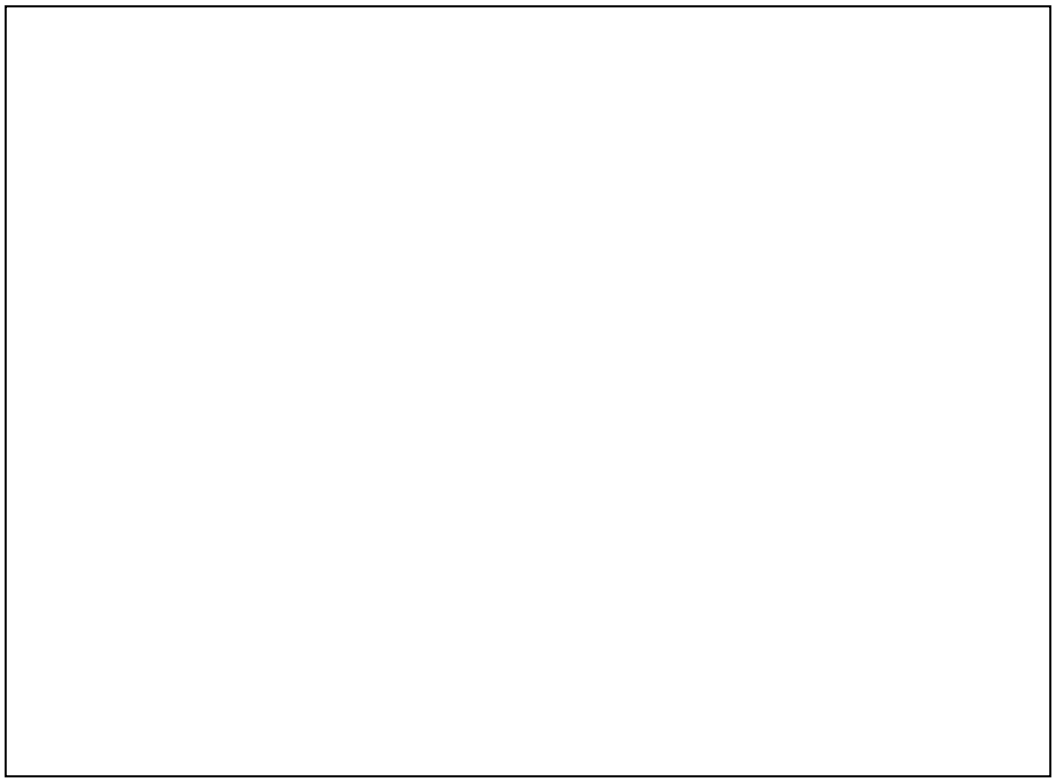
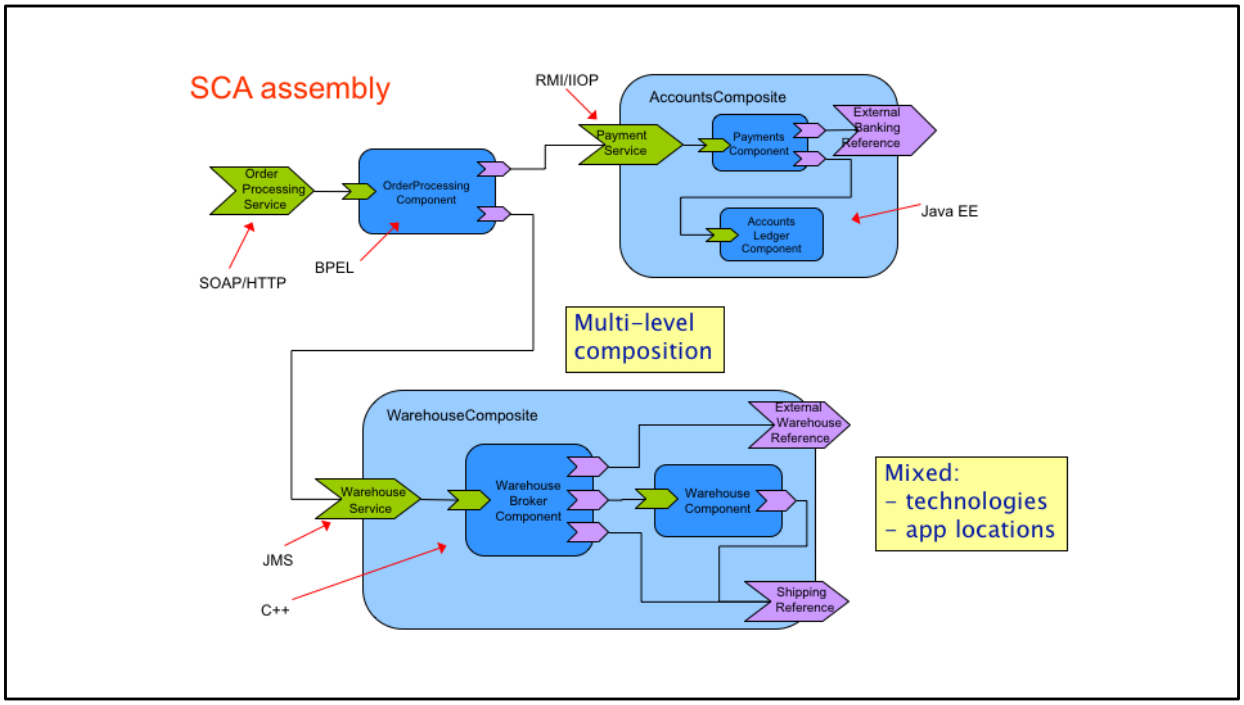
Service Component Architecture (SCA): Simplified Programming Model for SOA

