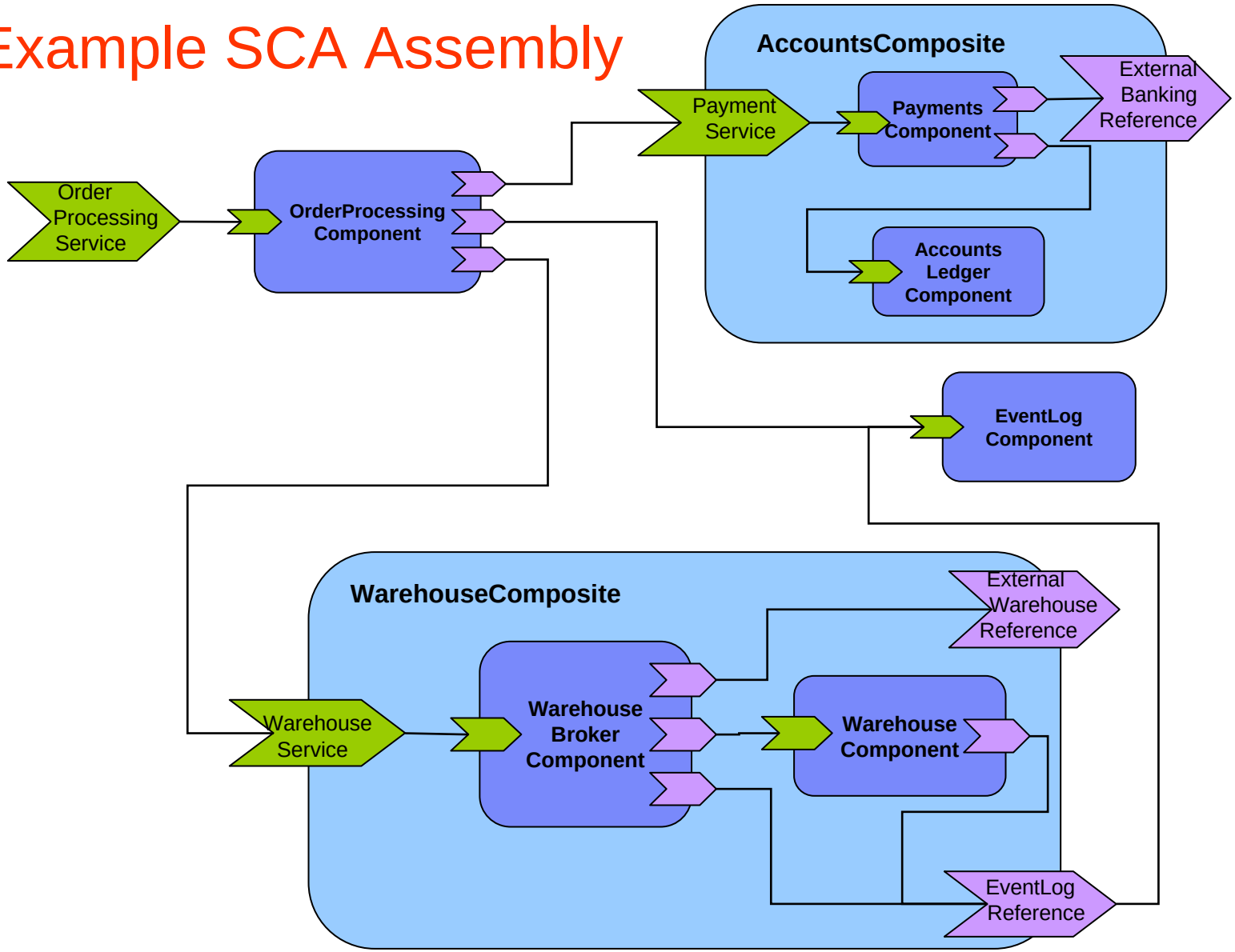
SCA Composite



Example Composite



Example SCA Assembly



Summary

Summary

- Fundamental Service Oriented Architecture value prop
  - less expensive integration, more flexibility
- SCA models systems built using a SOA
  - SDO provides the ideal data manipulation layer
- SCA is key enabler for SOA
  - Maybe “THE” key enabler

Thank you!

Questions?