Service Component Architecture

An overview

Pascal ANDRE

AeLoS / LINA – UMR CNRS 6241
{Firstname.Lastname}@univ-nantes.fr

Exposés AeLoS

Outline

Outline

Introduction / context

Study Context : Component and Services in Kmelia

- assembly rules
- service promotion
- multi-level contracts
- marry component and services
- verification aspects
- implementation issues

Compare with other models :

- compare with close related models
- situate Kmelia in the domain
- import/export ideas

Today : Service Component Architecture (SCA)

Open SOA

OSOA Collaboration

OSOA `http://www.osoa.org/display/Main`

- Collaboration
  - *The Open SOA Collaboration represents an informal group of industry leaders that share a common interest : defining a language-neutral programming model that meets the needs of enterprise developers who are developing software that exploits Service Oriented Architecture characteristics and benefits.*
  - *The Collaboration is not a Standards Body ; it is a set of vendors who wish to innovate rapidly in the development of this programming model and to deliver Specifications to the community for implementation.*
  - *These specifications are made available to the community on a Royalty Free basis for the creation of compatible implementations. When mature, the intent is to hand these specifications over to a suitable Standards Body for future shepherding.*
- Projects
  - Service Data Objects (SDO)
  - Service Component Architecture (SCA)
- Supporters (*not well-known*)

  `http://www.osoa.org/display/Main/Current+OSOA+Supporters+Community`

Service Data Objects (SDO)

Service Data Objects aims to provide consistent means of handling data within applications, whatever its source or format may be. SDO provides a way of unifying data handling for databases and for services. SDO also has mechanisms for the handling of data while detached from its source.

- Service Data Objects is a technology that allows heterogeneous data to be accessed in a uniform way.
  - Using SDO, application programmers can uniformly access and manipulate data from heterogeneous data sources, including relational databases, XML data sources, Web services, and enterprise information systems.
- The SDO specification was originally developed in 2004 as a joint collaboration between BEA and IBM and approved by the Java Community Process.
- Specifications for Java, C++, C, Cobol
- Version 2.0 of the specification was introduced in November 2005 as key part of the Service Component Architecture.

others JDO, EMF, JAXB and ADO.NET.

Sources :
`http://www.osoa.org/display/Main/Service+Data+Objects+Home`

`http://en.wikipedia.org/wiki/Service_Data_Objects`

Service Component Architecture

Service Component Architecture aims to provide a model for the creation of service components in a wide range of languages and a model for assembling service components into a business solution - activities which are at the heart of building applications using a service-oriented architecture.

- Partner vendors include :
  - the original members : BEA Systems, IBM, IONA Technologies, Oracle Corporation, SAP AG, Sybase, Xcalia and Zend Technologies
  - the additional members on July 26, 2006 : Cape Clear, Interface21, Primeton Technologies, Progress Software, Red Hat, Rogue Wave Software, Software AG, Sun Microsystems and TIBCO Software.
  - Siemens AG, on September 18, 2006.
- Short history
  - first published as a 0.9 version in November 2005 (Assembly Specification, Java and C++ Client and Implementation Specification)
  - improvements in july 2006 and new specifications (bindings and policies, service network)
  - a set of Final V1.0 Specifications were published in March 2007 (OASIS leads now)

`http://www.osoa.org/display/Main/Service+Component+Architecture+Home`

`http://en.wikipedia.org/wiki/Service_component_architecture`

Outline

Service Component Architecture

Specifications set (2007)

1. Assembly Model
2. Policy Framework
3. Transaction Policy
4. Java Common Annotations and APIs
5. Java Component Implementation
6. Spring Component Implementation
7. BPEL Client and Implementation
8. C++ Client and Implementation
9. Web Services Binding
10. JMS Binding
11. EJB Session Bean Binding
12. COBOL Client and Implementation
13. C Client and Implementation
14. JCA Binding
15. Java EE Integration Specification (2008)
16. Assembly Extensions for Event Processing and Pub/Sub (2009)

Service Component Architecture

Specifications set (2007)

1. Assembly Model 1 - basis
2. Policy Framework
3. Transaction Policy
4. Java Common Annotations and APIs
5. Java Component Implementation
6. Spring Component Implementation
7. BPEL Client and Implementation
8. C++ Client and Implementation
9. Web Services Binding
10. JMS Binding
11. EJB Session Bean Binding
12. COBOL Client and Implementation
13. C Client and Implementation
14. JCA Binding
15. Java EE Integration Specification (2008)
16. Assembly Extensions for Event Processing and Pub/Sub (2009)

Service Component Architecture

Specifications set (2007)

1. Assembly Model 1 - basis
2. Policy Framework 2 - NF requirements (constraints, capabilities and QoS expectations)
3. Transaction Policy 2 - ACID or not ACID (atomicity, consistency, isolation, durability)
4. Java Common Annotations and APIs
5. Java Component Implementation
6. Spring Component Implementation
7. BPEL Client and Implementation
8. C++ Client and Implementation
9. Web Services Binding
10. JMS Binding
11. EJB Session Bean Binding
12. COBOL Client and Implementation
13. C Client and Implementation
14. JCA Binding
15. Java EE Integration Specification (2008)
16. Assembly Extensions for Event Processing and Pub/Sub (2009)

Service Component Architecture

Specifications set (2007)

1. Assembly Model 1 - basis
2. Policy Framework 2 - NF requirements (constraints, capabilities and QoS expectations)
3. Transaction Policy 2 - ACID or not ACID (atomicity, consistency, isolation, durability)
4. Java Common Annotations and APIs 3 - Toward implementation
5. Java Component Implementation
6. Spring Component Implementation
7. BPEL Client and Implementation
8. C++ Client and Implementation
9. Web Services Binding
10. JMS Binding
11. EJB Session Bean Binding
12. COBOL Client and Implementation
13. C Client and Implementation
14. JCA Binding
15. Java EE Integration Specification (2008)
16. Assembly Extensions for Event Processing and Pub/Sub (2009)

Service Component Architecture

Specifications set (2007)

1. Assembly Model 1 - basis
2. Policy Framework 2 - NF requirements (constraints, capabilities and QoS expectations)
3. Transaction Policy 2 - ACID or not ACID (atomicity, consistency, isolation, durability)
4. Java Common Annotations and APIs 3 - Toward implementation
5. Java Component Implementation 4 - from 1 + 3
6. Spring Component Implementation
7. BPEL Client and Implementation
8. C++ Client and Implementation
9. Web Services Binding
10. JMS Binding
11. EJB Session Bean Binding
12. COBOL Client and Implementation
13. C Client and Implementation
14. JCA Binding
15. Java EE Integration Specification (2008)
16. Assembly Extensions for Event Processing and Pub/Sub (2009)

Service Component Architecture

Assembly Model Overview

- programming model for building applications and solutions based on a Service Oriented Architecture.
- based on the idea that business function is provided as a series of services, which are assembled together to create solutions that serve a particular business need
- aims to encompass a wide range of technologies for service components
- allows for a wide variety of implementation technologies, including "traditional" programming languages such as Java, C++, and BPEL, but also scripting languages such as PHP and JavaScript and declarative languages such as XQuery and SQL
- Composites can contain components, services, references, property declarations, plus the wiring that describes the connections between these elements.
- Composites are deployed within an SCA Domain. An SCA Domain typically represents a set of services providing an area of business functionality that is controlled by a single organization (*e.g.* cover all financial related function.
- These XML files define the portable 62 representation of the SCA artifacts.

Service Component Architecture

Preliminary remarks

1. emerging standart CBSE/SOA (since 2007) - *few scientific background* *(as far as I know)*
2. specifications $\Longrightarrow$ open models (e.g. dynamics)
3. semantics attempts [Fiadeiro et al., 2006] and [Ding et al., 2008]
4. in [Krämer, 2008]
   - BPEL activity orchestration
   - contracts introduction
   - component compatibility
5. no connectors
6. refinement, encapsulation by composition
7. remains a non-hierarchical service model

Service Component Architecture

Preliminary remarks

1. emerging standart CBSE/SOA (since 2007) - *few scientific background* *(as far as I know)*
2. specifications $\Longrightarrow$ open models (e.g. dynamics)
3. semantics attempts [Fiadeiro et al., 2006] and [Ding et al., 2008]
4. in [Krämer, 2008]
   - BPEL activity orchestration
   - contracts introduction
   - component compatibility
5. no connectors
6. refinement, encapsulation by composition
7. remains a non-hierarchical service model
8. XML everywhere

Service Component Architecture

Preliminary remarks

1. emerging standart CBSE/SOA (since 2007) - *few scientific background* *(as far as I know)*
2. specifications $\Longrightarrow$ open models (e.g. dynamics)
3. semantics attempts [Fiadeiro et al., 2006] and [Ding et al., 2008]
4. in [Krämer, 2008]
   - BPEL activity orchestration
   - contracts introduction
   - component compatibility
5. no connectors
6. refinement, encapsulation by composition
7. remains a non-hierarchical service model
8. XML everywhere

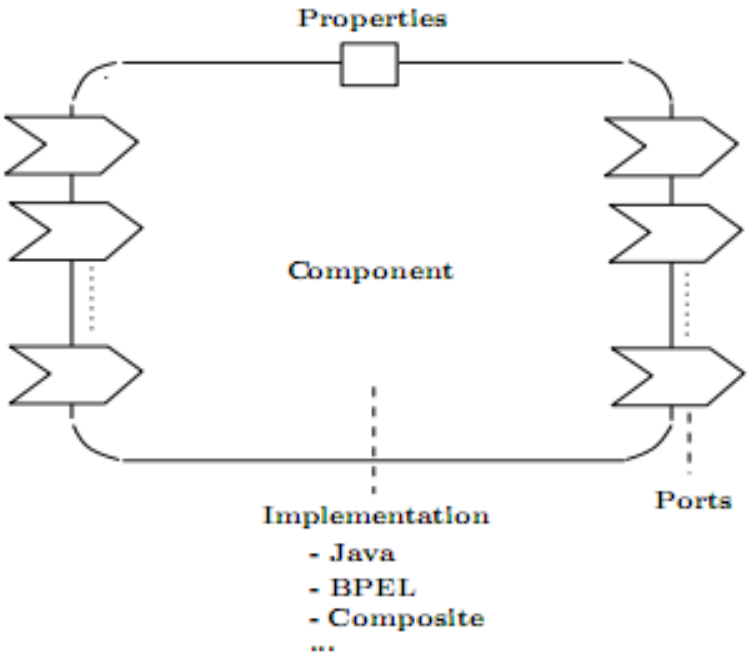
Books

- Understanding SCA Service Component Architecture [Marino and Rowley, 2009]
- SOA for the Business Developer [Margolis, 2007]
- Tuscany SCA in Action [Laws et al., 2009]

Outline

SCA Concepts

Component specification/implementation



source : [Ding et al., 2008]

- provide and consume service : operation activities
- interaction : message exchange
- port = adressable interfaces (sync/async)
  - `service port` (prov) | Kml direction
  - `reference port` (req) | reverse
- *port activities to describe the component dynamic behavior, the basic activity of which is assumed to be the message exchange between two ports.* [Ding et al., 2008]

