



## Service Component Architecture (SCA) - a reference architecture for SOA

Johan Eltes, Callista Enterprise AB  
Johan.eltes@callista.se

Presented by Cornerstone

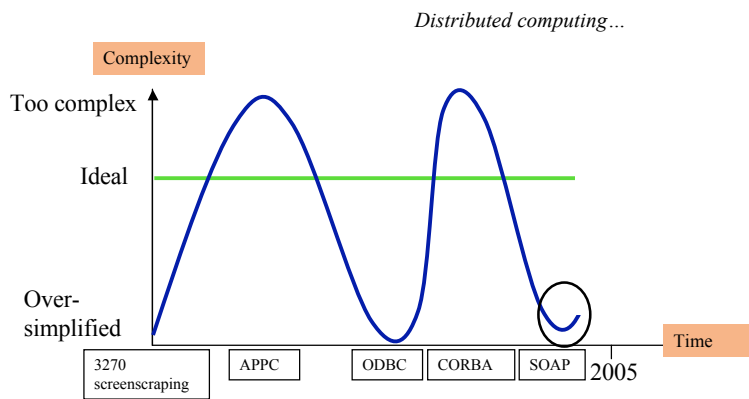
### Agenda

- SOA - from SOAP to Service Component Architecture
- Large-scale SCA within automotive
- Summary
- Q&A

xpertz

## SOA infrastructure background

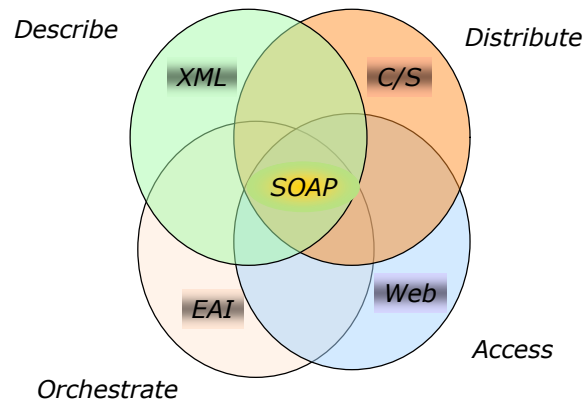
2001: SOA is distributed computing...



Cornerstone Slide 3

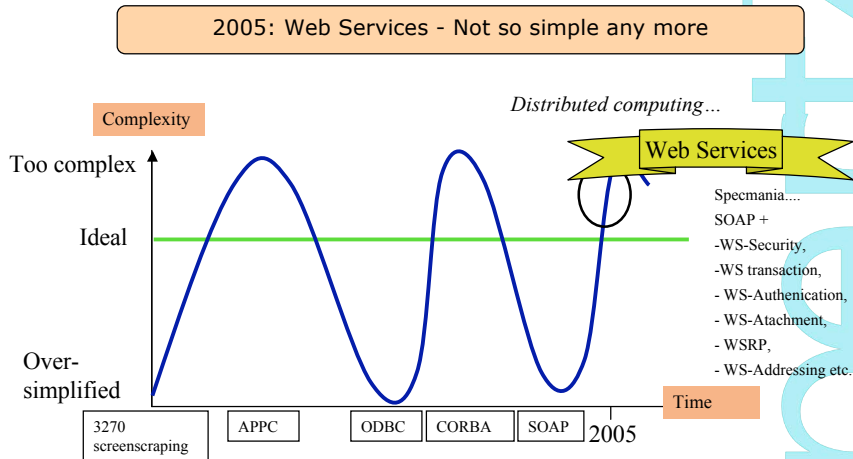
## The Evolution of SOA infrastructure

2001: SOAP (Simple Object Access Protocol) - The Holy Grail



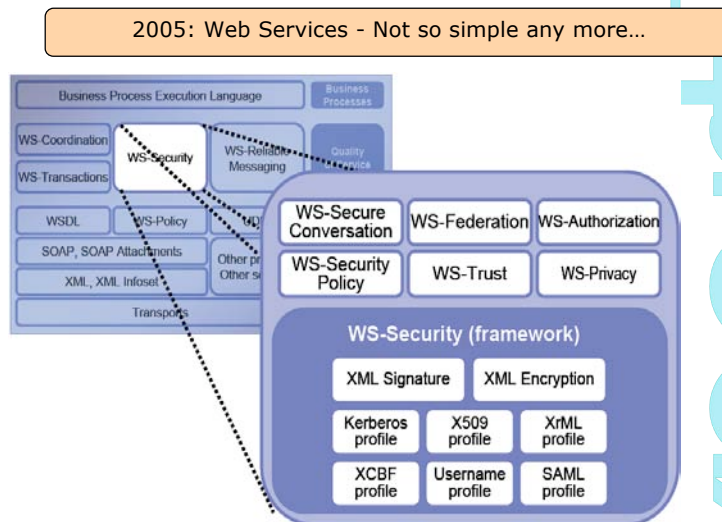
Cornerstone Slide 4

## The Evolution of SOA infrastructure



Cornerstone Slide 5

## The Evolution of SOA infrastructure



Cornerstone Slide 6

## SCA Background

- What is the SCA business case?
  - To help organisations develop services out of reusable components independently of service infrastructure (and programming language)
  - Define a reference architecture for successful implementation of services

Cornerstone Slide 7

## SCA Background

30 Nov 2005

“A group of tech industry heavyweights today unveiled a new programming model for service-oriented architecture.”