- model for:
- ***building*** service components
- ***assembling*** components into applications
- ***deploying*** to (distributed) runtime environments
  - Service components 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




SCA assembly
RMI/IIOP
AccountsComposite
Payment Service
Payments Component
External Banking Reference
Order Processing Service
OrderProcessing Component
Accounts Ledger Component
Java EE
SOAP/HTTP
BPEL
Multi-level composition
WarehouseComposite
External Warehouse Reference
Warehouse Service
Warehouse Broker Component
Warehouse Component
Mixed:
- technologies
- app locations
JMS
C++
Shipping Reference

Agenda

- A little history
- SCA and SOA
- ***SCA scenarios***
- SCA specifications
- OASIS SCA TCs
  - major challenges
- Related & future work

Bottom-up Composition

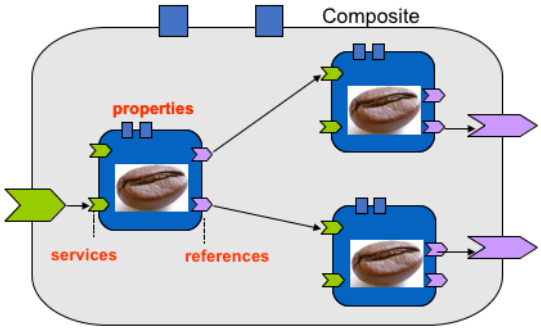
Select a set of existing component implementations for building the new composite

Configure the component properties

Draw internal wires

Wrap the components in a composite and configure external services/references

Hand off the composite to Deployer



Top-down Composition

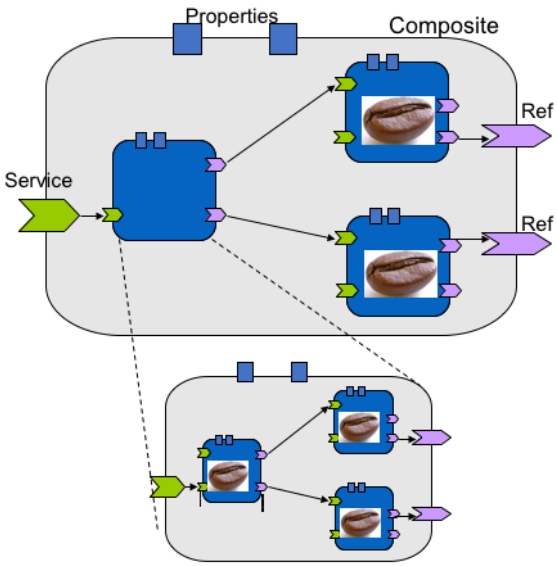
Start with gathering requirements for the top-level composite