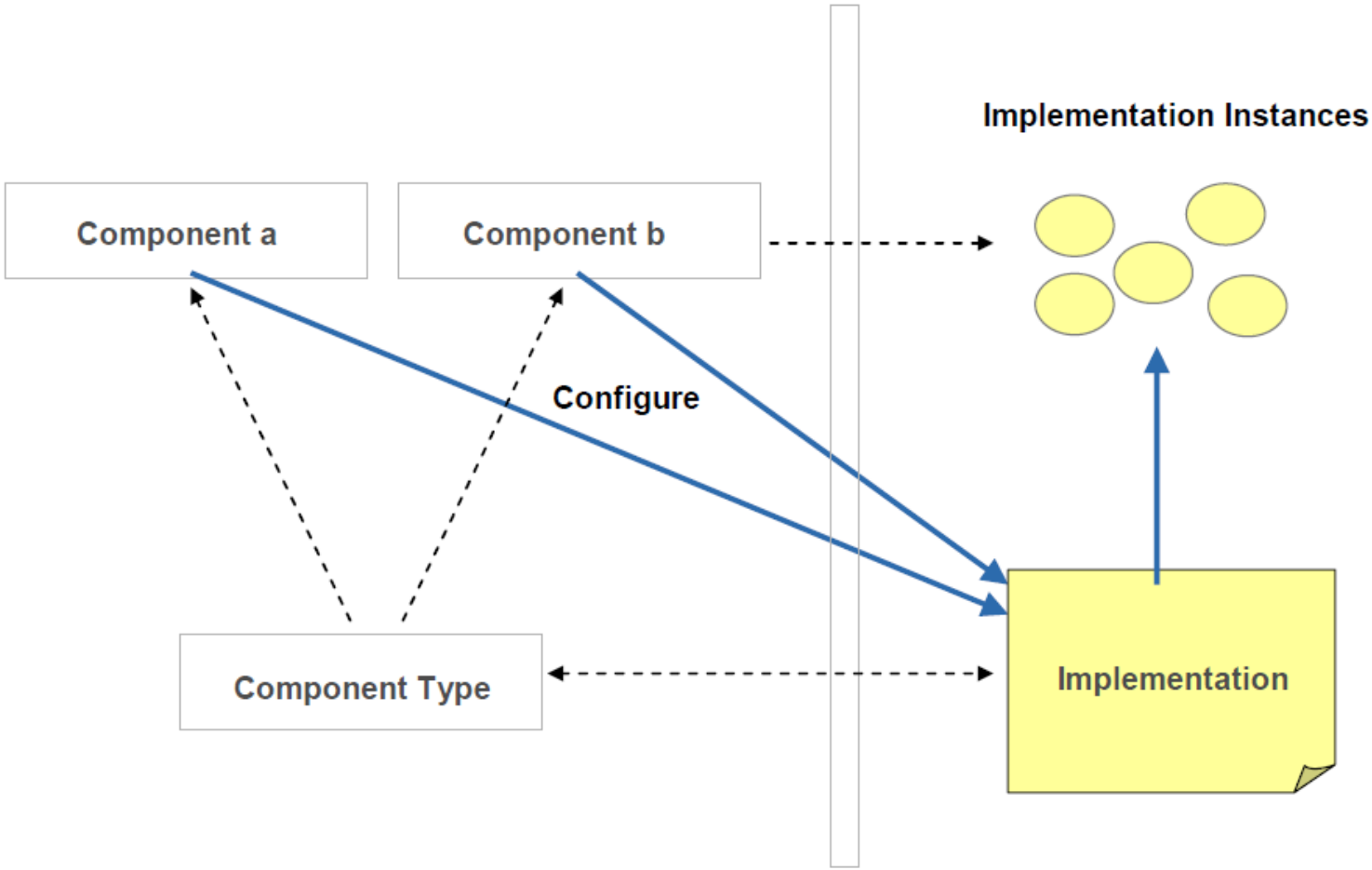
SCA Concepts

Component Interface

1. Define one or more business functions (service operations)
2. Each operation has zero or one request (input) message and zero or one response (output) message.
3. The request and response messages may be simple types such as a string value or they may be complex types.
4. Remotable Interfaces (`@Remotable` annotation in Java)
5. Bidirectional Interfaces ($\rightsquigarrow$ peer-to-peer)
6. Conversational Interfaces ($\rightsquigarrow$ protocols)

SCA Concepts

Component Type

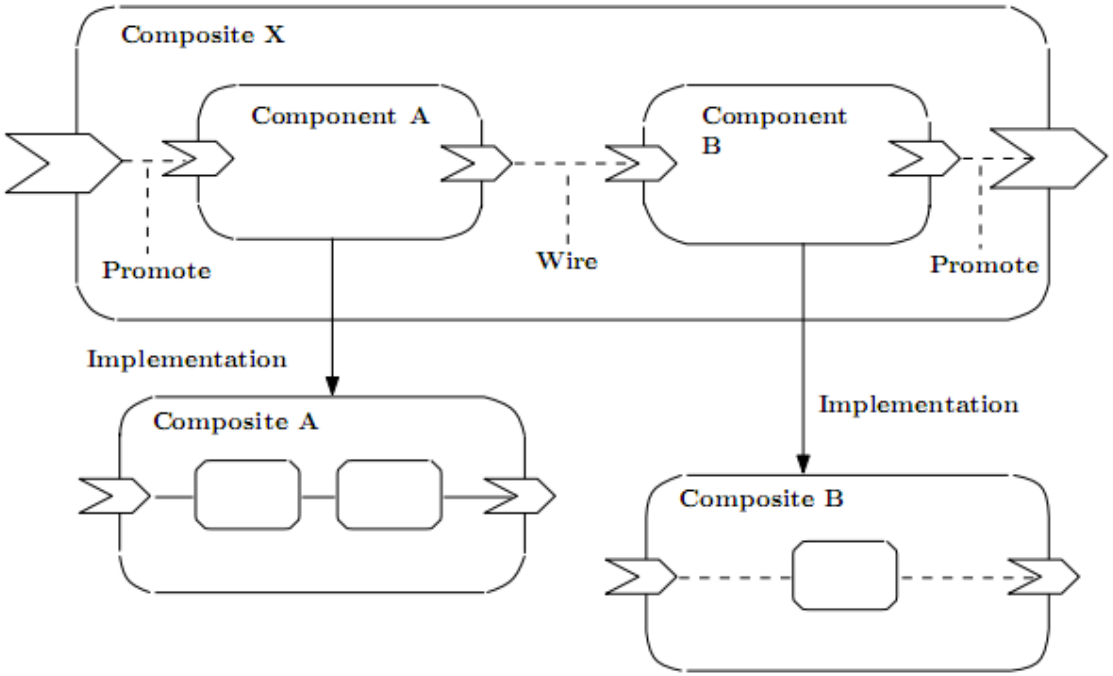


A component type represents the configurable aspects of an implementation (calculated).

source : [OSOA, 2007]

SCA Concepts

Architecture and Composite



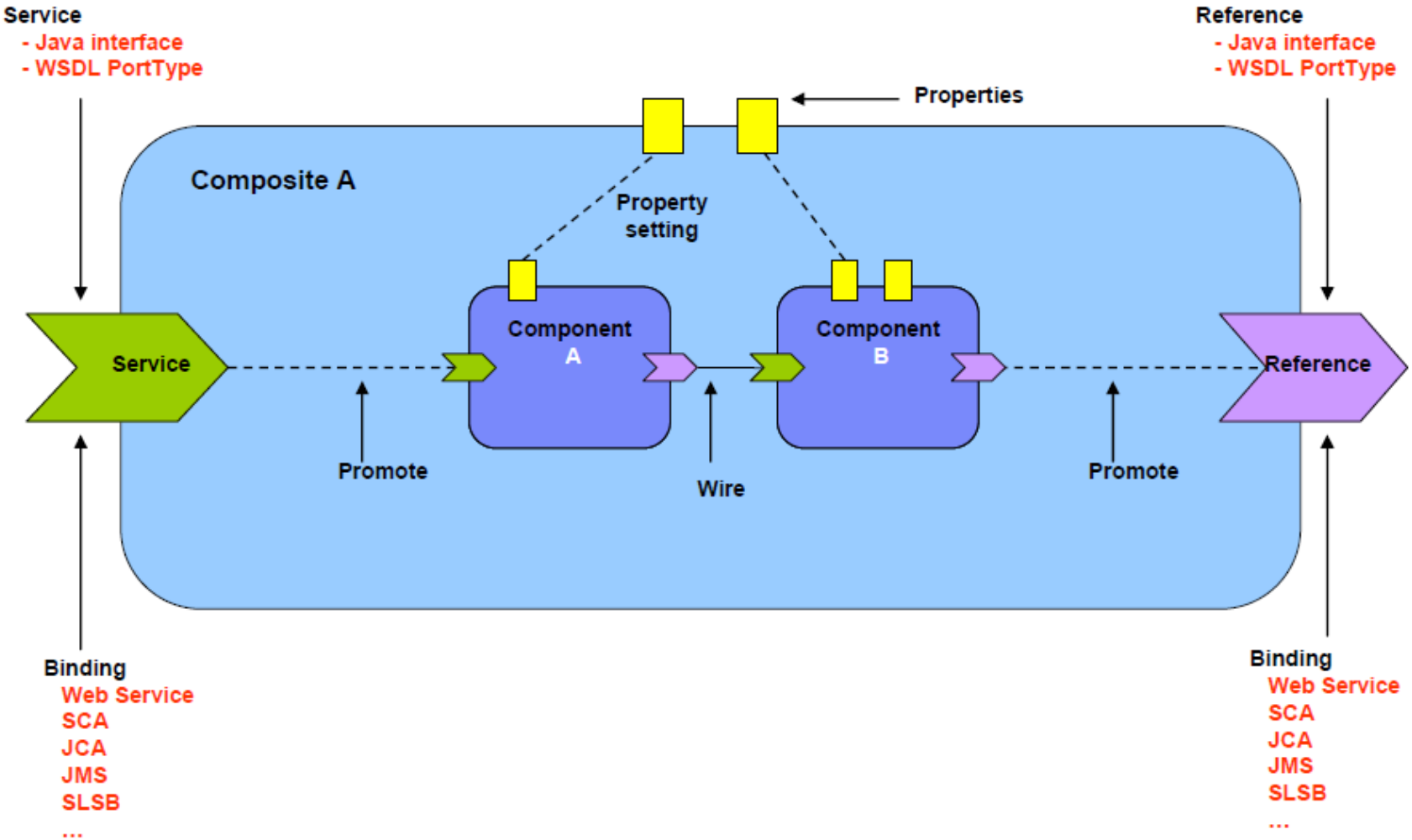
a composite is a component with subcomponents ($G \neq \emptyset$)

- promotion / wire / autowire (implicit wire research)
- simple port binding : no transformation
- no variable promotion
- implicit component composition on ports

source : [Ding et al., 2008]

SCA Concepts

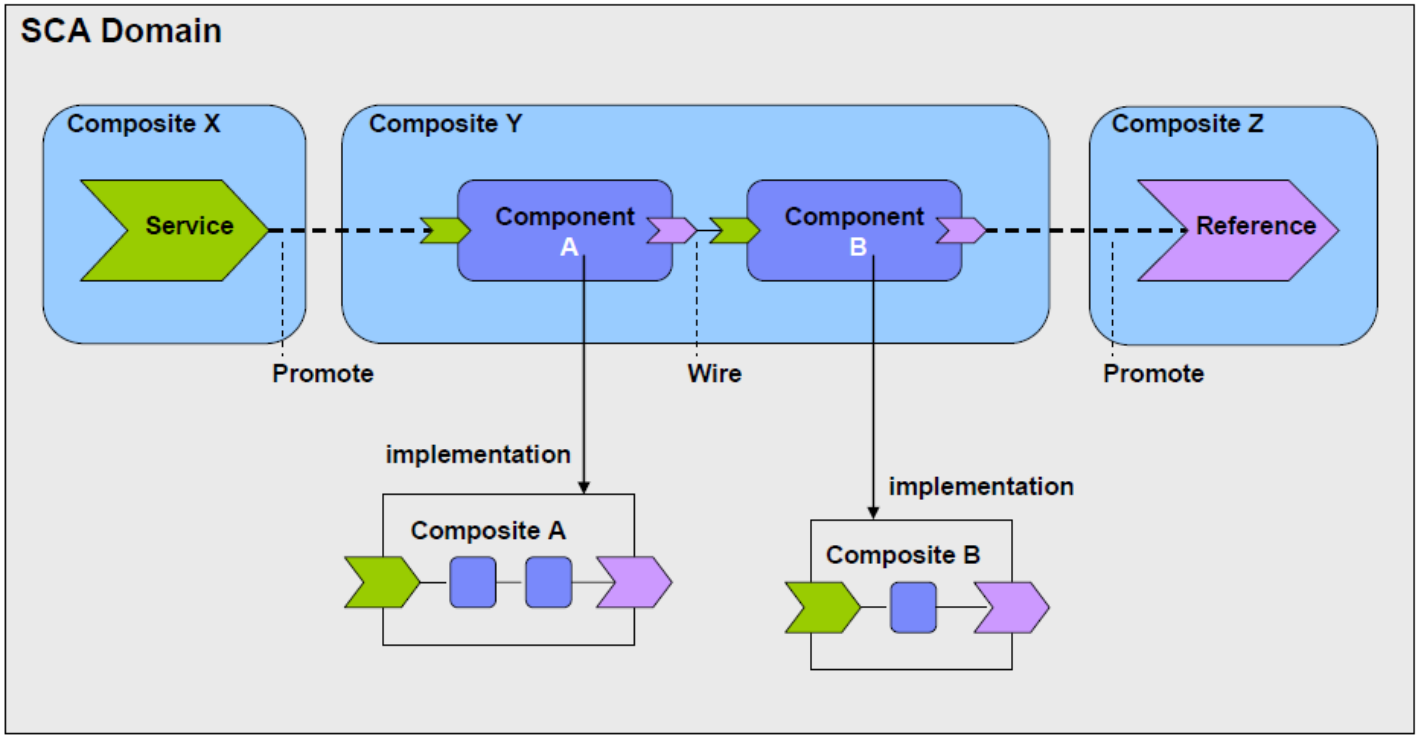
Composite Diagram



source : [OSOA, 2007]