- What is it, formally?
  - An initiative represented by a set of specifications jointly published by industry giants (nov. 2005)

Cornerstone Slide 8

xpertz xpertz

## SCA Background

- Who exactly are those giants?
  - BEA, IBM, Iona, Oracle, SAP, Siebel, Sybase, Interface21
  - SCA (pre-standard) support included in WebSphere 6 product family (WPS and WID)
  - 25% of sessions (approx. 90) at WebSphere Technical Exchange Nov. 2005, referenced SCA

Cornerstone Slide 9

## SCA Background

- In what context does SCA belong?
  - Service development, composition and deployment
  - Any level of development language - from BPEL to Java
  - Application- and integration space

Cornerstone Slide 10

xpoertz(xpoertz)

## The SOA strategy

- What business management expects from IT
  - Agility in supporting new processes and integrating new businesses
  - Reduce TCO of the system portfolio
- What IT management expects from SOA
  - Re-use of business logic
  - Service composition (create new services from existing ones)
  - Interoperability (Net - JavaEE, Application to Application)
  - Virtualization (location- and infrastructure transparency)
  - De-fragmentation (one service in one place)
  - Data consistency (e.g. one source for customer data)

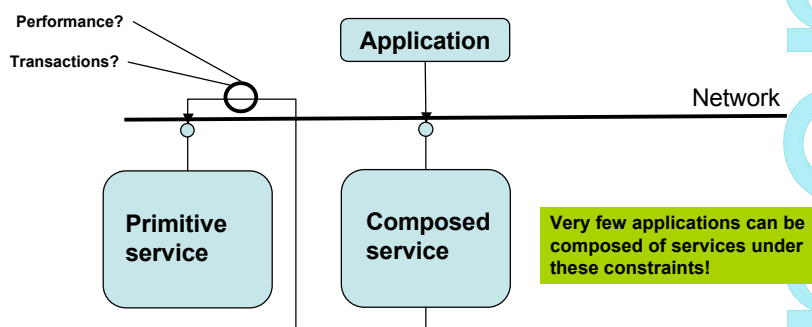
X

Cornerstone Slide 11

## SOA without SCA

SOA as we are used to think about it:

- The runtime view
- Re-use = invoke deployed service over the network



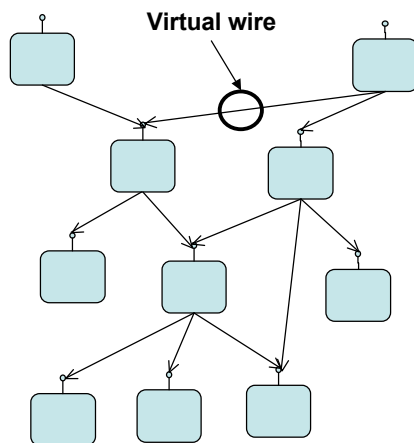
Cornerstone Slide 12

## The business case for SCA

- Re-use and composition simply doesn't happen
  - Current SOA is about deployed, networked services
  - Infrastructure- and runtime constraints prevents re-use and composition
  - *We need to decouple business logic from service infrastructure*
- This creates the business case for SCA
  - SCA is a reference architecture for creating service components that become building blocks of a SOA.
  - Service components are re-used in multiple services and applications

Cornerstone Slide 13

## The SCA view



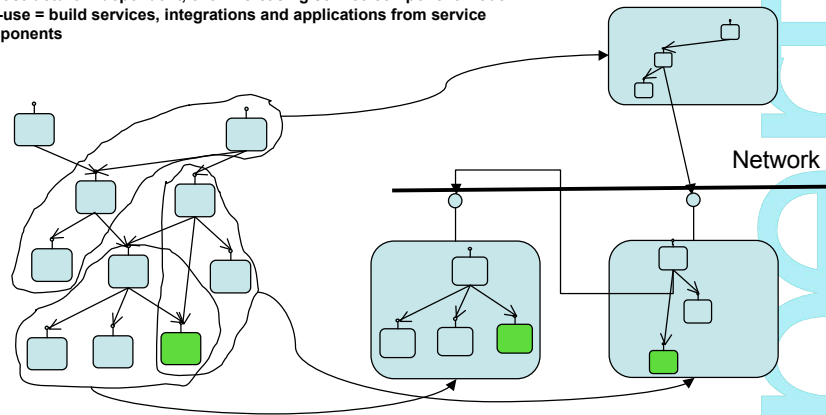
We go from "I'm developing a WebService"  
to "I'm developing a piece of re-usable  
business logic"

SCA decouples the investment in business  
logic from the evolution of infrastructure

Cornerstone Slide 14

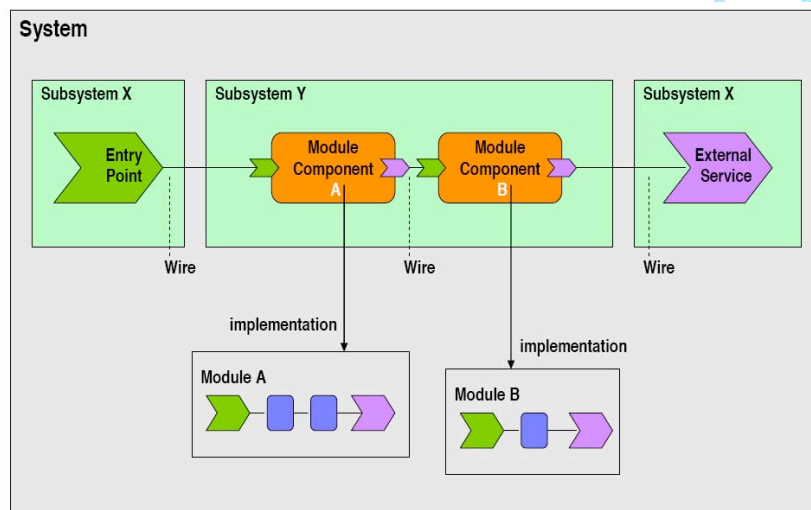
## SOA with SCA

- SCA decouples re-use and composition from the runtime view:
- Infrastructure-independent, ever-increasing service component model
  - Re-use = build services, integrations and applications from service components



