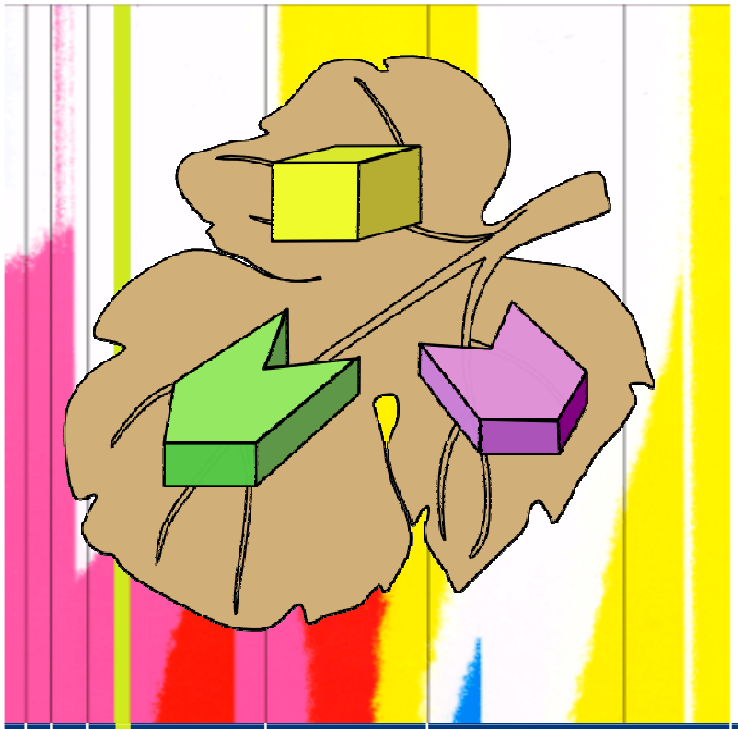


FraSCAti

«*Open SCA Platform*»