SCA Concepts

Domain Diagram

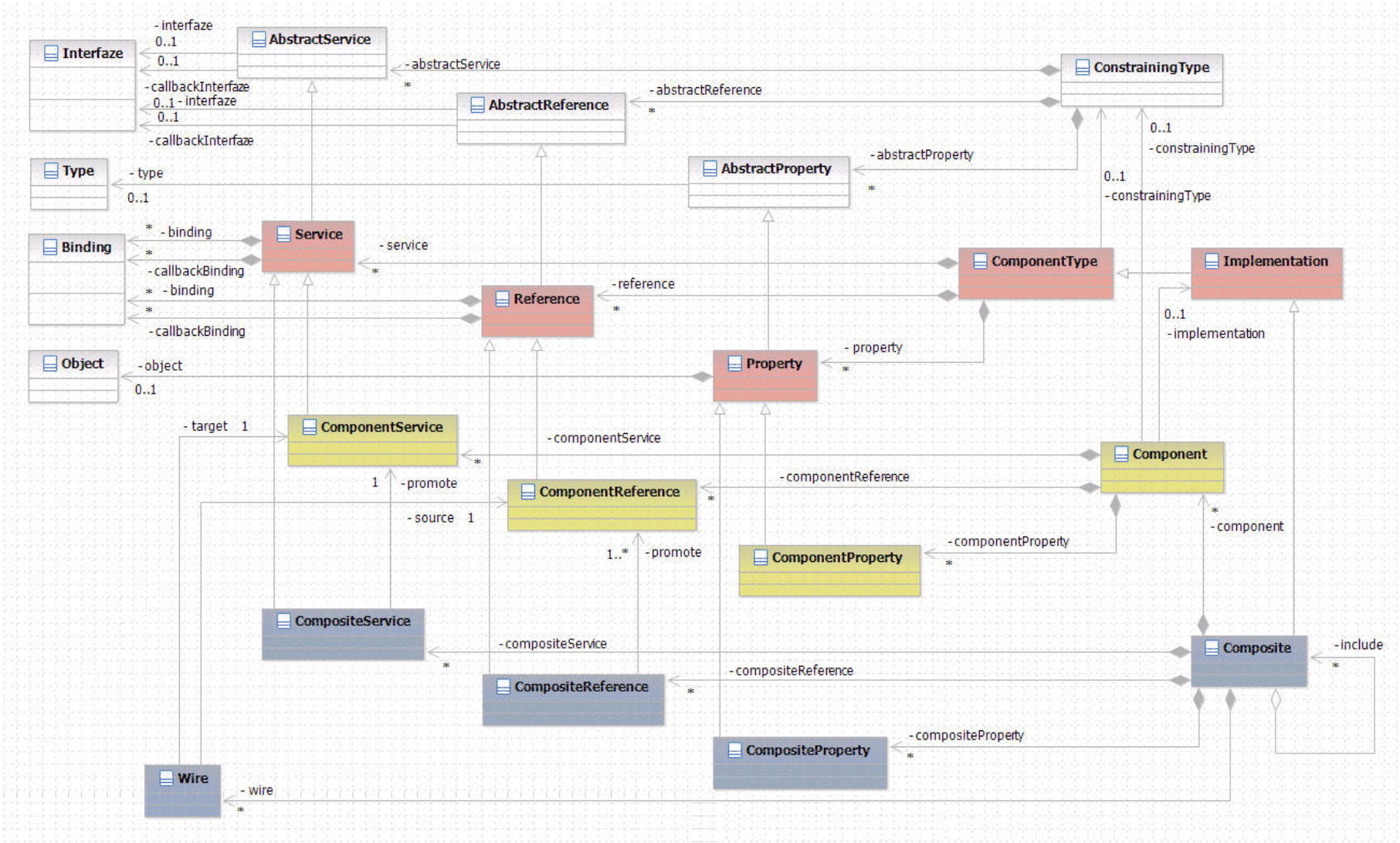


An SCA Domain represents a complete runtime configuration, potentially distributed over a series of interconnected runtime nodes. (cf URI domains, packages)

- A virtual domain-level composite whose components are deployed and running
- A set of installed contributions that contain implementations, interfaces ...
- A set of logical services for manipulating the set of contributions and the virtual domain level composite.

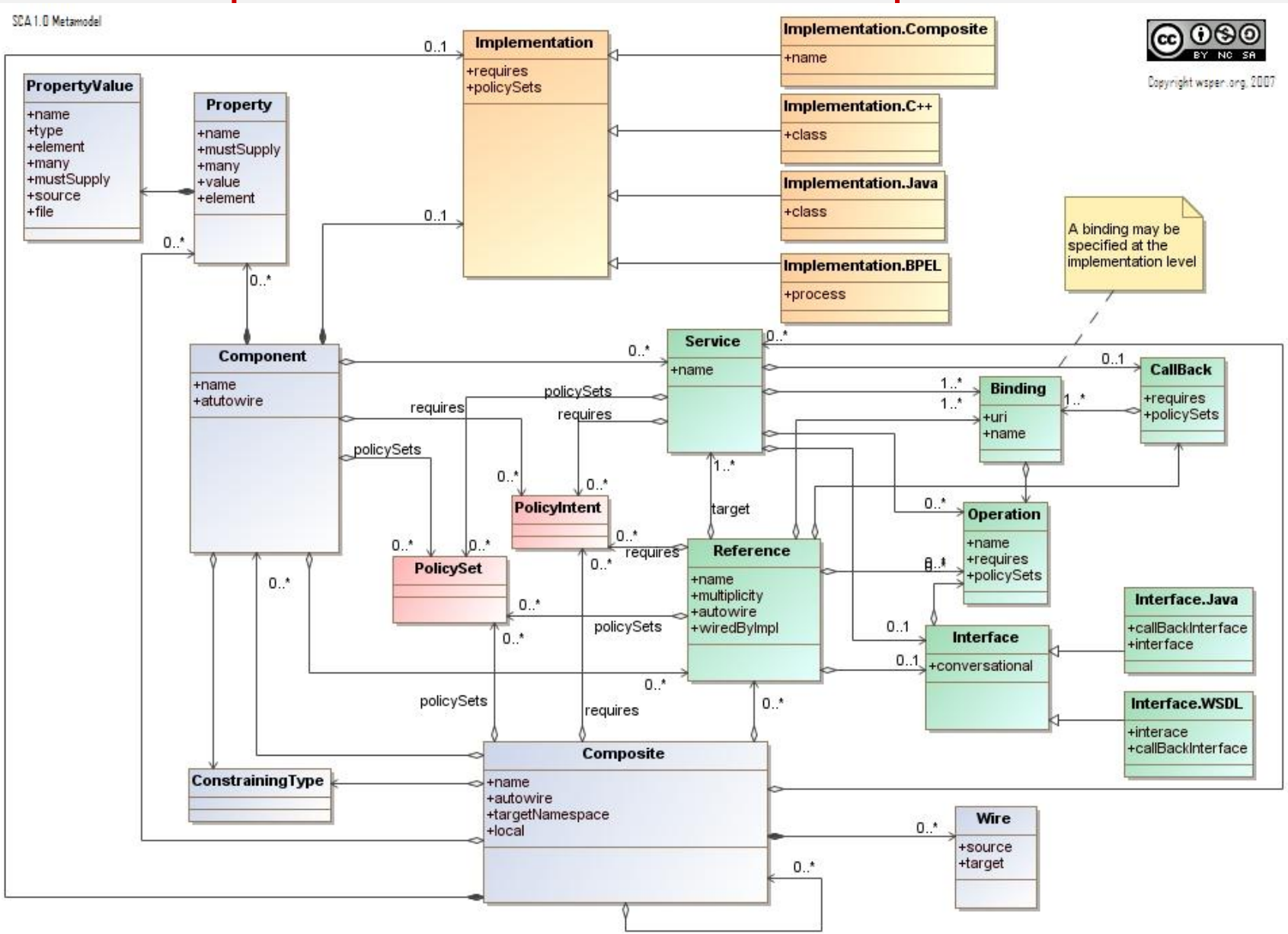
source : [OSOA, 2007]

Service Component Architecture - metamodel



source : `http://www.osoa.org/display/Main/SCA+Expressed+as+a+UML+Model`

Service Component Architecture - concepts



SCA Model

Comparing the Service Component Architecture and Fractal Component Model

	SCA	Fractal
Abstractions	Component	Component
	Composite component	Composite component
	Implementation	Primitive component
	Service	Server interface
	Reference	Client interface
	Property	Attribute
	Wire	Binding

Model

	SCA	Fractal
Hierarchical composition	+	+
Component type definition	+	+
Shared components	-	+
Connectors	±	-
Subtype definition	-	+

Development

	SCA	Fractal
Development from scratch	+	+
Reuse and integration of non-component code	+	±
Reuse of components	+	±

Applicability

	SCA	Fractal
Distributed apps.	+	+
Heterogeneous apps.	+	-
Enterprise apps.	+	±
Operating sys. development	-	+

RunTime

	SCA	Fractal
Runtime environment	-	+
Dynamic architecture changes	-	+

source : [Hnetynka et al., 2010]

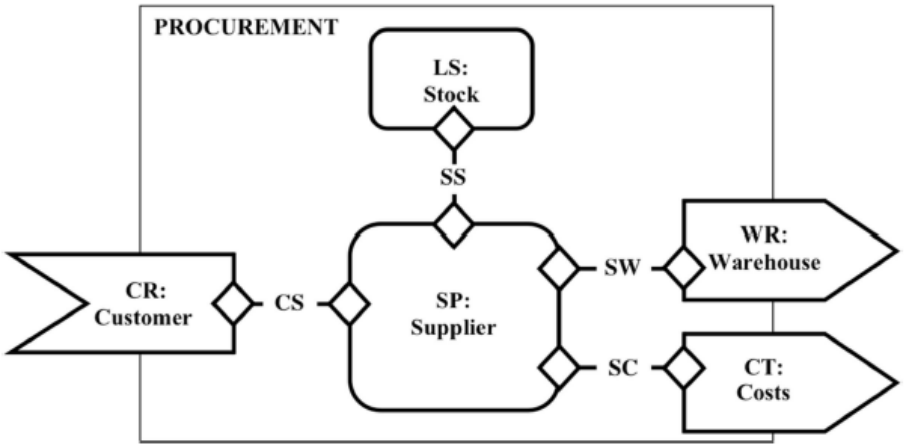
SCA Formal Models

Approaches

1. A Formal Approach to Service Component Architecture [Fiadeiro et al., 2006]
2. A Rigorous Model of Contract-Based Service Component Architecture [Du et al., 2008]
3. A Rigorous Model of Service Component Architecture [Ding et al., 2008]

SCA Formal Models 1/3

[Fiadeiro et al., 2006]



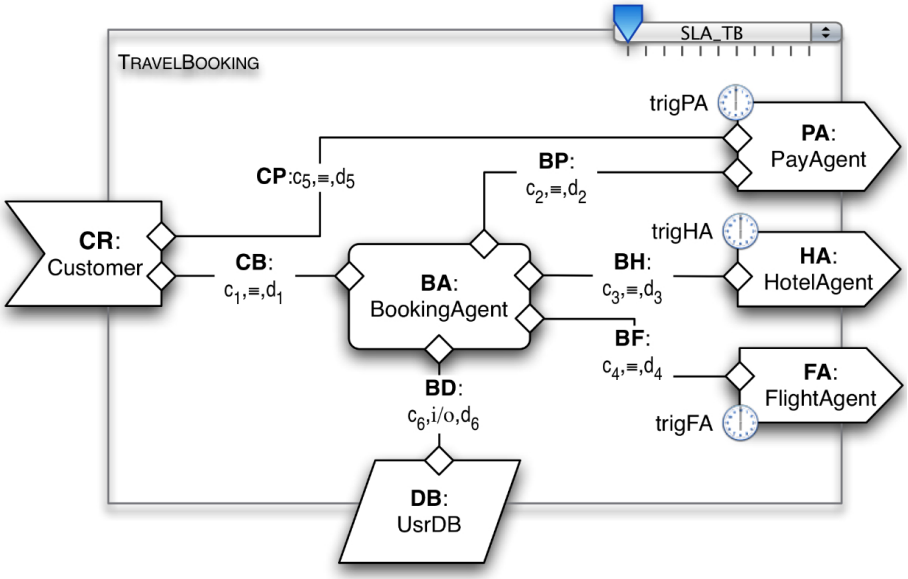
- SENSORIA Reference Modelling language for comPosition (EU project) SRML-P
- module : business process
- component : coordinate the business process

- service - interface : business roles
- language for interaction : event (msg, reply, commit, cancel,...), bLTL,
- orchestration
  - `business protocol` (customer) | Kml direction
  - `interaction protocol` (wire) | reverse
- Eclipse Editor and Modelling Environment for SRML-P
- Formal : algebraic formalisation of modules
- verification ?