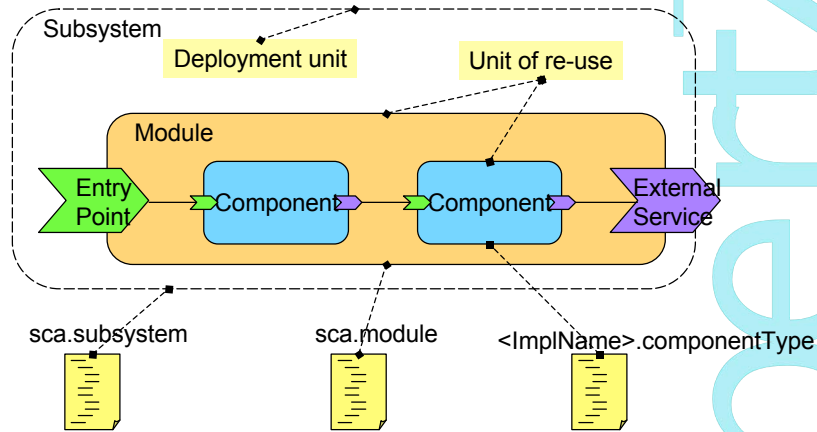
Cornerstone Slide 15

## What does SCA look like?



Cornerstone Slide 16

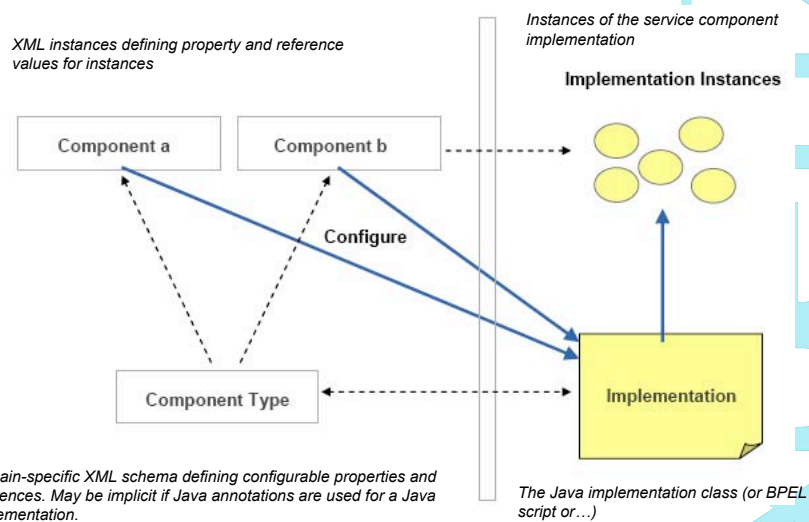
## What does the standard contribute?



SCA standardizes how composition- and infrastructure bindings are defined. Allows subsystems and modules to be migrated, re-used, composed and re-deployed across tool- and infrastructure vendors

Cornerstone Slide 17

## Service Component Configuration



Domain-specific XML schema defining configurable properties and references. May be implicit if Java annotations are used for a Java implementation.

The Java implementation class (or BPEL script or...)

Cornerstone Slide 18

## Example Java Service Component - using annotations

```
@Remotable
public interface MyService {
    public void serviceMethod(String s);
}

public class MyServiceImpl implements MyService {

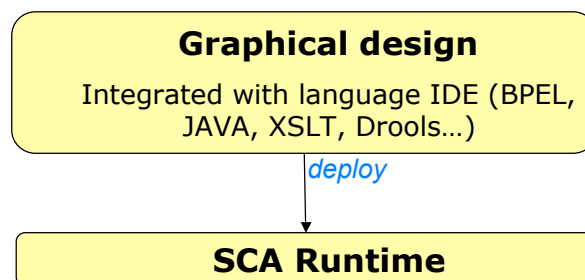
    @Property
    private String configProperty;

    @Reference
    private AnotherService anotherService;

    public void serviceMethod(String s) {
        // ...
    }
}
```

Cornerstone Slide 19

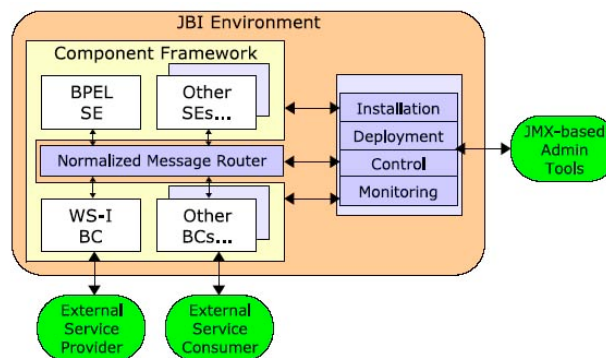
## SCA Tooling



Cornerstone Slide 20

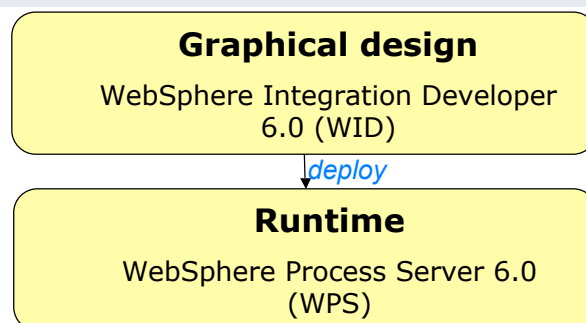
## JBIs as SCA runtime

- A Java Business Integration container - the ideal SCA deployment environment?
  - Binding Components
  - Service Engines
  - NMR (wiring)