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
LILLE - NORD EUROPE

Université
Lille1
Sciences et Technologies

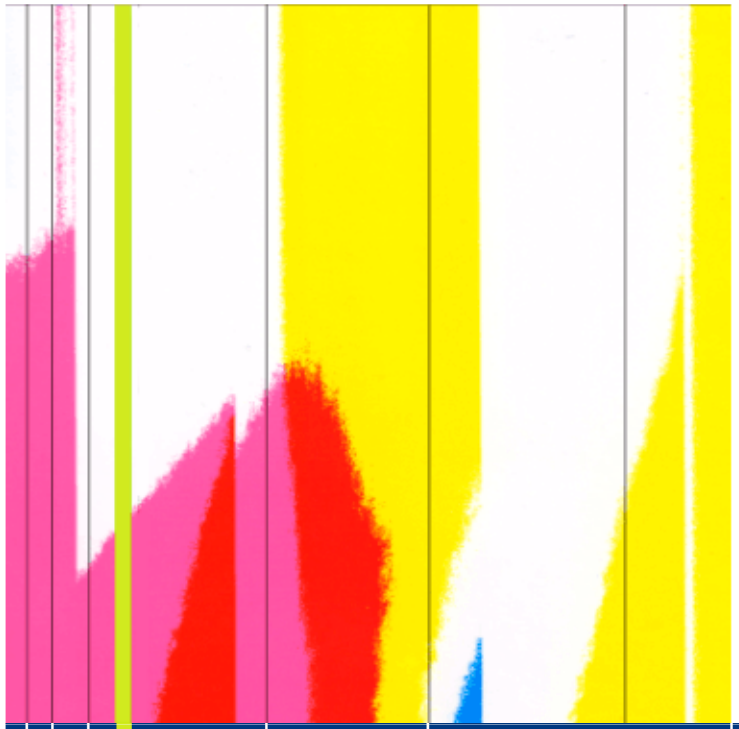
lifL

cnrs
dépasser les frontières

INRIA
Team
A D A M

Outline

- From SOA to SCA
- FraSCAti
- SCOrWare
- Conclusion



From SOA to SCA

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
LILLE - NORD EUROPE

Université
Lille1
Sciences et Technologies

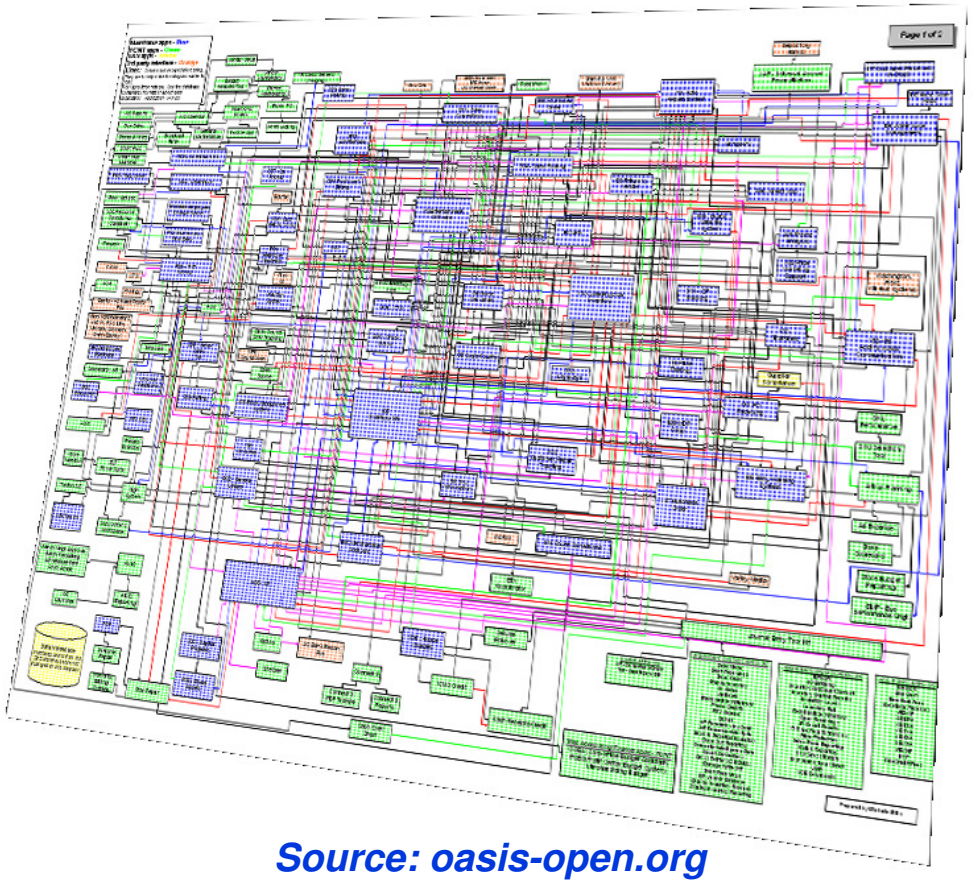
lifL

cnrs
dépasser les frontières

INRIA
Team
A D A M

From SOA challenges…

- IT architectures
- Complexity
  - Managing 10^n lines of code
- Monolithic
  - Breaking application «silos»
- Seldom evolvable
  - Freeing systems from immutable dependencies

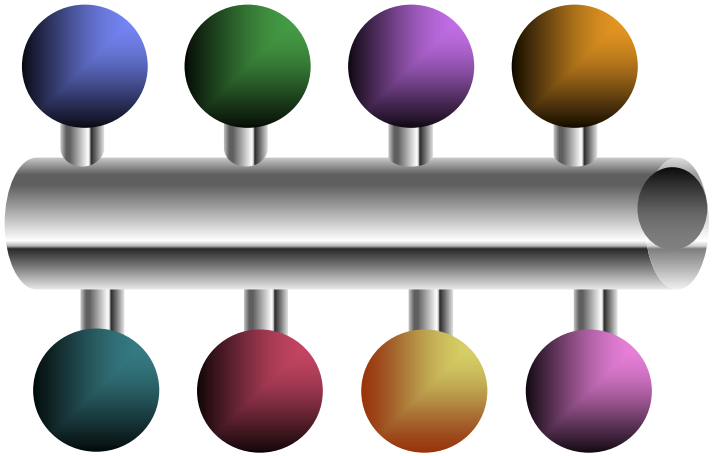


Source: oasis-open.org

…to existing SOA, but…

SOA leverages complexity and promotes flexibility

- Loose coupling
- Service composition and orchestration
- Well defined and contractualized interfaces
- Standard tools and technologies