SCA Formal Models 1/3 contd

[Fiadeiro et al., 2006]

Model checking via SRML-P [Abreu et al., 2009, Abreu, 2009]

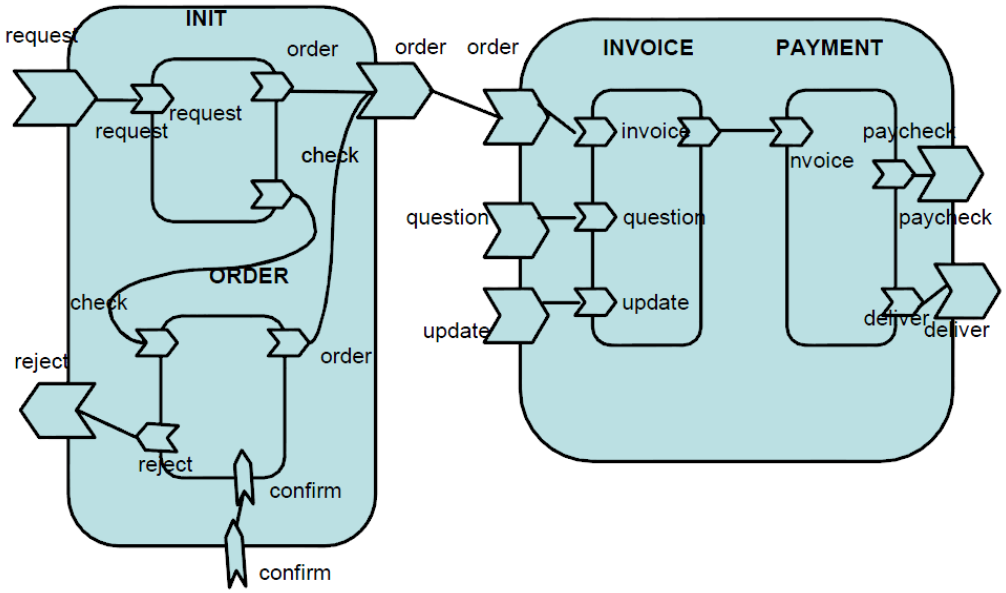


- SRML-P, not stictly SCA (no related SCA translation)

- UML statecharts translation
- UCTL logic (Action CTL + branching time)
- UMC (Pisa) model-checker - details in [Abreu, 2009]
  - `http://fmtlab.isti.cnr.it/umc/V3.7/umc.html`
  - not tested
  - no demo
- see also COWS *Calculus for Orchestration of Web Services*

SCA Formal Models 2/3

[Du et al., 2008]

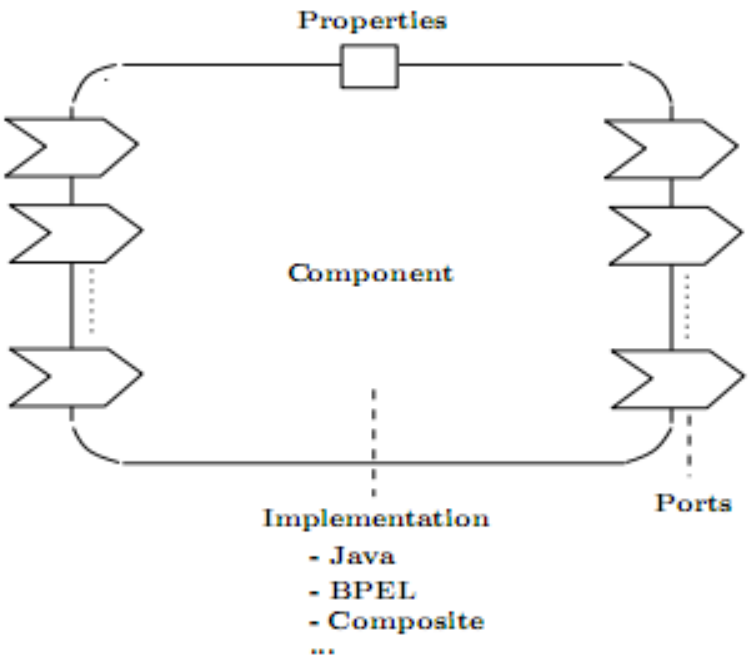


- "rigourous model" (rCOS)
- design by contract
- contract in port interfaces (sync/async)
  - operations specs (signature, guard, *reactive design*)
  - behavior protocols
- contract behavior : protocol + failures + divergences
- contract consistency $\rightarrow$ trace

SCA Formal Models 3/3

[Ding et al., 2008]

one shared author with [Du et al., 2008]



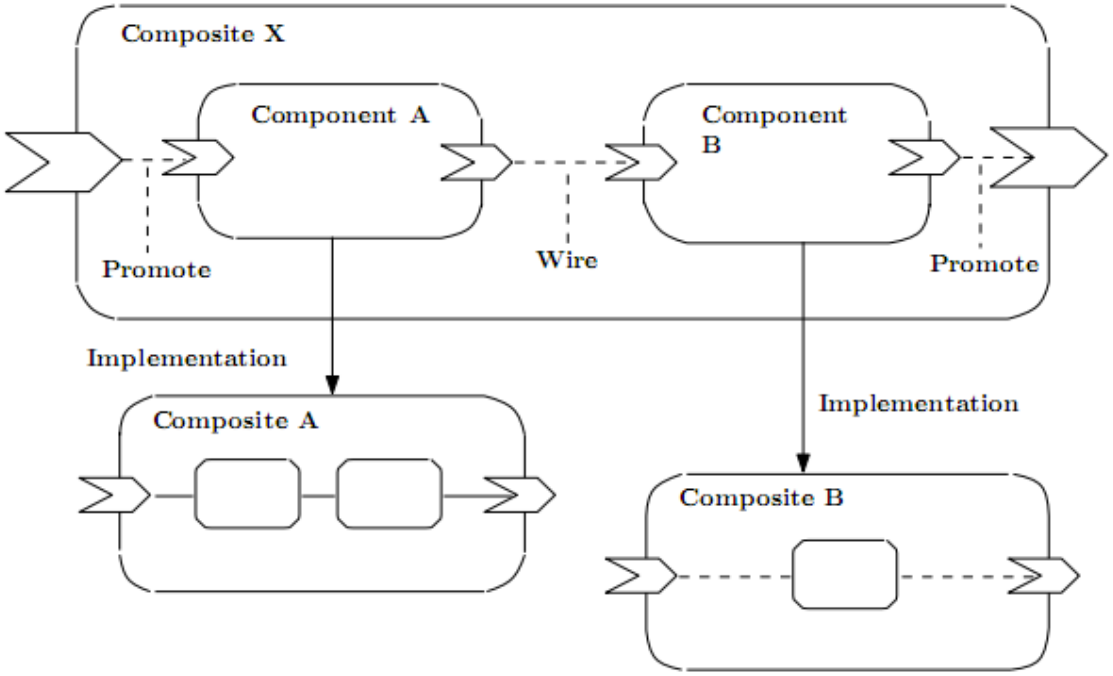
- SC = configured instance of a component implementation

- service : operation activities / business function
- interaction (message exchange) and business flow (msg exchg flow)
- port = adressable interfaces (sync/async)
  - `service port` (prov)
  - `reference port` (req)

  reverse Kml direction
- *port activities to describe the component dynamic behavior, the basic activity of which is assumed to be the message exchange between two ports.*
  [Ding et al., 2008]

SCA Formal Models 3/3 contd

[Ding et al., 2008]

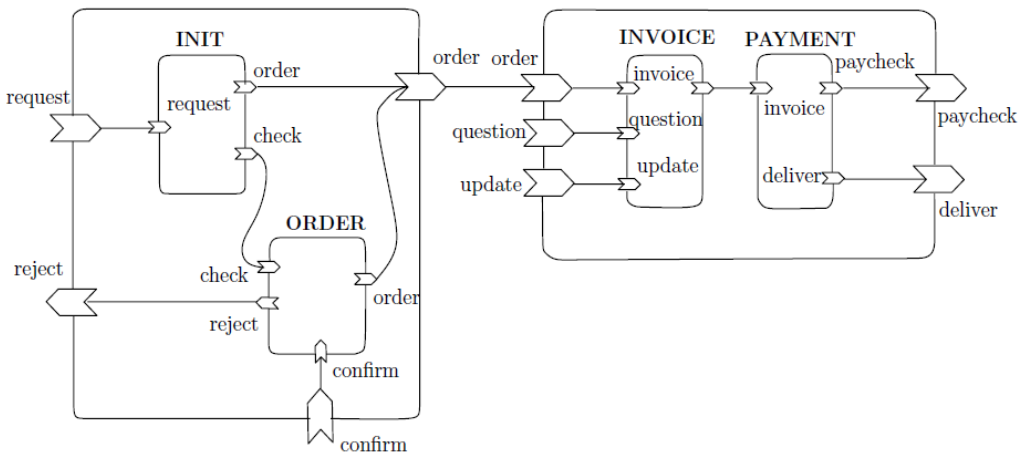


a composite is a component with subcomponents ($G \neq \emptyset$)

- promotion / wire
- simple port binding : no transformation
- implicit component composition on ports

SCA Formal Models 3/3 contd

[Ding et al., 2008]



- dynamic calculus ⇒ port activities
  - behaviour expression
  - LTS operational semantics
- process composition
- promotion operators novariant ? ?
- analysis (Design/Net, SPIN...)
- Model Checking Service Component Composition by SPIN [Ding et al., 2009] not accessible reference

Outline

SCA Examples

- SCA Building your first application - simplified BigBank [OSOA, 2005] Tutorial
- Assembly of Business Systems Using Service Component Architecture [Karmarkar and Edwards, 2006] Explanation
- Component meets service : what does the mongrel look like ? [Krämer, 2008] Comparison
- A Service Component Deployment Architecture for e-Banking [Ruiz et al., 2008]
- Service Component Architecture for Vending Machine System in Cloud Computing Infrastructure [Lin et al., 2009]
- Research and Design of Electronic Commerce Systems Based on SCA [mu Ji and cen Jiang, 2010] - IEEE not accessible text

SCA Examples 1/5

Tutorial [OSOA, 2005]

SCA Building your first application - simplified BigBank

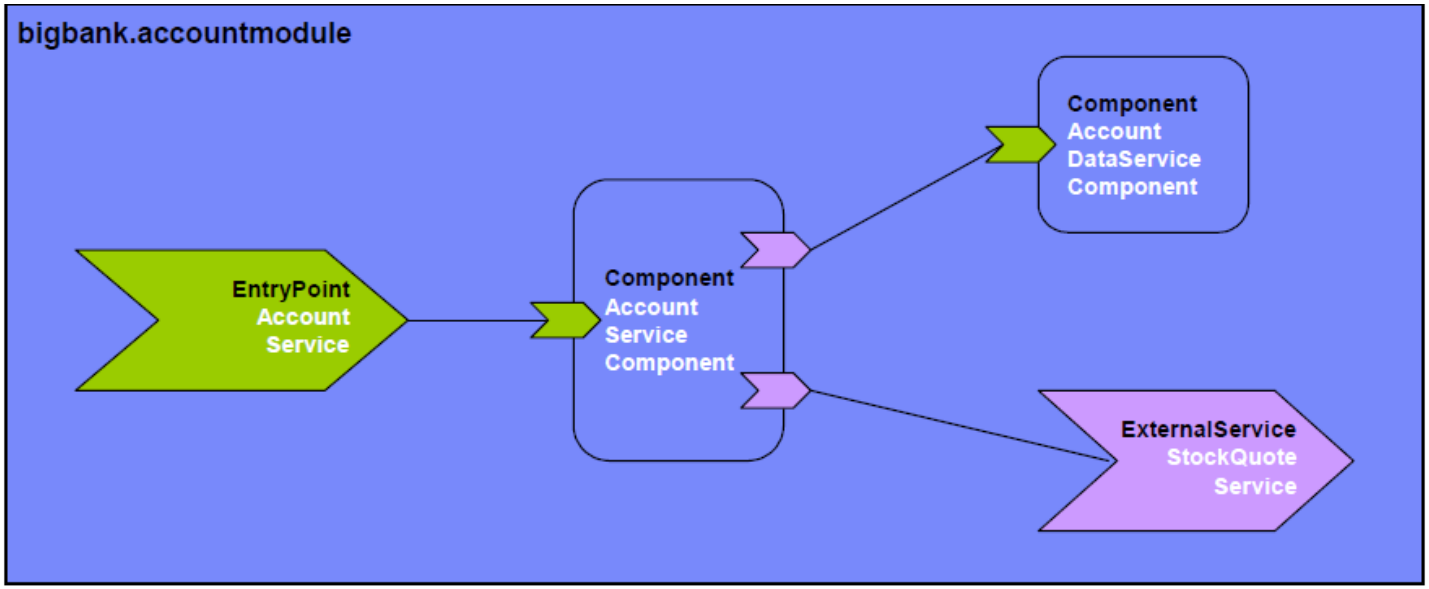
- Creating component implementations that provide remotable services in the JavaTM language. Remotable services can be published to remote clients over various protocol bindings, e.g. as web services.
- Creating component implementations that provide local services in Java. Local services implement internal application business logic such as tracking user state and are not exposed remotely.
- Creating component implementations that have configuration properties and service references to other services
- Creating components that use and configure the properties and references of component implementations
- Creating entry points to publish remotable services via a Web Service binding.
- Creating external services to consume remotable services via a Web Service binding
- Assembling implementation, components, entry points and external service into modules.
- Creating a module and all of its artifacts as part of a web application to show a frontend access to SCA services
- Configuring and deploying a module into a SCA system using subsystem configurations.

service for viewing customer account balance

(i.e. checking account, savings account, and stock account)

SCA Examples 1/5 contd

Tutorial [OSOA, 2005]