Cornerstone Slide 21

## IBM SCA tooling

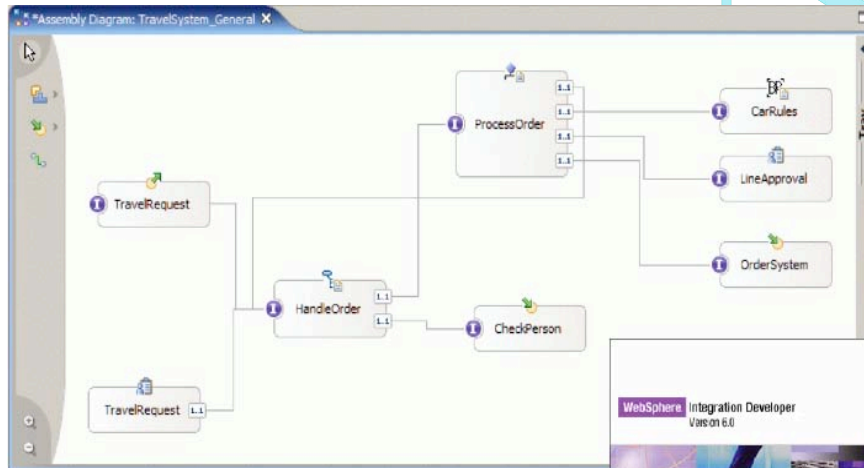


*Upcoming open source tools:*

- *Apache Tuscany (in incubator)*
- *Eclipse SOA Tools Platform*

Cornerstone Slide 22

## IBM WebSphere Integration Developer



Cornerstone Slide 23

## How SCA may affect you

- Infrastructure management
  - Standard allows you to select best-of-breed SOA infrastructure and tooling across vendors
  - Corporate infrastructure management not locked by applications being bound to yesterdays decisions
  - Tactical infrastructure decisions can be made to keep projects going
- Application- and service portfolio management
- Integration office

Cornerstone Slide 24

## How SCA may affect you

- Infrastructure management
- Application- and service portfolio management
  - Investment in *Business Logic* survives infrastructure evolution. This is a *revolution* for projects of the past 15 years!
  - *Software Quality* increases since software testing becomes much more efficient when decoupled from infrastructure
  - Applications, services, batches and utilities are all composed of re-usable, versioned service components
- Integration office

Cornerstone

Slide 25

## How SCA may affect you

- Infrastructure management
- Application- and service portfolio management
- Integration office
  - Tooling, people, infrastructure and concepts are shared with application portfolio management.

Cornerstone

Slide 26

xpoe rtz( )  
xpoe rtz( )  
xpoe rtz( )

## SCA adoption strategies

- Wait for standard-based products (1 - 2 years)
- Use Pre-standard SCA tooling
  - WebSphere Integration Developer and WebSphere Process Server supports most SCA concept today. A safe road to standard SCA.
- Start applying SCA architectural principles
  - Use Open Source binding technologies, like the Spring framework
  - By definition, business logic will seamlessly migrate to SCA standard tooling
  - Large scale case-study in a minute!

Cornerstone Slide 27

## Large-scale SCA within automotive

- Mission
  - Replace a large number of plant-specific systems (20 plants) with a common (global) portfolio of systems
- Business case
  - Disarm the legacy bomb
  - Reduce cost of maintenance and operations
  - Improve business agility and business process support
- Business areas
  - Part manufacturing
  - Vehicle Assembly
- System domains
  - Material ordering
  - Quality control
  - Material Distribution
  - Customer order slotting
  - Production control
  - Production planning
  - Device communication

Integration of  
Packaged  
Solutions

New  
development



Cornerstone Slide 28

## Architectural non-functional goals

- Minimal real-time dependencies between Subsystems (service hosting processes)
- Avoid redundant implementation of business logic
- Fulfil SOA strategy without trading performance
- Investment in business logic must sustain 20 years of technology evolution
- Testability must be built into the software
- Support flexible (high-end -> low-cost) deployments (20 plants)

Cornerstone Slide 29

## Outcome Top 5 SCA contributions

- Application development and deployment infrastructure could evolve in parallel
- High level of re-use achieved
- Service component architecture helped the large project organisation become efficient
- SOA vision could be achieved without compromising performance, resilience and manageability for production-critical plant floor systems
- Broad applicability: From device communication to customer order slotting
- Re-use facilitated through corporate service component library

SCA works well, because it is based on a pragmatic approach to SOA!

Cornerstone Slide 30

## The Toolbox

- Formalized service component modelling (analysis- and design) approach
  - The Business Component Approach + UML
- Governance of service models
- Business Object / Schema / WSDL management and Versioning strategy
- Test automation / TDD integration
- Build System specifically selected and tuned to support the SCA model
- Change Control structure mapped to service modelling approach
- Open Source binding framework (Spring) (instead of high-end SCA tooling)
- Reference application / proof-of-concept
- Extensive toolbox for test-driven development
- Seven full days of training for new team members (includes full TDD training)

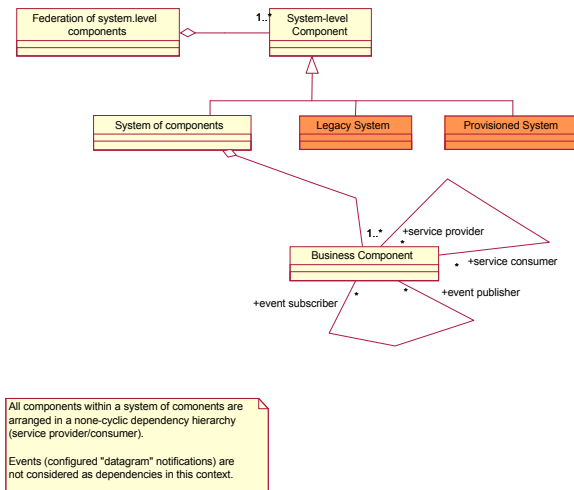
Cornerstone Slide 31

## Service modeling

- Logical Modeling of the Services
  - This is where re-use happens
- Chosen concept:
  - Business Component Approach
  - "Modeling for Service Component Architecture"
- The core is a logical construct named "Business Component"