Source: oasis-open.org

…Still a partial solution

Today's SOA need to be…

- Deployable in different environments
- Ensure security and reliability
- Adaptable to changing business needs

…and thus, SOA lack…

- Structured architectures
  - What is behind the scene?
- Reuse capabilities
  - Reuse the wheel when possible…
- Flexibility support
  - …Or tune it if not!

You want SCA!

for your business

SCA in a Nutshell

SCA (Service Component Architecture)

- Aka a «*Component Model for SOA*»
- Since 11/2005

Hosted by the Open SOA consortium

- http://www.osoa.org

Community connected to OASIS

- http://www.oasis-opencsa.org

Existing platform providers

- Open Source (4): Apache Tuscany, Newton, Fabric3, **FraSCAti**
- Vendors (7): IBM WebSphere FP for SOA, TIBCO ActiveMatrix, Covansys SCA Framework, Paremus, Rogue Wave HydraSCA, Oracle Fusion Middleware

SCA in a Nutshell

15 focused specifications (09/2008) + SDO to access data sources

Assembly model specification (structured architectures ☺)

- How to structure composite systems?

Component implementation specifications (flexibility support ☺)

- How to develop IT services in specific programming languages?
  - Java, C++, PHP, Spring, BPEL, EJB, SLSB, COBOL, C…

Binding specifications (flexibility support ☺)

- How to access remote services?
  - Web services, JMS, JCA, RMI-IIOP…

Policy framework specification (flexibility support ☺)

- How to integrate infrastructure services?
  - Logging, security, transaction, reliable messaging…

Integration specifications (structured architectures ☺)

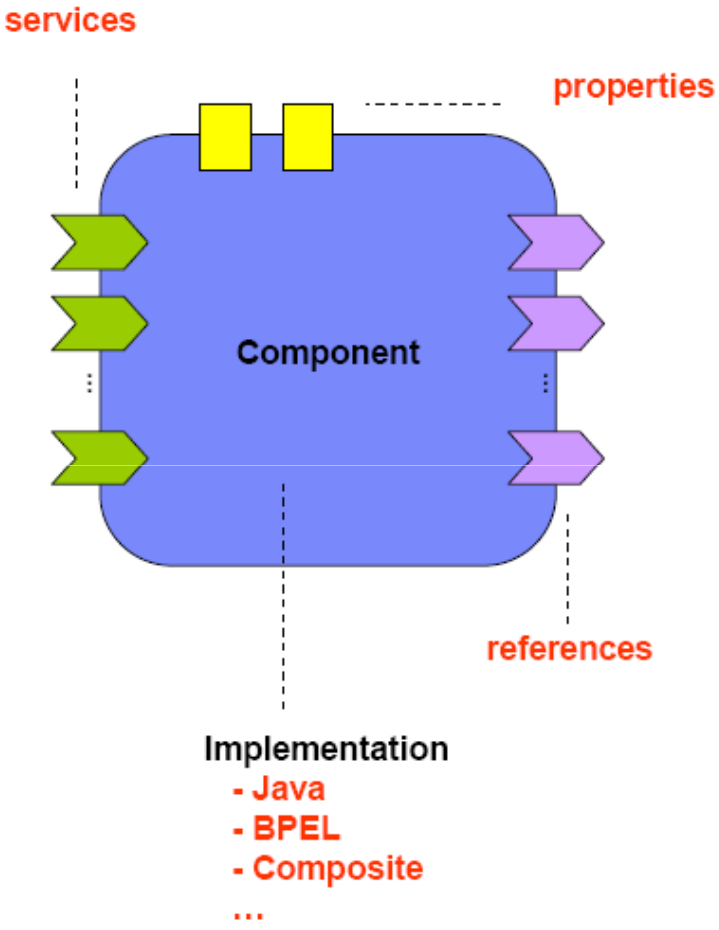
- SCA Java EE Integration
- SCA OSGi/Spring (draft)