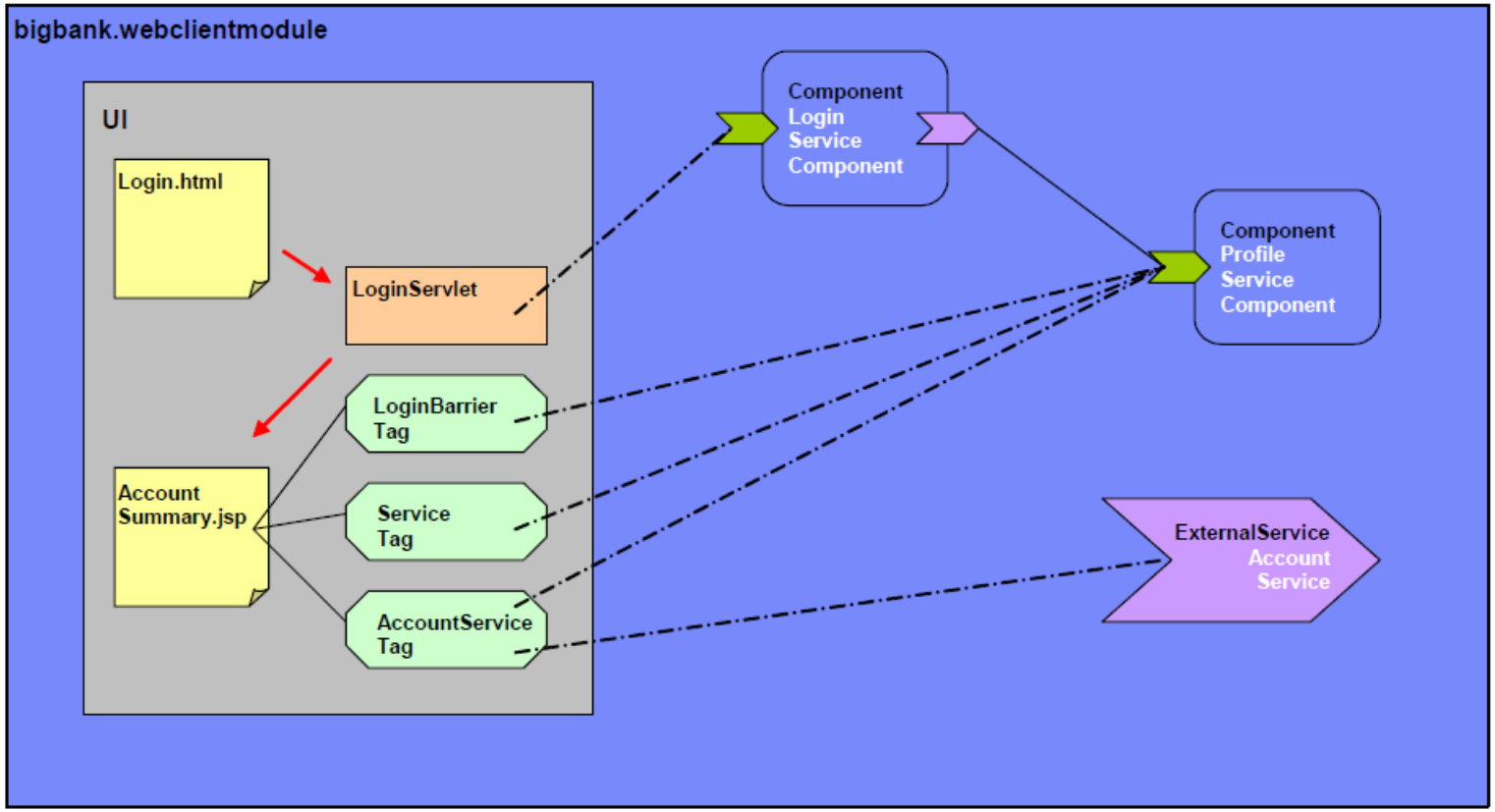


The module `bigbank.accountmodule` exposes the account service for accessing account information in a legacy system using web services protocols. It contains :

- The account service component, which provides the remotable account service and aggregates the report on the customers checking, savings, and stock account
- The account data service component which takes the role of the legacy system, and provides checking account, savings account, and stock account information to the account service
- The external stock quote service which provides current quotes on stocks to the account service
- The entry point account service that publishes the account service over a web service binding for access by the web client module and other remote web services clients.
- The assembly that configures and wires all the elements of the module.

SCA Examples 1/5 contd

Tutorial [OSOA, 2005]



The module `bigbank.webclientmodule` provides browser-based functionality for logging into the system and accessing account information.

- The login HTML file, the login servlet, and the account summary JSP for processing web requests and displaying account information (⇝ JSF or Struts).
- The login service and profile service components that provide local services for managing user state
- The external account service for accessing the remote account service of the `bigbank.accountmodule`.
- The assembly that configures and wires all the elements of the module.

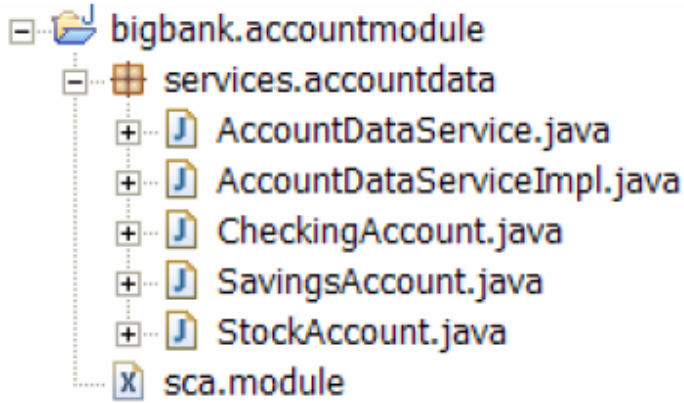
SCA Examples 1/5 contd

Tutorial [OSOA, 2005] Development

- Module `bigbank.accountmodule`

```
<?xml version="1.0" encoding="ASCII"?>
<module xmlns="http://www.osoa.org/xmlns/sca/0.9"
name="bigbank.accountmodule" >
</module>
```

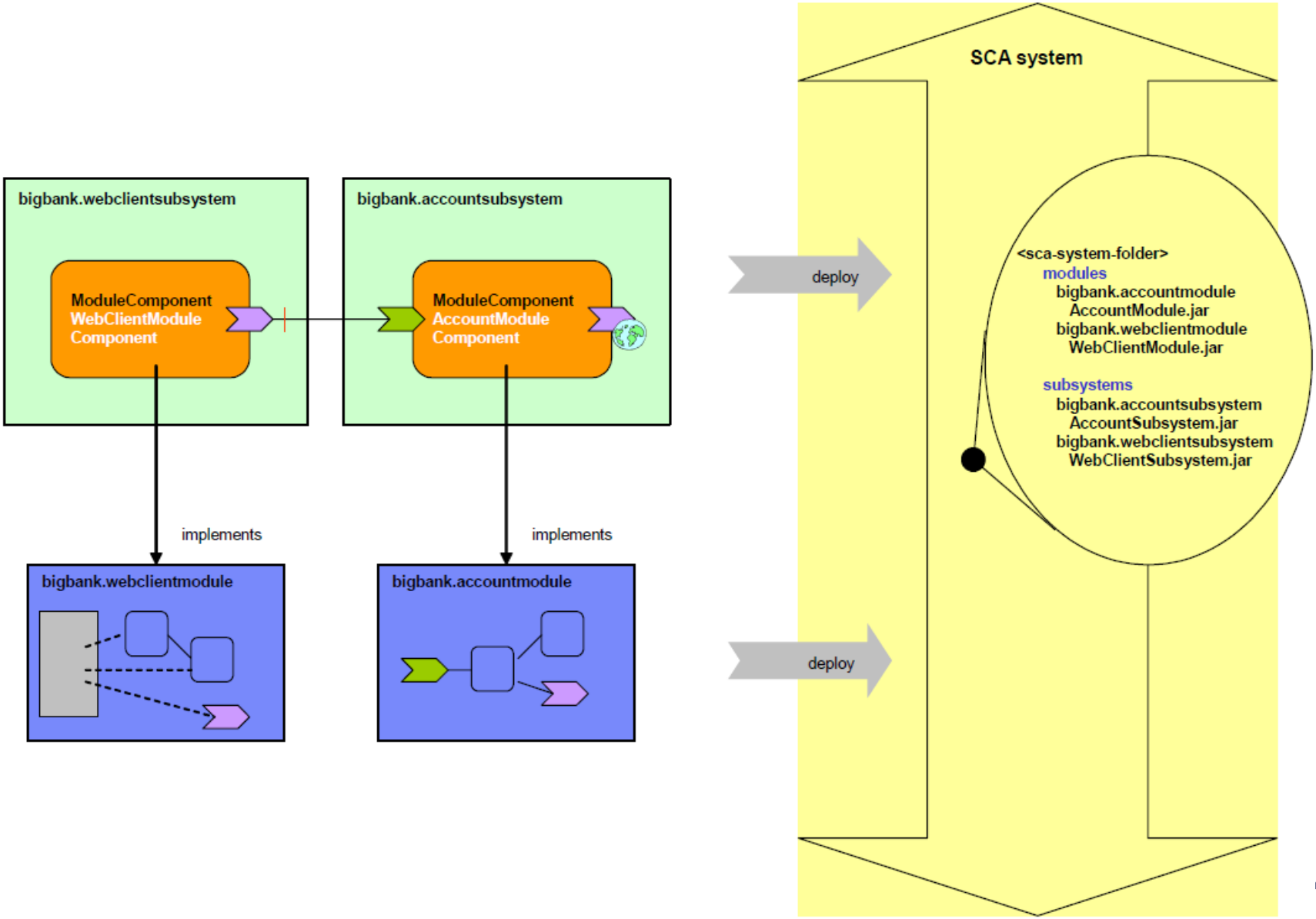
- Account Data Service Implementation



(détails des classes)

SCA Examples 1/5 contd

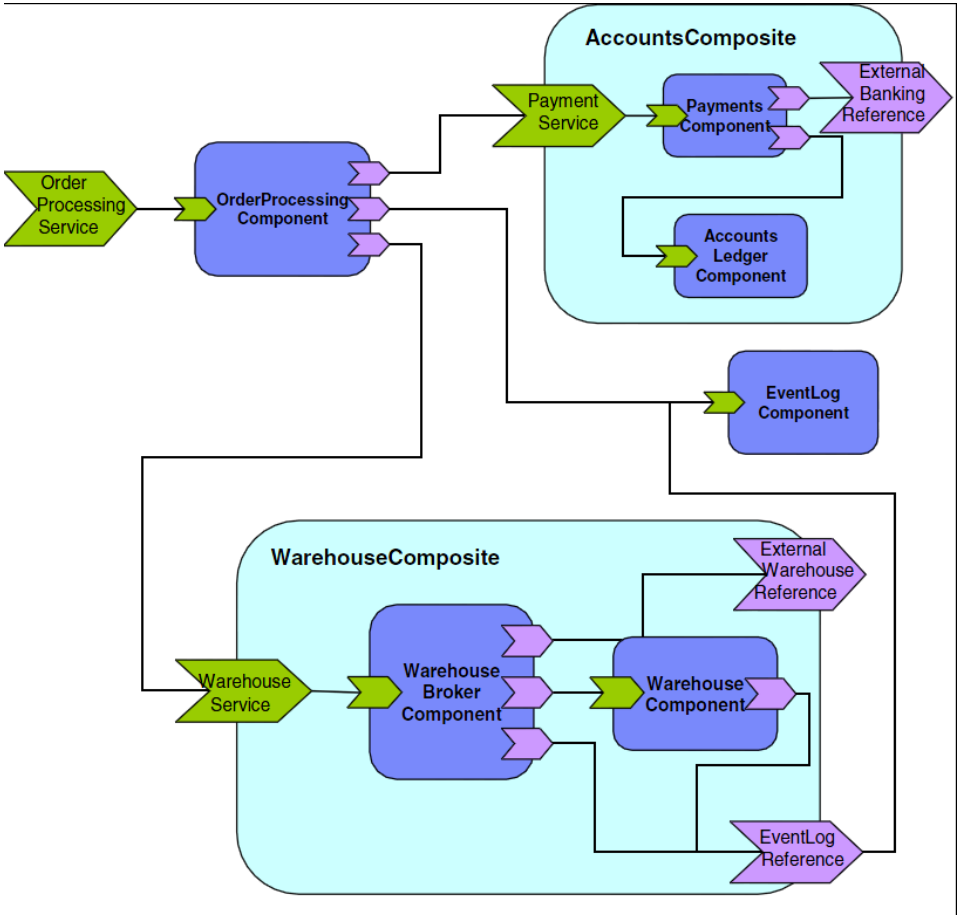
Tutorial [OSOA, 2005] - Deployment



SCA Examples 2/5

SCA explanation : [Karmarkar and Edwards, 2006]

Assembly of Business Systems Using Service Component Architecture



SCA Examples 3/5

[Krämer, 2008]

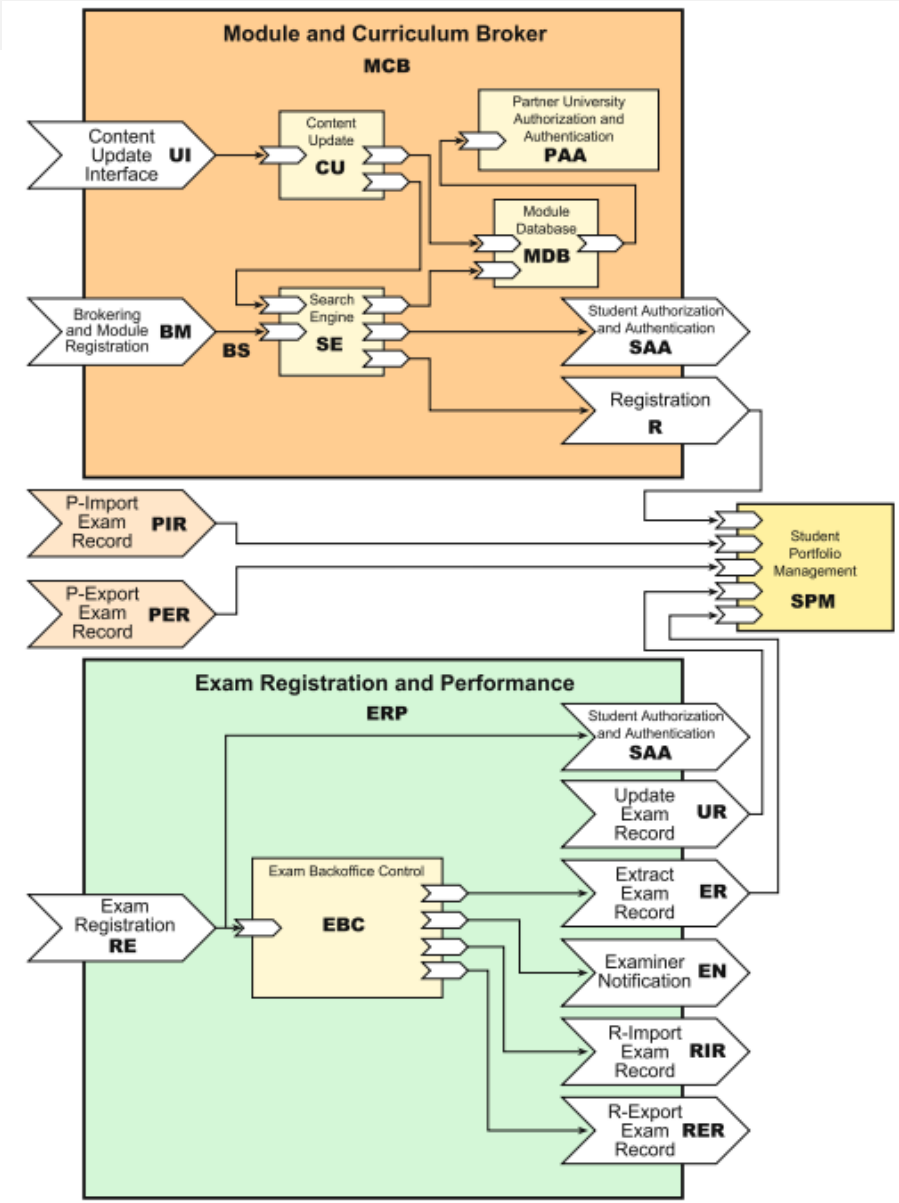
Component meets service : what does the mongrel look like ?