Cornerstone Slide 32

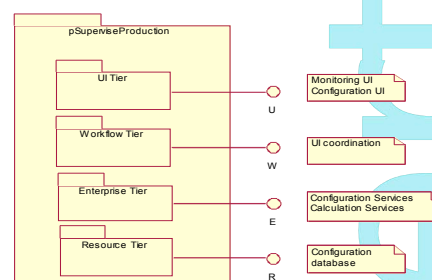
## Business Component Architecture



Cornerstone Slide 33

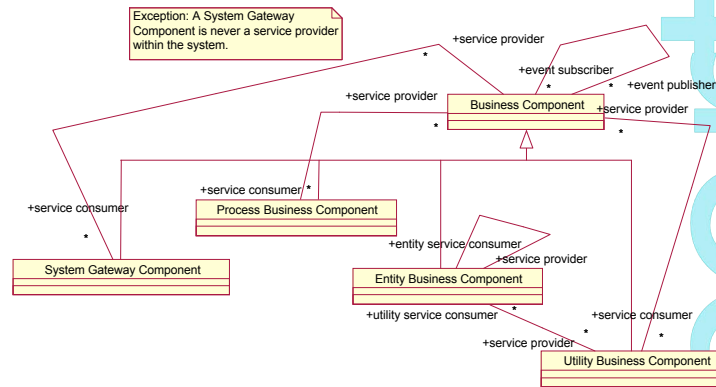
## Autonomy of a Business Component

- A business component is a business-centric, logical construct
- Interactions within a business component are less formal than across business components
  - Pragmatic shortcuts "allowed"
- Base-lining across all tiers
- A business component may have 1 - 4 tiers