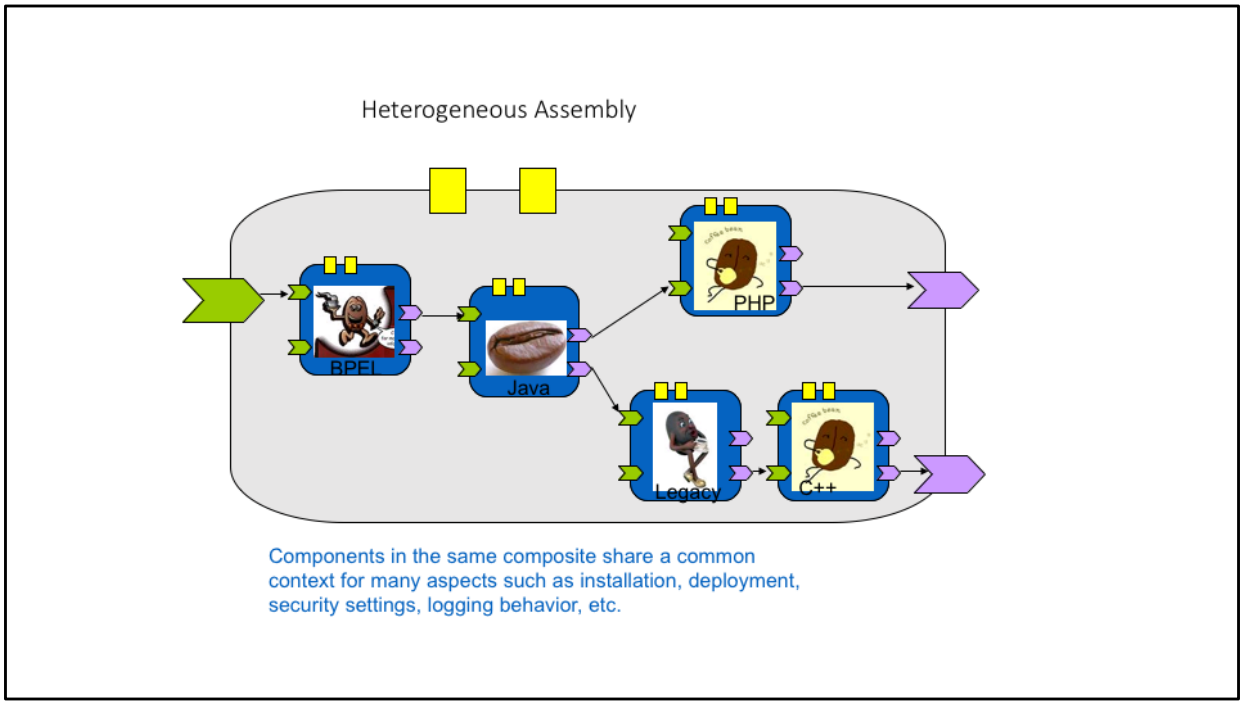
Define the services/references and properties for the composite

Break down the composite into individual components and wires between them

Recursively break down each component

Hand off the individual component contracts to developers for implementation





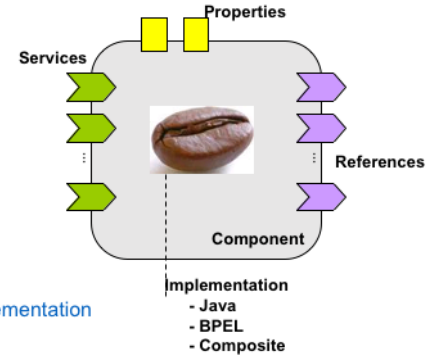
•In top-down composition scenario, it is important to choose an implementation type best suited for the component functionality
•In bottom-up composition scenario, reusing existing heterogeneous components is a practical need