- SCA tries to reconcile Components and Services (*me : see also [Collet et al., 2007]*)
- BPEL activity orchestration
- example : *virtual mobility in e-universities*
- SCA evaluation
  - contracts introduction
    - behavioral contracts (OCL, CASL)
    - synchronization contract (not yet but [Ding et al., 2008])
    - QoS contract (not yet)
  - component compatibility
  - ...

source : [Krämer, 2008]

SCA Examples 3/5 contd

[Krämer, 2008] example



SCA Examples 4/5

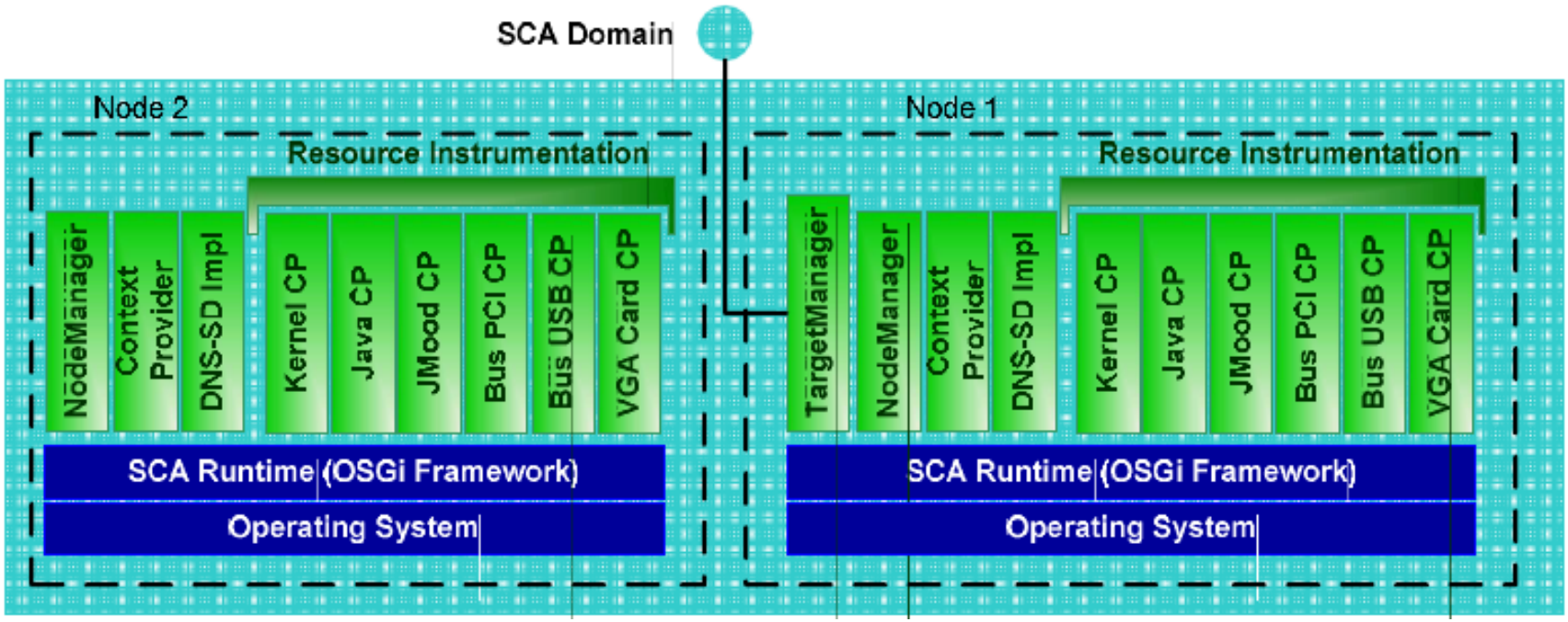
[Ruiz et al., 2008]

A Service Component Deployment Architecture for e-Banking

- deployment solution for service-oriented JEE distributed systems based upon available specifications (SCA, OSGi and OMG Deployment and Configuration) and OSS implementations
- JBI (Java Business Integration)
- example : *Spanish banking market*
- SCA case-study
  - deployment extension
    - service versioning
    - component distribution
    - context adaptation
  - OMG Deployment and Configuration of Component-Based Distributed Applications
  - ...

SCA Examples 4/5 contd

[Ruiz et al., 2008] example



SCA Examples 5/5

[Lin et al., 2009]

Service Component Architecture for Vending Machine System in Cloud Computing Infrastructure

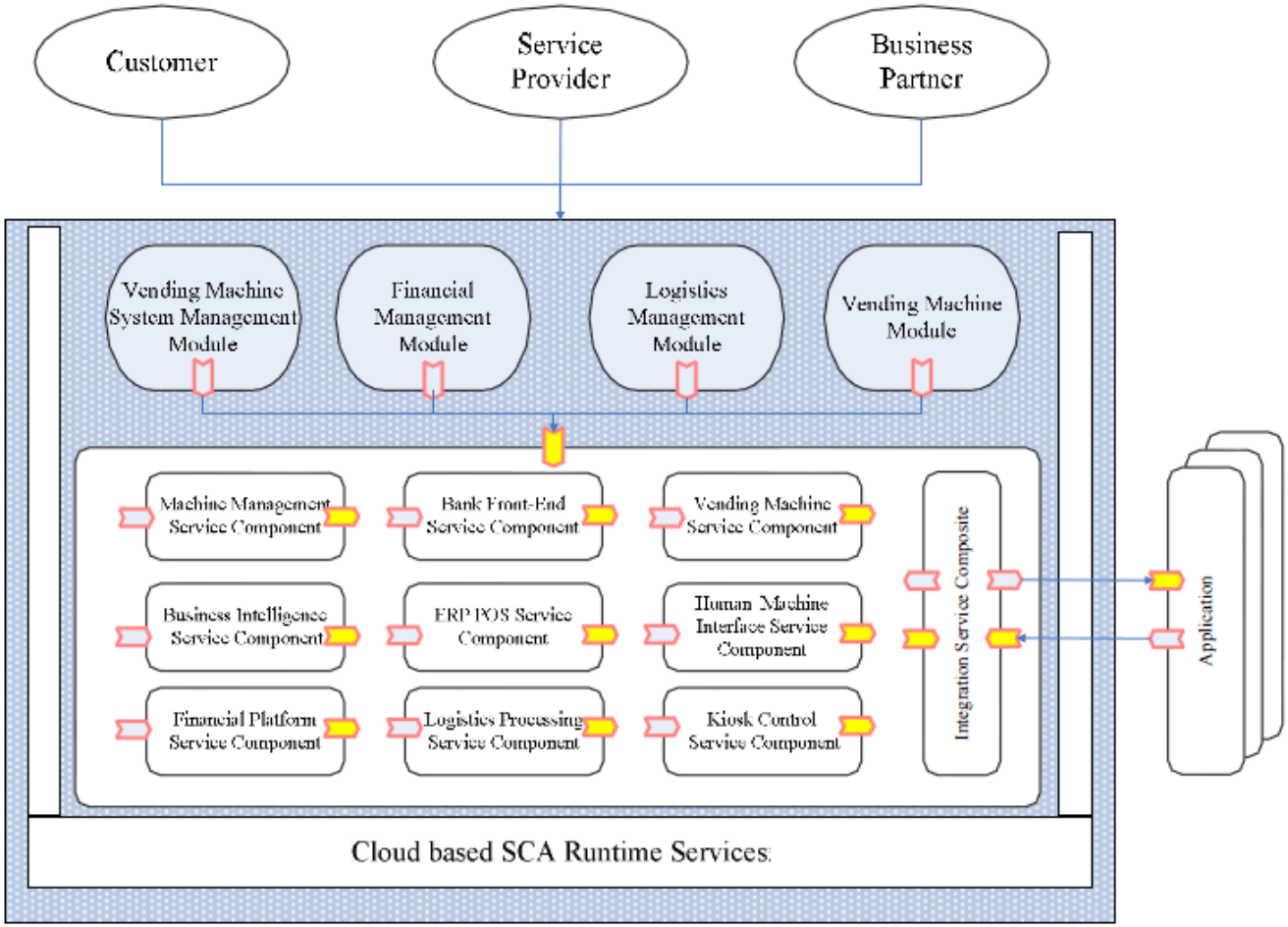
- Cloud implementation of SCA services
- Integrate software components
- architecture of Platform as a Service (PaaS)

- service intergration
- service overloading ($\rightsquigarrow$ multi)

- outside the scope (?)

SCA Examples 5/5 contd

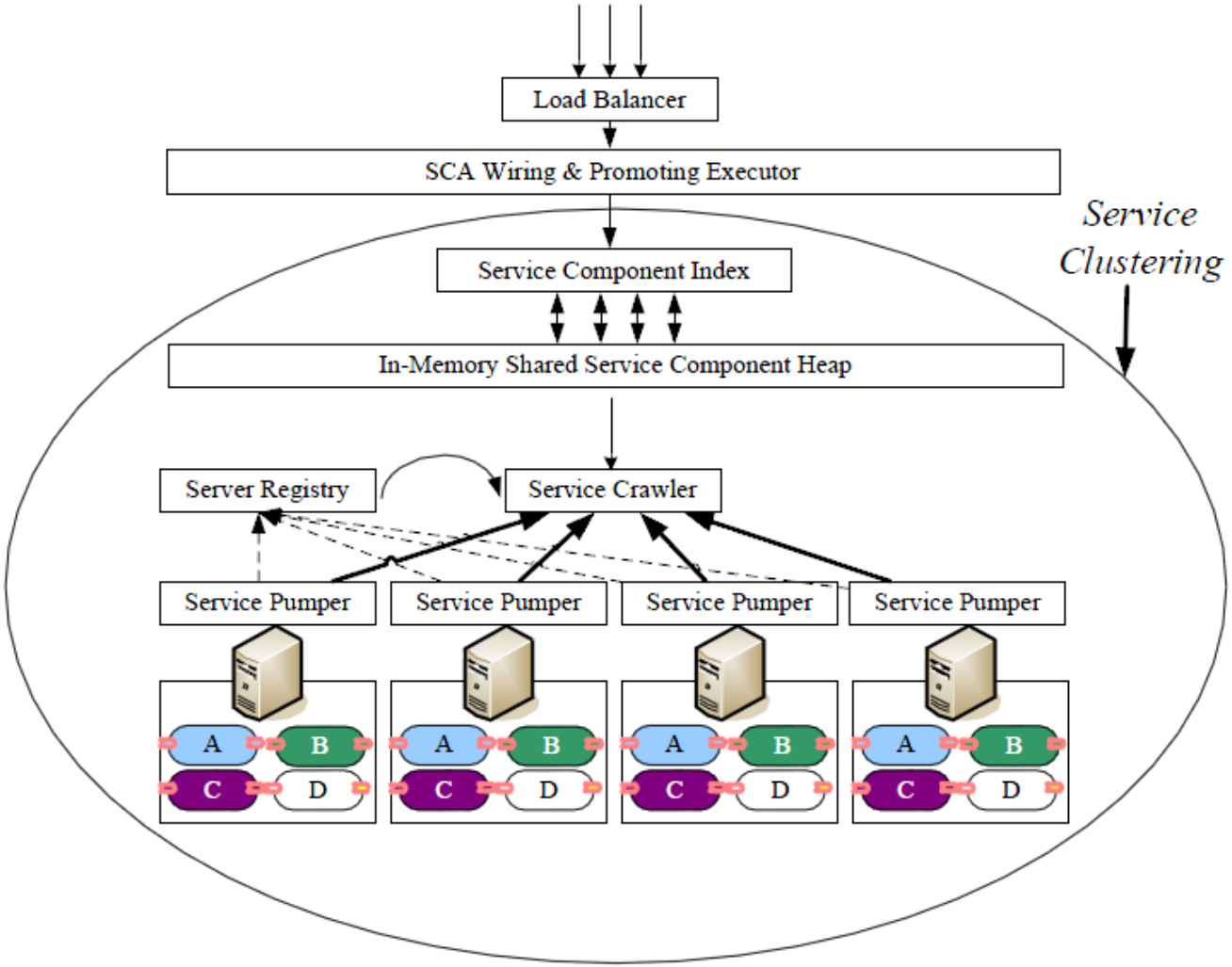
[Lin et al., 2009] example



- Vending Machine System Management Module,
- Financial Management Module,
- Logistics Management Module and
- Vending Machine Module