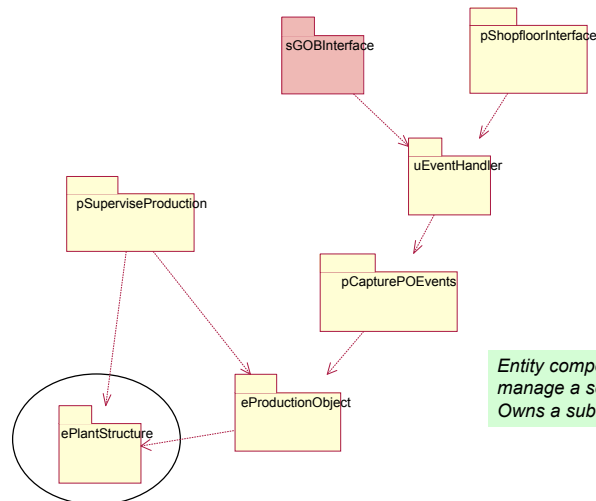
Cornerstone Slide 34

## Business Component Types



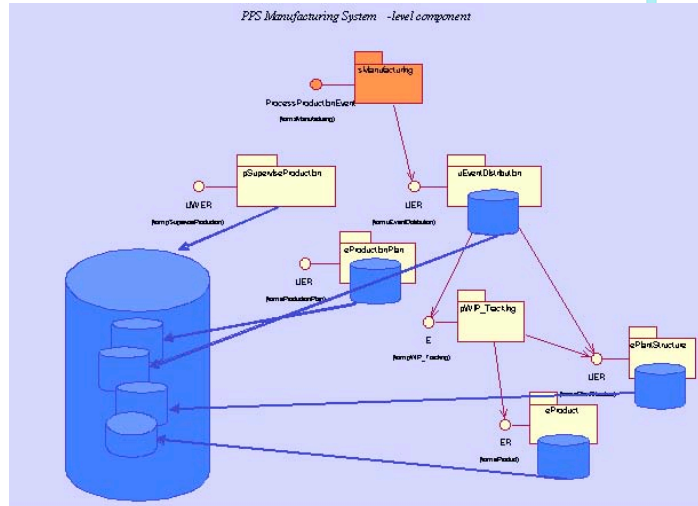
Cornerstone Slide 35

## Entity Business Components



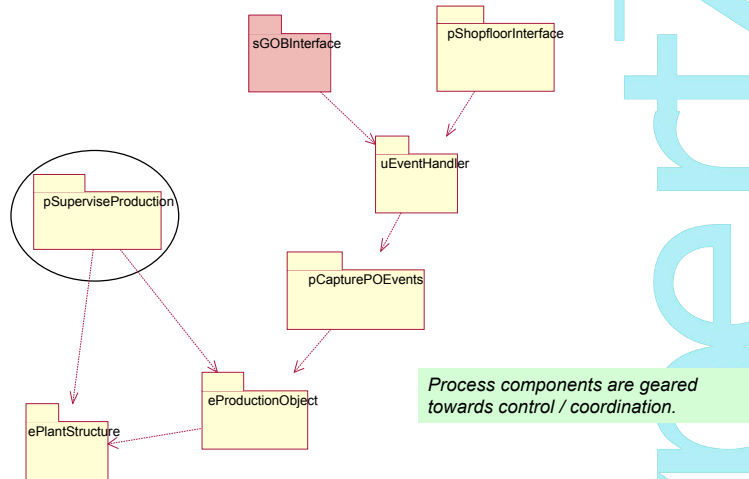
Cornerstone Slide 36

## Business Components and data ownership



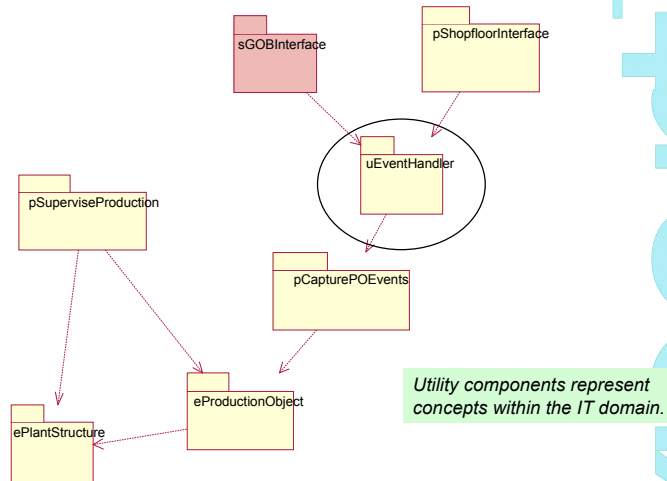
Cornerstone Slide 37

## Process Business Components



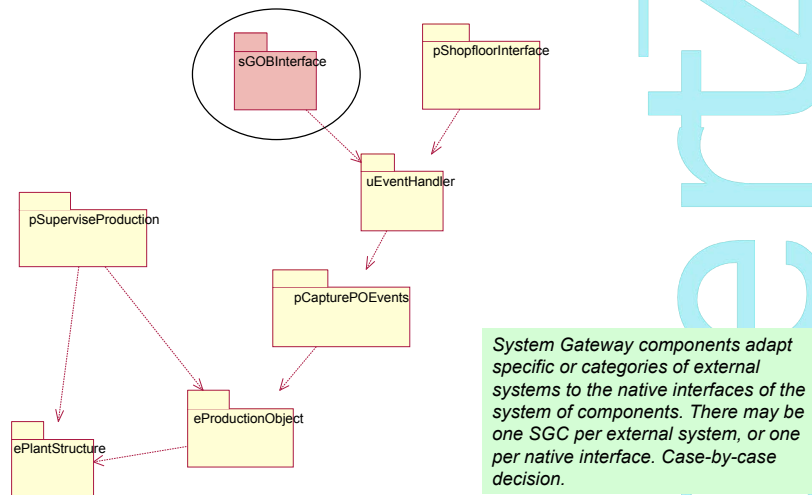
Cornerstone Slide 38

## Utility Business Components



Cornerstone Slide 39

## System Gateway Business Components



Cornerstone Slide 40

## Example of top-level specification of a Business Component

The *uEventHandler* business component represents the concept of a configurable event handling mechanism. It provides the event handling support as well as the configuration support.

### UI tier workflows:

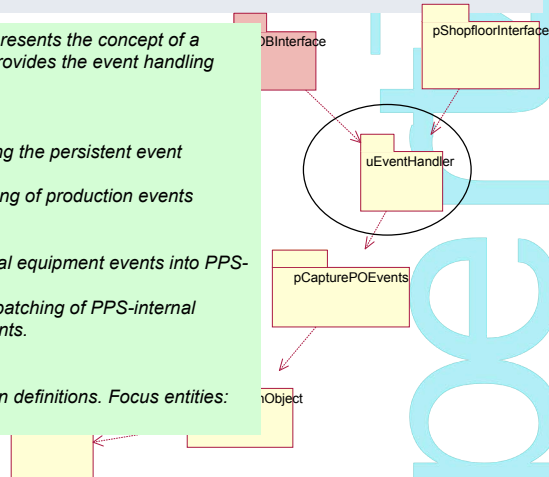
- Configuration web user interface, for defining the persistent event subscriptions.
- Event-driven Info Bus interface for processing of production events

### Enterprise Services:

- Event handler service for converting external equipment events into PPS-internal events
- Event handler service for resolving and dispatching of PPS-internal events to subscribing Info Bus queue endpoints.

### Resource tier:

- Entities for managing persistent subscription definitions. Focus entities: Eventmask, Subscription, Actions



Cornerstone Slide 41

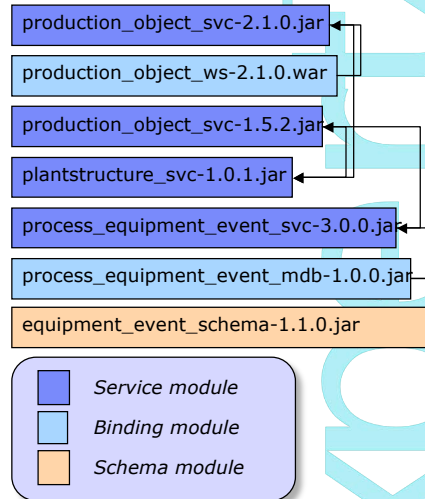
## Repository-based management of service libraries

- Corporate repository of versioned modules
  - Unit of re-use.
- Three types of modules:
  - Service modules, binding modules and schema modules
- Service modules
  - Business logic (service components)
  - Jar files with java classes and spring config files (except infra config)
- Binding modules
  - Required to compensate for lack of SCA-aware tooling and deployment
  - Protocol binding (WS, JMS etc), data binding (message payload -> java classes) and service activation (invoke referenced services in various business modules) for publishing of coarse-grained external ("managed") services.
- Schema modules
  - XML Schemas defining business objects + compiled binding classes
  - The sum the schema modules of a domain represents the business object model upon which exported (runtime) services are modelled. Same status as the data model.

Cornerstone Slide 42

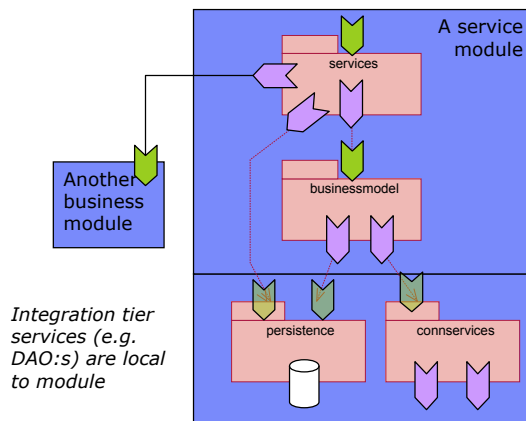
## Module Repository

- The Maven build system is used to manage a repository of versioned binary modules including runtime dependencies between modules
- Module names are unique across the business



Cornerstone Slide 43

## Service module composition (Layered model)



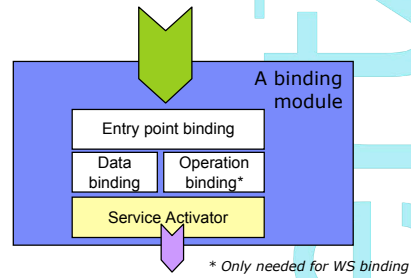
Dependency injection (wiring) is conducted by the Spring framework.