Implementation Reuse – By Configuration

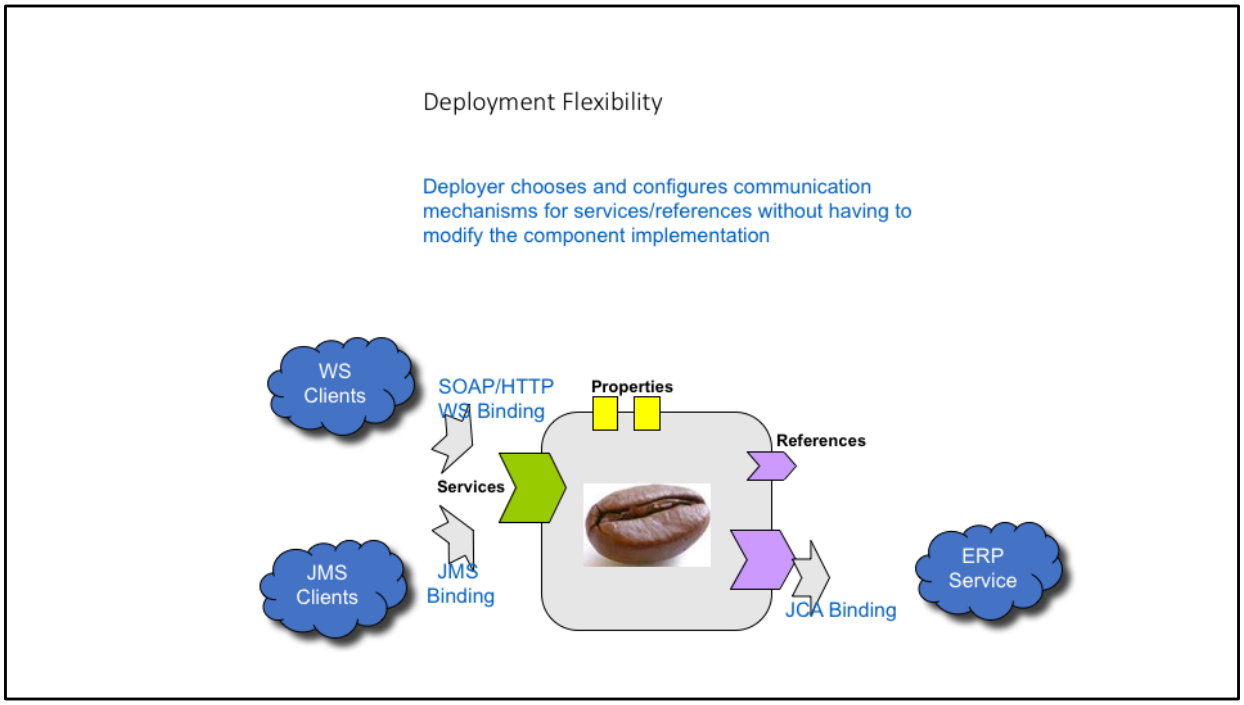
Select an existing component implementation

Configure its behavior (via setting props, refs) to match the current requirements

E.g. Configure multiple instances of product pricing component, each with different currency, tax rate, discount charts, etc.



Deploy the component implementation
- Multiple instances of the same implementation may be running simultaneously



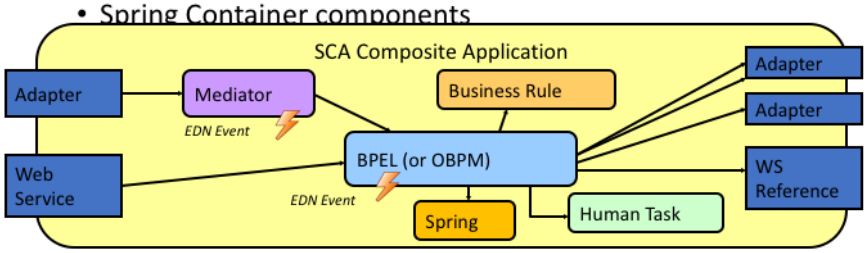
Case A: Use different middleware technologies for different deployments of the same component

- **Stateless EJB**
- **Web services**
- **JMS Queue**

Case B: Configure the same middleware (e.g. Web Services) differently for each deployment of that component

SOA Suite 11g SCA Composite "Components"

- The configured elements that can be part of a SOA Suite 11g SCA Composite Application are:
  - Mediator components
  - BPEL Process components
  - OBPM Process components
  - Business Rule components
  - Human Task components
  - Spring Container components
  - Bindings (services or references
  - Adapters (inbound or outbound)
  - EDN Events