SCA Examples 5/5 contd

[Lin et al., 2009] example



- Cloud-Based Integration Model of Service Component Architecture
- Service operation
- Distributed Shared Memory

Outline

SCA Tools

Tools and Implementations

1. The SCA Tools project (Eclipse)
2. Tuscany
3. The FraSCAti Platform
4. OBEO
5. Visio 2010 sheet

SCA Tools

The SCA Tools project (Eclipse)

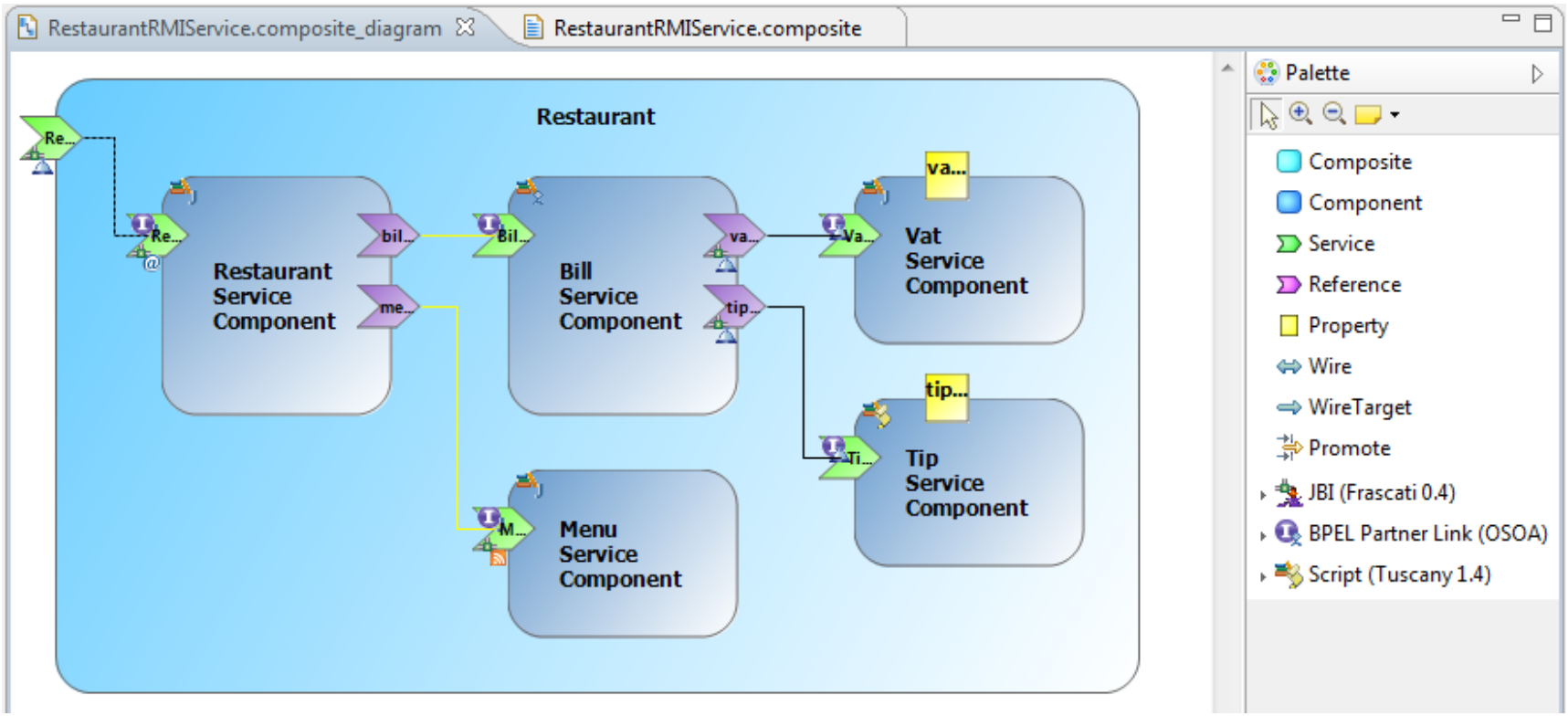
`http://www.eclipse.org/stp/sca/`

1. The SCA Meta Model, an extensible EMF meta model based on
   $\Longrightarrow$ the SCA specifications version 1.0 + an extension for Tuscany + an extension for FraSCAti
2. The SCA Composite Designer, a graphical (GMF) development environment for the construction of composite applications.
3. The SCA XML Editor for composite and componentType files.
4. The SCA Form Editor, an editor based on Eclipse forms for SCA assembly files.
5. The SCA Builder to validate SCA artifacts.
6. The SCA Launcher to run and debug SCA Java projects on SCA platforms from Eclipse.
7. The SCA Composite to Java Generator to generate Java code skeletons.
8. The SCA Composite Introspection tool to construct SCA assembly files in a bottom-up approach from existing POJO and component type files.
9. The SCA Semantics tool to annotate components, services, and references with SAWSDL.
10. The SCA Samples, a set of ready to use SCA projects to discover SCA and the SCA tools.

$\rightsquigarrow$ to test

SCA Tools

The SCA Tools project (example)



SCA Tools

Tuscany SCA

source : [Laws et al., 2009]

1. Apache project
2. developing SOA solutions by providing a comprehensive infrastructure for SOA development and management that is based on SCA standard
3. Tuscany is integrated with various technologies and offers :
   - a wide range of bindings (pluggable protocols)
   - various component types including and not limited to Java, C++, BPEL, Spring and scripting
   - an end to end service and data solution which includes support for Jaxb and SDO
   - a lightweight runtime that works standalone or with other application servers
   - a modular architecture that makes it easy to integrate with different technologies and to extend
   - Integration with web20 technologies
4. Apache Tuscany SCA is implemented in Java and C++ (referred to as Native)

source : `http://tuscany.apache.org/` ⇝ to test

SCA Tools

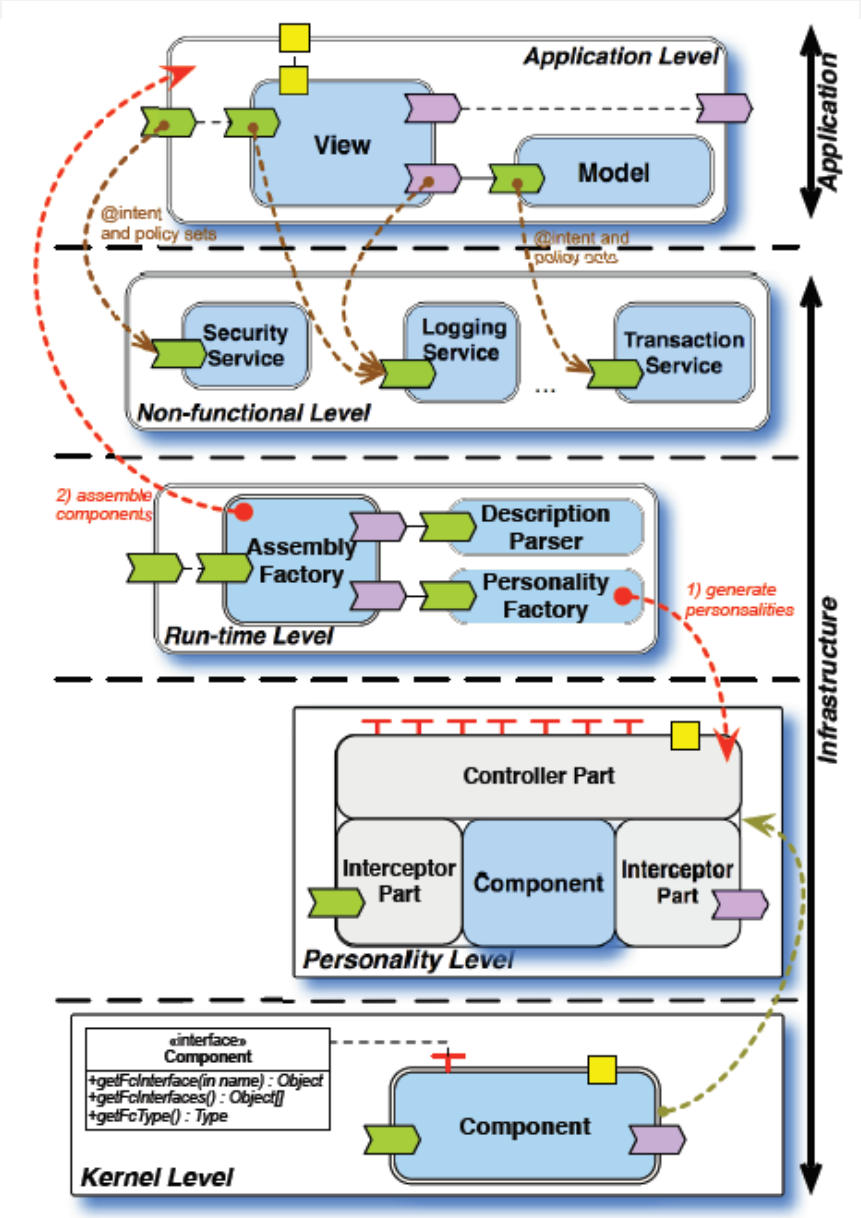
The FraSCAti Platform

source : [Seinturier et al., 2009, Mélisson et al., 2010]
Reconfigurable SCA Applications with the FraSCAti

- FraSCAti is a reflective platform for deploying, hosting, and managing SCA applications, and its different subsystems are implemented as SCA components.
- Provides run-time support, deployment capabilities, and run-time management for SCA.
- Address the issues of configurability and manageability in a systematic fashion, at the business (application components), and platform (non-functional services, communication protocols, etc.) levels.
- Compared to state-of-the-art platforms, FraSCAti brings a dynamic reflective support to SCA and enables both introspecting and reconfiguring service-oriented architectures at run-time.