Simplified migration to SCA spec, by standardizing on map able spring features.

Cornerstone Slide 44

## Binding module composition

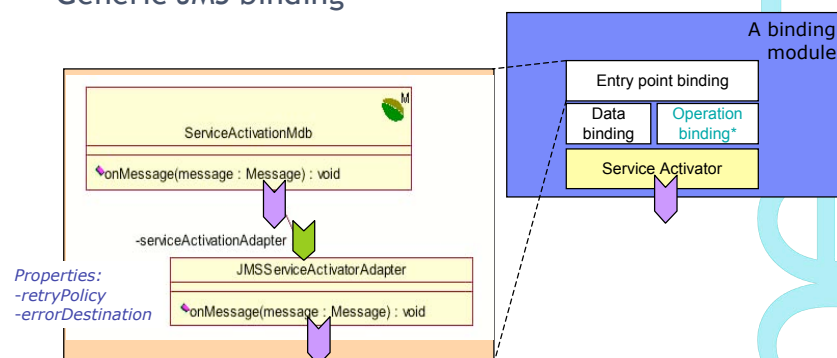
- Publishes a service to the ESB
  - Explicitly defines a networked coarse-grained service (in the classic sense of SOA)
- Binding module
  - Listens to an ESB protocol
  - Conducts data binding
  - Invokes service activator
- Layers of binding modules:
  - Entry point binding (generic component per protocol)
  - (Operation binding: WSDL -> Java stub)
  - Data binding (generic component for JAXB)
  - Service activator (custom component)
- Binary representation
  - WS binding: WAR
  - JMS binding: EJB-JAR with MDB:s only



Cornerstone Slide 45

## Binding module example - JMS entry point

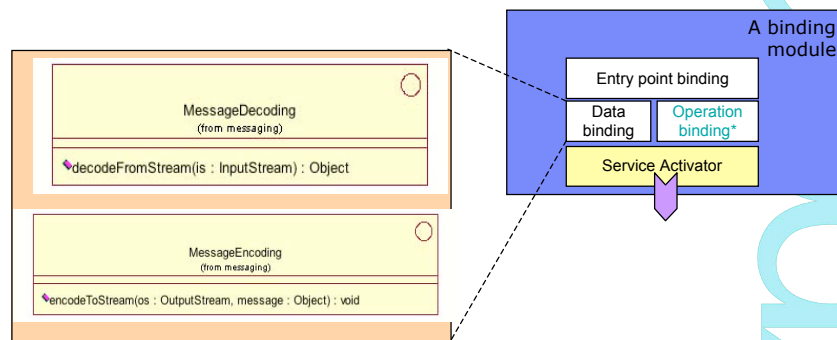
- Generic JMS binding



Cornerstone Slide 46

## Binding module example - JAXB data binding

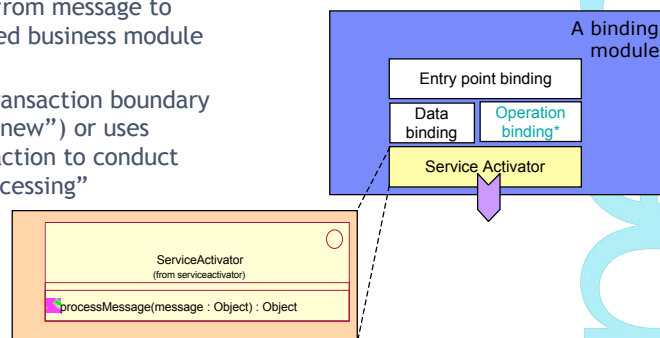
- Data Binding service interfaces
- Generic component (implementation) for JAXB
  - Injected with generated JAXB-factory



Cornerstone Slide 47

## Binding module example - Service Activator

- Custom ServiceActivator component
  - Processes inbound JAXB message
  - Uses data from message to invoke wired business module services
  - Declares transaction boundary (“requires new”) or uses UserTransaction to conduct “batch processing”



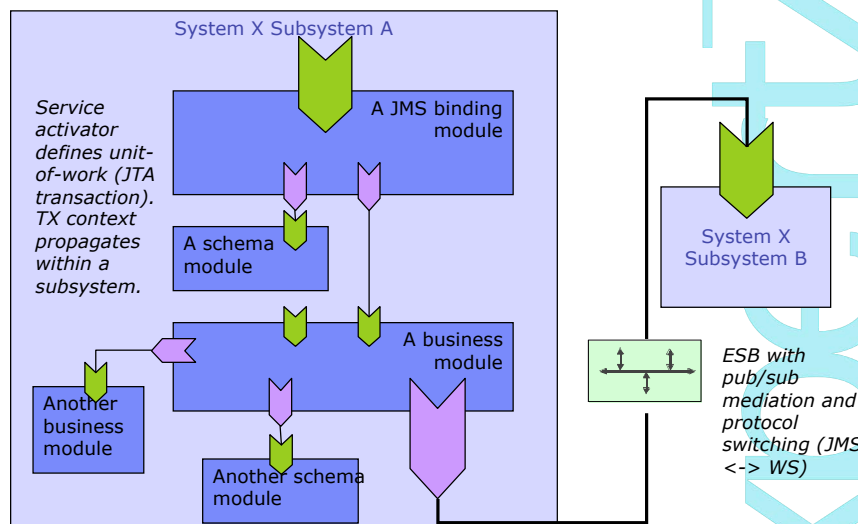
Cornerstone Slide 48

## Schema module

- Models business object types
  - XML schema ComplexType
  - E.g. Production order, Production Object Event
- Used to compose messages
  - Root element for JMS payload
  - WSDL message for WS / SOAP (document/literal)
- Applies versioning strategy
  - Backwards- and forwards interface compatibility through the “any”-strategy
  - Schema + generated JAXB classes in versioned jar-file
    - E.g. production\_object\_schema-1.1.0.jar
    - Build system automatically generates JAXB classes and builds snapshot jar when schema file is updated
- A schema of a schema module may import schemas from other schema modules
  - Integrated into build system!