Oracle SOA Suite 11*g* Components

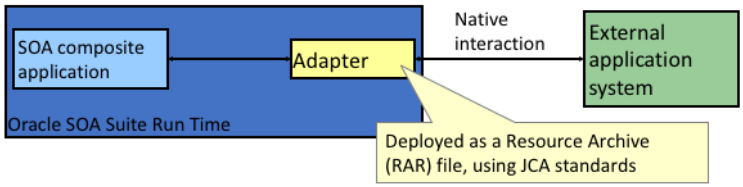
The key components you work with to build SOA composite applications in this course include:

- Various adapter services, such as File, JMS, and Database adapters
- Oracle Mediator components for filter, data enrichment, and routing
- Oracle BPEL process components, for service orchestration and process flow
- Oracle Business Rule components, for business rule implementation and execution
- Oracle Human Task components, for human workflow and human interaction.

You also explore publishing and subscribing to business events through Mediator components. The diagram represents a collection of cooperating components that can make up a composite application, which may expose multiple entry points. Subsequent pages in the lesson provide more information about each of these components.

Adapter Services

- Adapters provide a service interface that:
  - Exposes external application functionality in a form that can be used by SOA composite application components
  - Converts request and responses into a form suitable for other (external) systems
  - Implements interfaces by using the Java Connector Architecture (JCA) API standards



Adapter Services

The role of adapters is to execute operations (functional) exposed in a standard way for a SOA composite application to translate message data in a form that can be accepted by or received from an external application. Adapters, based on JCA API standards, provide custom implementations for exposing existing functionality in the form a service, such that investment in existing applications can be preserved and reused in a SOA environment. Oracle SOA Suite 11*g* is shipped with a variety of adapters commonly used to access functionality that is not normally available in a service-oriented context, such as:

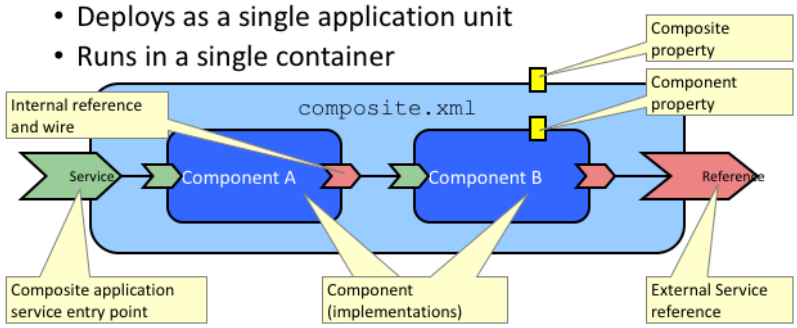
File and FTP Adapters for reading, write, and transferring of files
Database Adapter for interacting with relational database data
JMS Adapter for communicating asynchronously with other applications through message-oriented middleware (MOM) services, such as Oracle Advanced Queuing, IBM Message Queue among others.
Custom Adapters that expose a variety of functionality as service operations.

Adapters developed with the JCA API standards supports the ability to create custom implementations to expose existing functionality that is not usually

available to an SOA application context. Adapters are an integration technology that extend the reach of SOA composite application.

Composite Applications

- A composite application:
  - Implements an **assembly model** stored in a SCA descriptor file called `composite.xml`
  - Conforms with the SCA specifications
  - Deploys as a single application unit
  - Runs in a single container