SCA in a Nutshell (cont'd)

Component implements the business logic

Concepts

- Service(s)
  - Interface type: Java , WSDL
- Reference(s)
- Property(s)
- Implementation
- Non functional property(s)
  - Intent & policy

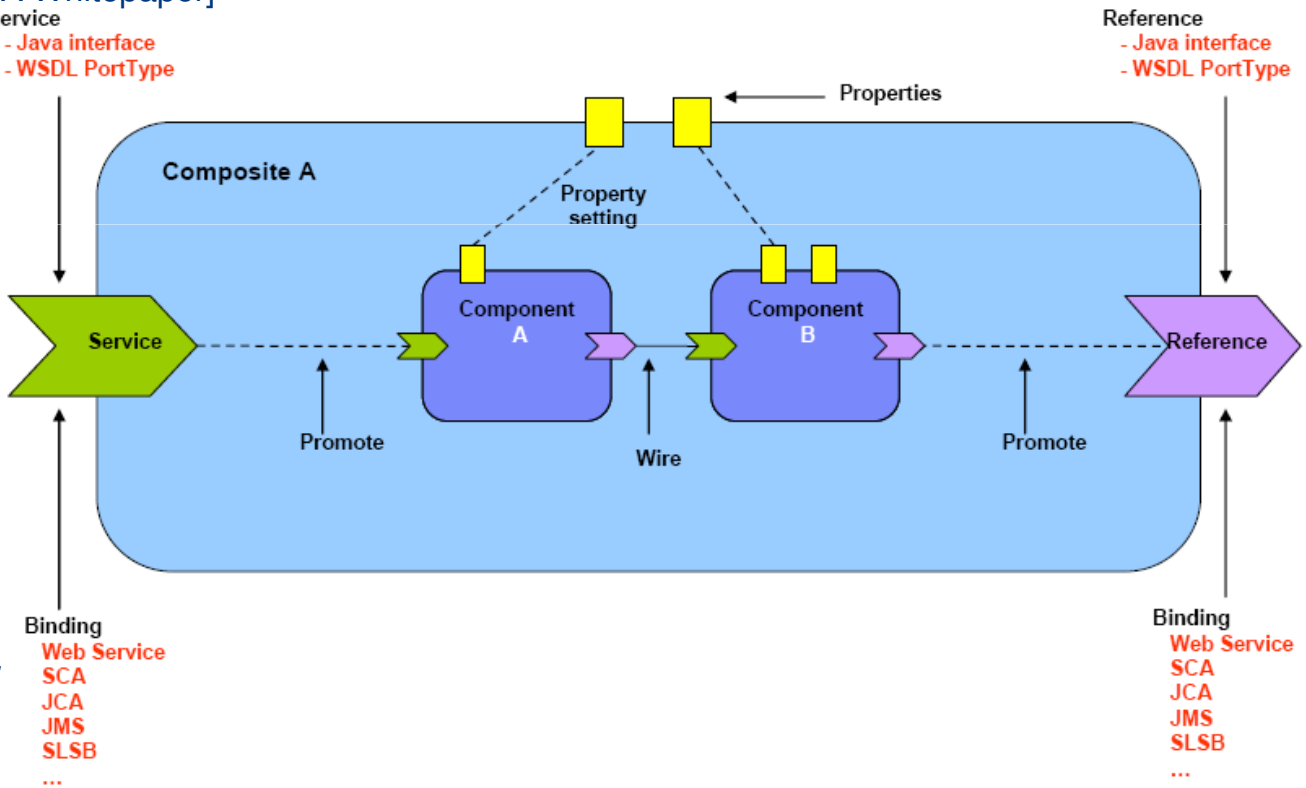


SCA in a Nutshell (cont'd)