Cornerstone Slide 49

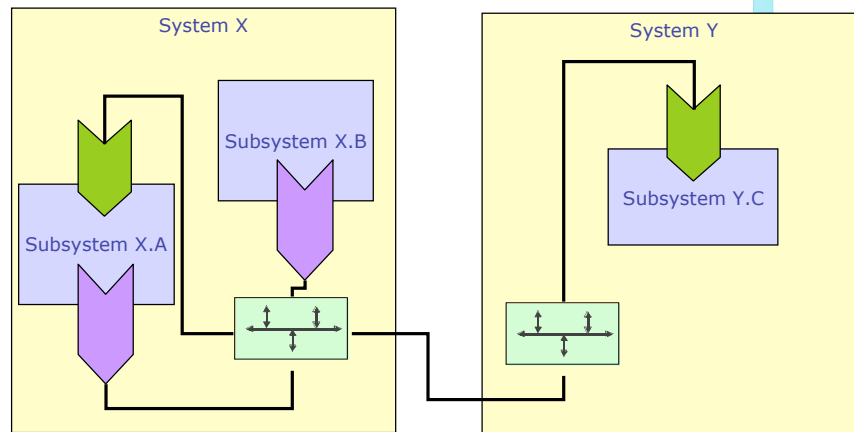
## Example subsystem assembly



Cornerstone Slide 50

## Federation of systems

- SOA domains are linked by connecting the ESB:s



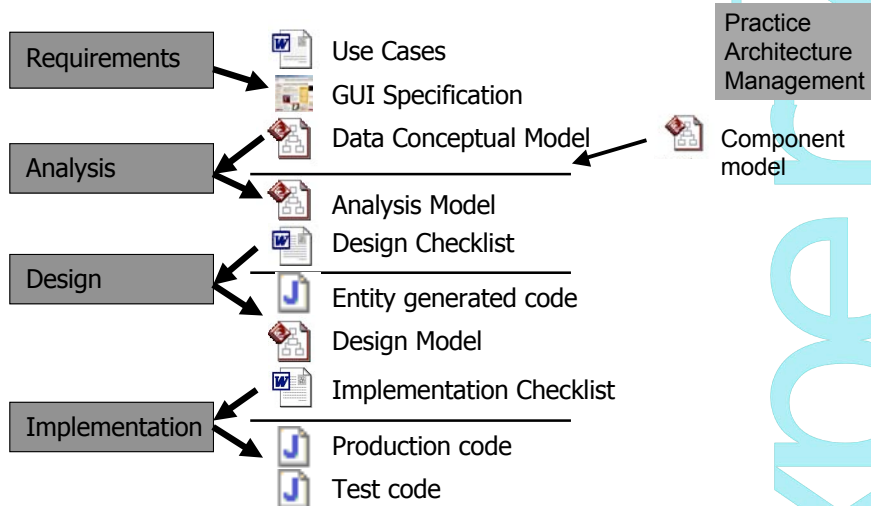
Cornerstone Slide 51

## Governance of service models

- A business component model per System (or domain)
- Integrate business component modelling into the development process
- Corporate team of practice architects (by business area) create business component models together with project solution architects
- Review gates of analysis models

Cornerstone Slide 52

## RUP integration of Business Component Modelling



Cornerstone Slide 53

## Summary

- Service Component Architecture
  - A concept for decoupling of business logic from infrastructure
  - Defines an architecture for building applications and services from re-usable service components
  - Prescribes dependency injection mechanism for "glueing" infrastructure into the logical wires between components
- Business Component Approach
  - Provides a robust and architecture-centric modelling approach for Service Component Architecture
- Applied with excellent outcome in enterprise set-up
  - Still light-weight enough to support small projects
  - Pragmatic approach based on mature open source technology

Cornerstone Slide 54

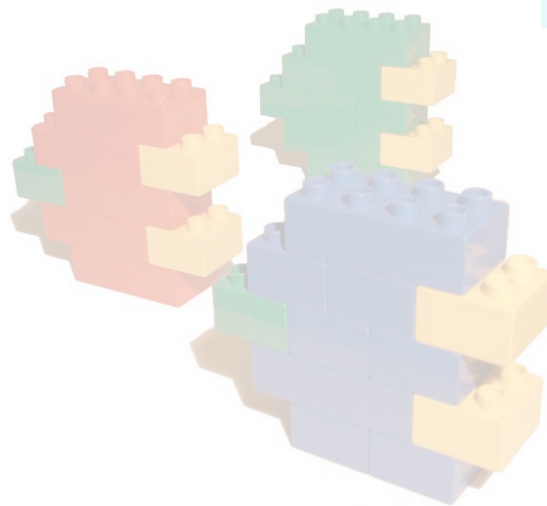
## SCA get-started kit

- Would you like to build on our customers success?
- Packaged SCA best-practice
  - Introductory workshop
  - Service modelling
  - Business Object / Schema / WSDL management and Versioning strategy, internal versus external interface strategy
  - Test automation / TDD integration
  - SCA Build System set-up (binary version / dependency management)
  - Change Control structure
  - Reference application
  - Training program
- Open Source- or High-End tooling best-practice
  - Guidelines for applying Spring in an SCA context
  - WebSphere Process Server and WID

Cornerstone Slide 55

## Software Lego has arrived!

- Q&A



Cornerstone Slide 56

xpoertzc xpoertzc