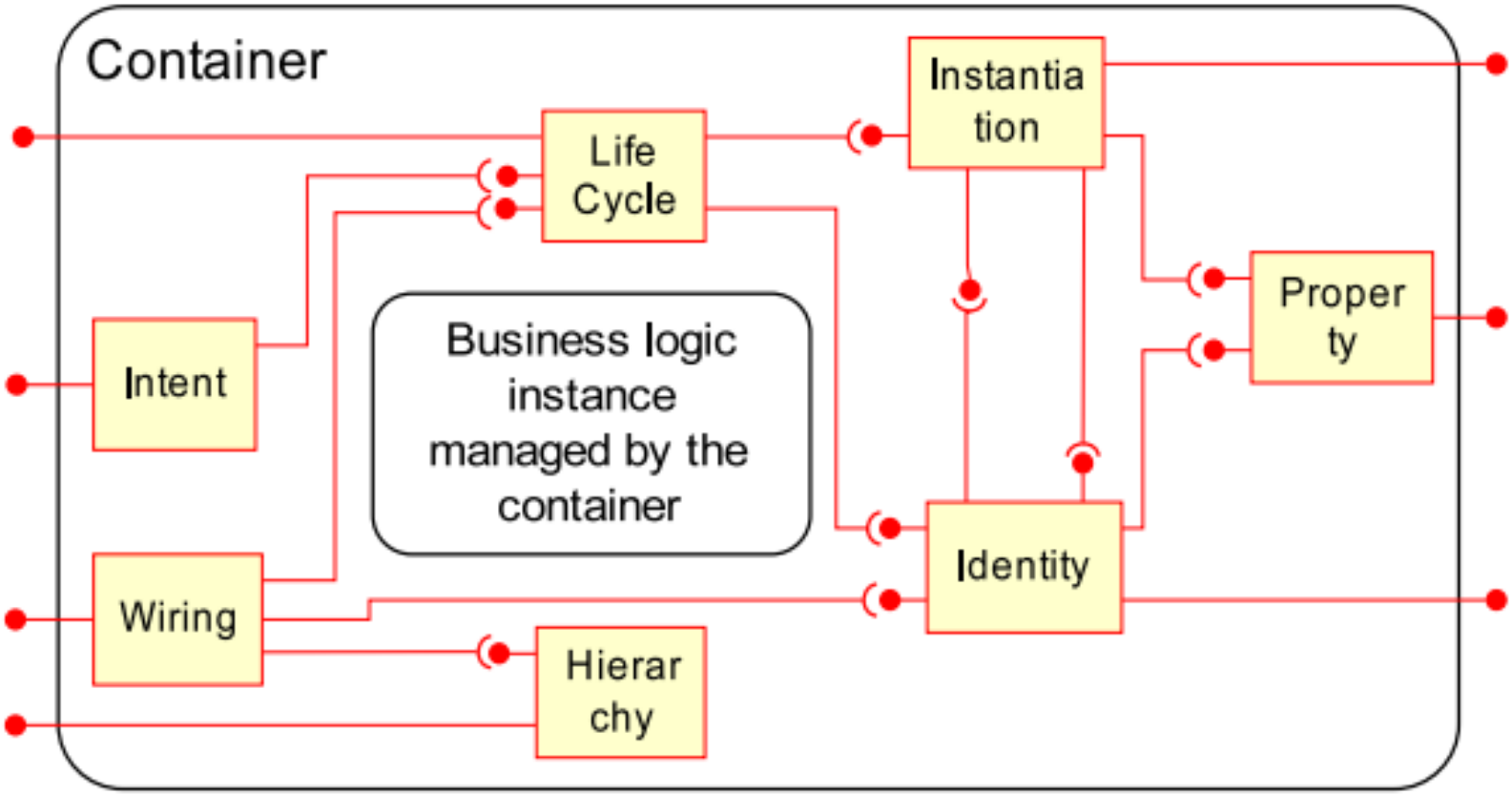
SCA Tools

The FraSCAti Platform contd



SCA Tools

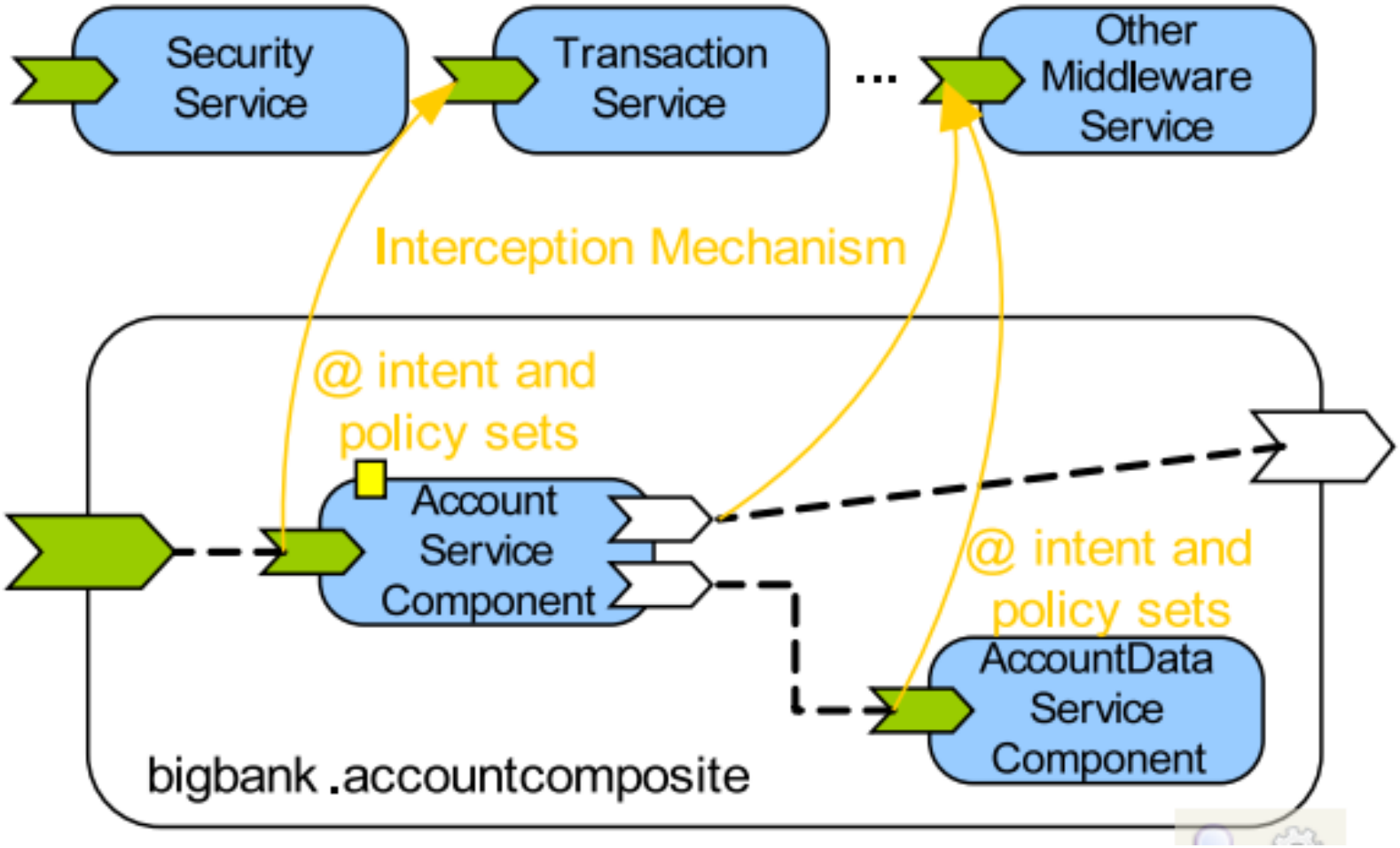
The FraSCAti Platform contd



FraSCAti container architecture

SCA Tools

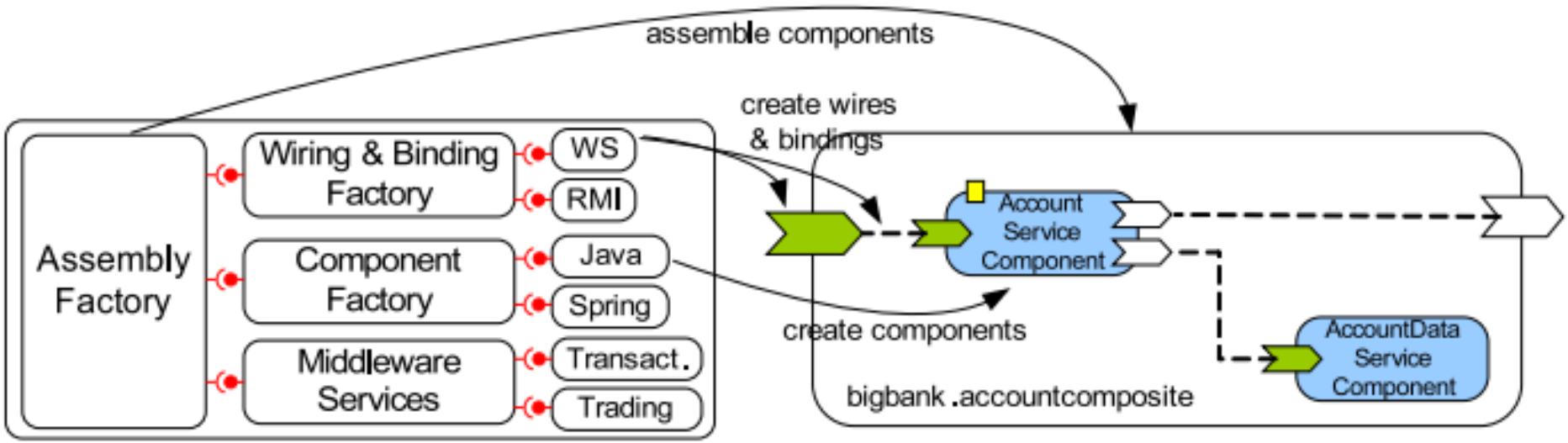
The FraSCAti Platform contd



FraSCAti interception mechanism.

SCA Tools

The FraSCAti Platform contd



FraSCAti platform architecture.

SCA Tools

OBEO

Obeo SCA

1. expertise in the Eclipse SCA Tools project
2. experience as participant in the Steering Committee of the SOA Industrial Working Group of the Eclipse Foundation
3. Tools based on SCA for cartography and migration of legacy code.
   - Extractors based on Obeo Agility that detect components, services and bindings in legacy code,
   - Graphical viewpoints based on Obeo Designer enabling the analysis of legacy code,
   - Generators allowing you to generate automatically SCA applications from legacy code
   - Customisation tools.

source : `http://obeo.fr/pages/sca/fr`

Outline

Bilan

Petit tour d'horizon de SCA :

- standard industriel
  - norme
  - mise en pratique progressive, quelques exemples
- formalisation
  - plutôt maigre
  - contrats comportementaux
- outils :
  - STP principalement
  - support Tuscany, FraSCAti
- contacts locaux, nationaux

SCA vs Kmelia

- Cohérence - Proximité des modèles
  - services hiérarchiques, contrats fonctionnels et dynamiques
  - assemblage et liens (wiring)
  - composites et promotions
- Complémentarité
  - SCA manque de support formel $\Longrightarrow$ formaliser avec Kmelia, vérifier avec Costo
  - Kmelia manque de visibilité $\Longrightarrow$ habillage SCA (?)
  - Kmelia manque d'exemples de services $\Longrightarrow$ reprendre ceux de SCA
  - Kmelia manque d'ancrage implantation $\Longrightarrow$ implanter avec SCA (?)
- Extensions / Spécifique
  - lien avec le prototypage Kmelia/Costo
  - lien avec le Test Kmelia/Costo
  - lien avec le contrôle de la promotion
  - lien avec les contrats

Références I


Abreu, J. a., Mazzanti, F., Fiadeiro, J. L., and Gnesi, S. (2009).
A model-checking approach for service component architectures.
In *Proceedings of the Joint 11th IFIP WG 6.1 International Conference FMOODS '09 and 29th IFIP WG 6.1 International Conference FORTE '09 on Formal Techniques for Distributed Systems*, FMOODS '09/FORTE '09, pages 219–224, Berlin, Heidelberg. Springer-Verlag.


Abreu, J. P. A. d. (2009).
Modelling Business Conversations in Service Component Architectures.
PhD thesis, University of Leicester.


Collet, P., Coupaye, T., Chang, H., Seinturier, L., and Dufrêne, G. (2007).
Components and Services : A Marriage of Reason.
Research Report I3S-RR2007-17FR, I3S Lab, Sophia Antipolis, France.


Ding, Z., Chen, Z., and Liu, J. (2008).
A rigorous model of service component architecture.
Electr. Notes Theor. Comput. Sci., 207 :33–48.


Ding, Z., Jiang, M., and Liu, J. (2009).
Model checking service component composition by spin.
In Miao, H. and Hu, G., editors, *ACIS-ICIS*, pages 1029–1034. IEEE Computer Society.

Références II


Du, D., Liu, J., and Cao, H. (2008).
A rigorous model of contract-based service component architecture.
In *Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 02*, pages 409–412, Washington, DC, USA. IEEE Computer Society.


Fiadeiro, J. L., Lopes, A., and Bocchi, L. (2006).
A formal approach to service component architecture.
In Bravetti, M., Núñez, M., and Zavattaro, G., editors, *WS-FM*, volume 4184 of *Lecture Notes in Computer Science*, pages 193–213. Springer.


Hnetynka, P., Murphy, L., and Murphy, J. (2010).
Comparing the service component architecture and fractal component model.
Computer.


Karmarkar, A. and Edwards, M. (2006).
Assembly of business systems using service component architecture.
In Dan, A. and Lamersdorf, W., editors, *ICSOC*, volume 4294 of *Lecture Notes in Computer Science*, pages 529–539. Springer.


Krämer, B. J. (2008).
Component meets service : what does the mongrel look like ?
ISSE, 4(4) :385–394.

Références III


Laws, S., Combellack, M., Feng, R., Mahbod, H., and Nash, S. (2009).
Tuscany SCA in Action).
Manning Publications.


Lin, F.-C., Lee, Y.-S., Hsu, C.-H., Chen, K.-Y., and Weng, T.-C. (2009).
Service component architecture for vending machine system in cloud computing infrastructure.
E-Business Engineering, IEEE International Conference on, 0 :591–595.


Margolis, B. (2007).
SOA for the Business Developer : Concepts, BPEL, and SCA (Business Developers series).
Mc Press.


Marino, J. and Rowley, M. (2009).
Understanding SCA (Service Component Architecture).
Addison-Wesley Professional, 1st edition.


Mélisson, R., Merle, P., Romero, D., Rouvoy, R., and Seinturier, L. (2010).
Reconfigurable run-time support for distributed service component architectures.
In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, ASE '10, pages 171–172, New York, NY, USA. ACM.


mu Ji, Y. and cen Jiang, L. (2010).
Research and design of electronic commerce systems based on sca.
International Conference on E-Business and E-Government, 0 :2312–2315.

Références IV

OSOA (2005).
Sca building your first application - simplified bigbank.
Specification Version 0.9, Open SOA Collaboration.
online http ://www.osoa.org.

OSOA (2007).
Service component architecture (sca) : Sca assembly model v1.00 specifications.
Specification Version 1.0, Open SOA Collaboration.
online http ://www.osoa.org.

Ruiz, J. L., Dueñas, J. C., and Cuadrado, F. (2008).
A service component deployment architecture for e-banking.
In *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops*, pages 1369–1374, Washington, DC, USA. IEEE Computer Society.

Seinturier, L., Merle, P., Fournier, D., Dolet, N., Schiavoni, V., and Stefani, J.-B. (2009).
Reconfigurable SCA Applications with the FraSCAti Platform.
In *6th IEEE International Conference on Service Computing (SCC'09)*, pages 268–275, Bangalore Inde. IEEE.
CORE A. Acceptance rate : 18