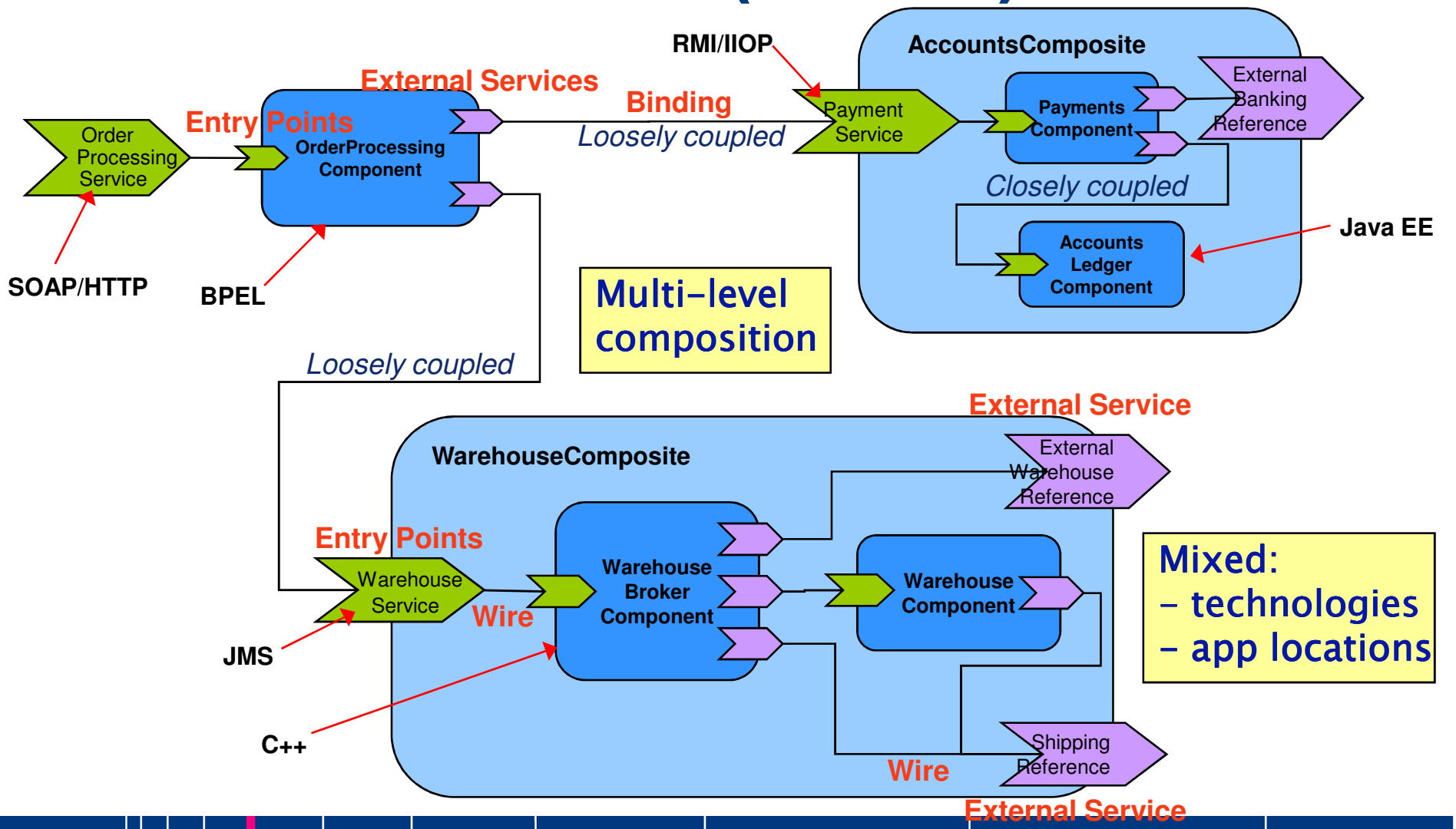
Assembly: "Process of composing business applications by configuring and connecting components that provide service implementations" [SCA Whitepaper]