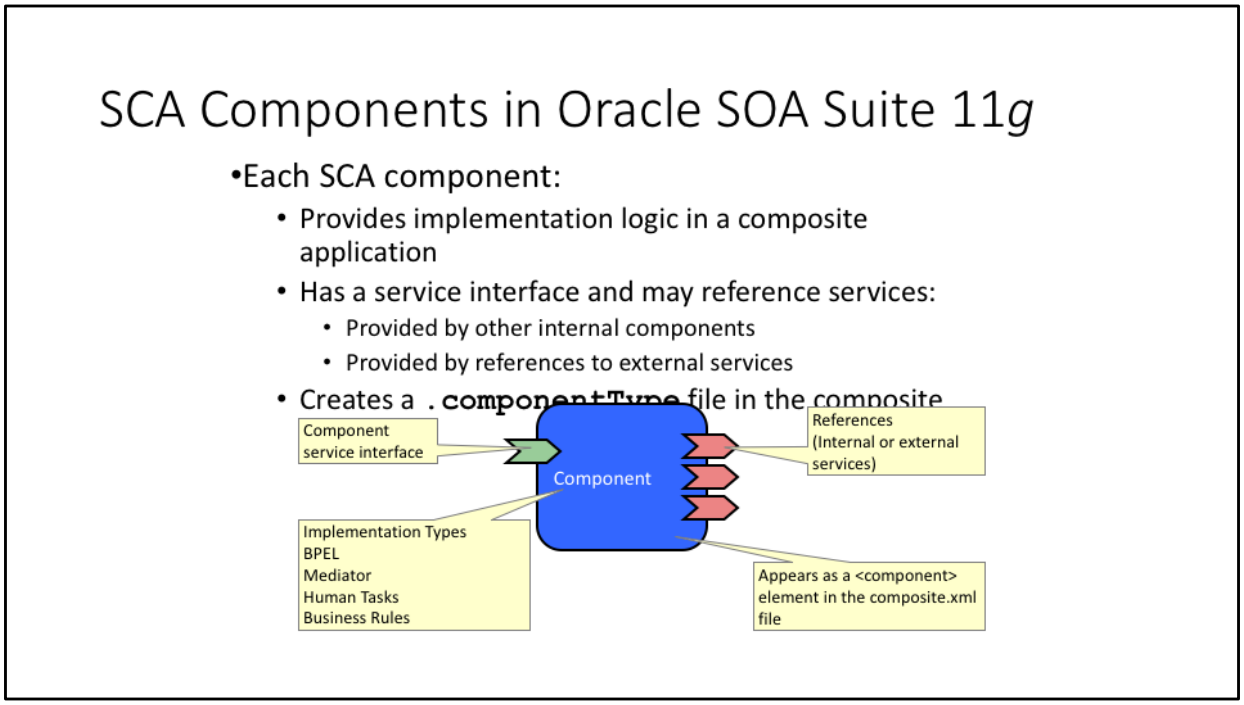
Composite Applications

A composite application provides a coarsely-grained service for a business application, which processes information supplied in XML messages. The composite application is an implementation of an **assembly model**, expressed in an SCA descriptor file called `composite.xml`, which can be comprised of XML elements representing:

- Service entry points, provide external clients with a service interface
- Components, providing the implementation of the composite application functionality
- External references, represent services used by components to implement composite application functionality
- Wires, define messages paths from service entry points to components, between components, and from components to external references.
- Bindings and properties, configured for the composite application and components

The SCA Descriptor conforms with SCA specifications and enables the SCA Composite and its internal components to be deployed as a single application unit. Services for external references are deployed independently from the composite application.

Note: A composite application is deployed to a single (or clustered) container and executes within the run-time context of single container (or cluster). Remember, that external referenced services are distributed. However, the internal implementation used to invoke a service is optimized at run-time, for an external reference to a service, which runs in the same container as the composite application.



SCA Components in Oracle SOA Suite 11*g*

Components implement parts of a composite application. Oracle SOA Suite 11*g* provides the following types of component implementations:

- BPEL components, for service orchestration in a business process
- Mediator components, for message routing and enrichment within a composite application
- Human Task components, for human workflow interactions
- Business Rules components, for externalizing evaluation of business rules

Each component implementation provides service interfaces that can be exposed to external clients of a composite application, or referenced internally within a composite by other components. Components may also use service references to interact with services internal or external to the composite, and may be configured with properties used within the component.

Note: The SCA specification defines components that support implementations of many different technologies, which reflect the reality of businesses containing mixed systems with different technologies developed over many years. The flexibility of implementation language enables the selection of technologies better suited to different types of work—for example,

using BPEL for business processes, and Java or C++ for detailed numerical processing.

Note: A composite can itself be used as an implementation for a component.

Each component is identified by a `<component>` element in the SCA descriptor. The `<component>` element identifies the component type and the location of the component source.