2 Levels:

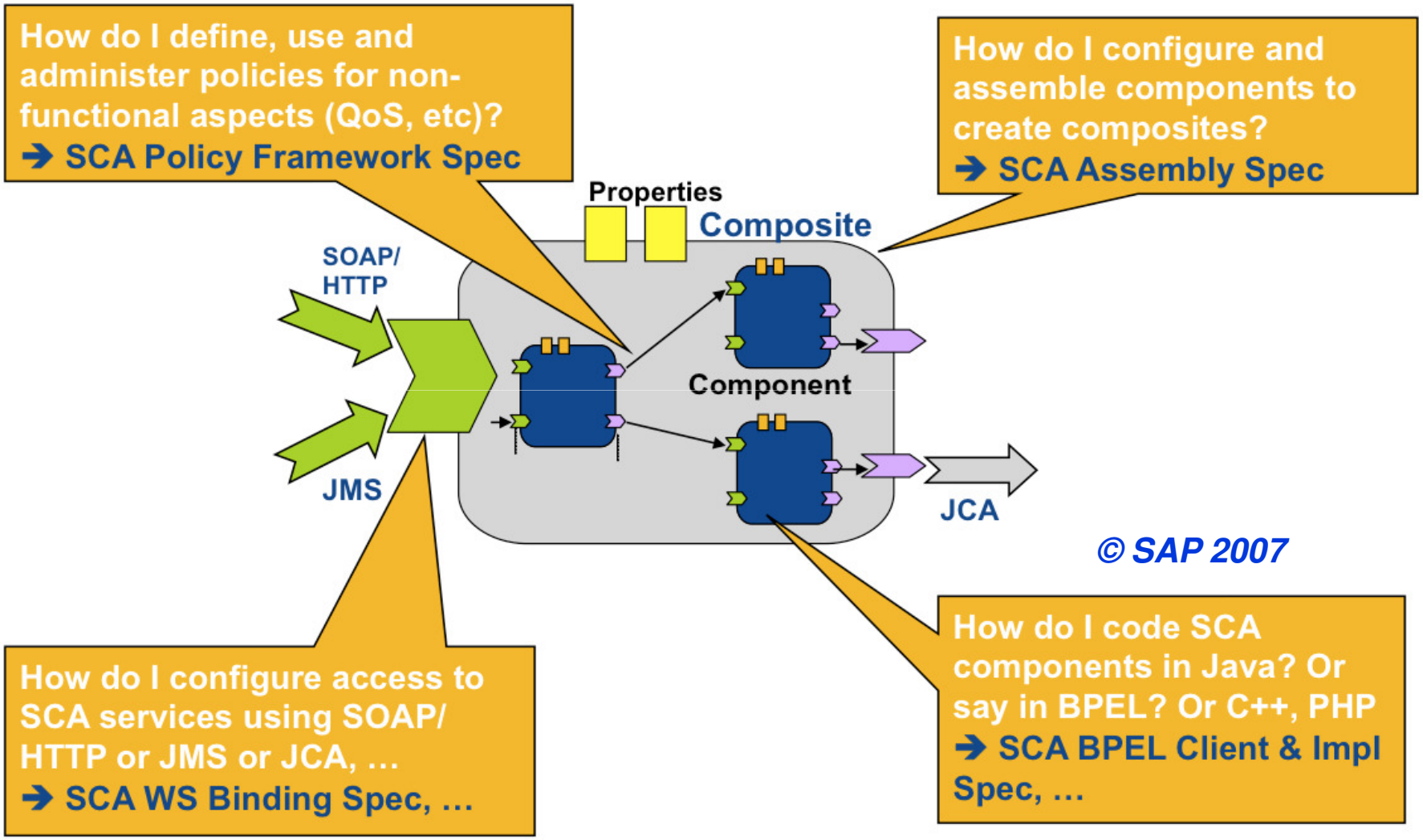
- Module assembly
  - *Closely coupled*
  - See figure
- System assembly
  - *Loosely Coupled*



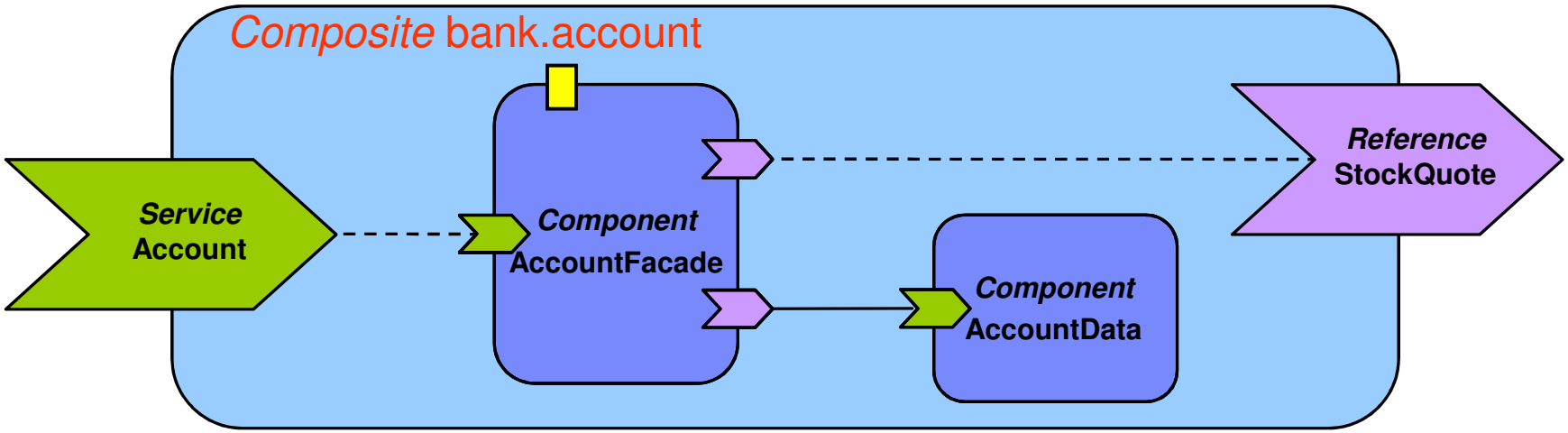
SCA in a Nutshell (cont'd)



SCA in a Nutshell (cont'd)



Simple SCA Assembly



[Mike Edwards]
IBM Hursley Lab, England

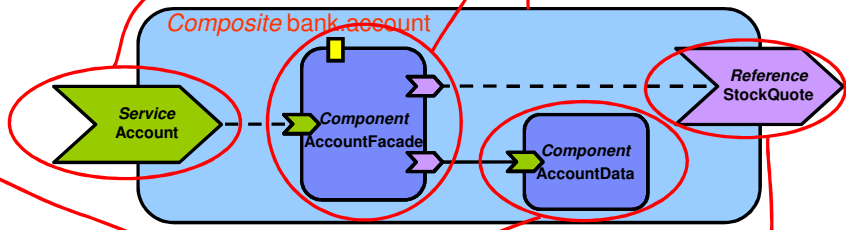
```
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
           name="bank.account" >

    <service name="Account" promote="AccountFacade">
        <interface.java interface="services.account.Account"/>
        <binding.ws port="http://www.example.org/Account#
          wsdl.endpoint(Account/AccountSOAP)"/>
    </service>

    <component name="AccountFacade">
        <implementation.java class="services.account.AccountFacadeImpl"/>
        <reference name="StockQuote"/>
        <reference name="AccountData"
                   target="AccountData/Data"/>
        <property name="currency">EURO</property>
    </component>

    <component name="AccountData">
        <implementation.bpel process="QName"/>
        <service name="Data">
            <interface.java interface="services.account.Data"/>
        </service>
    </component>

    <reference name="StockQuote" promote="AccountFacade/StockQuote">
        <interface.java interface="services.stockquote.StockQuote"/>
        <binding.ws port="http://example.org/StockQuote#
          wsdl.endpoint(StockQuote/StockQuoteSOAP)"/>
    </reference>
</composite>
```



Java Implementation Example: Service

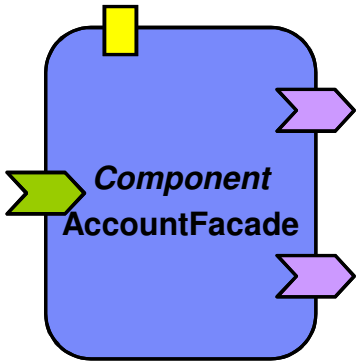

```
package services.account;

@Remotable
public interface Account {

   AccountReport getAccountReport(String customerID);
}
```

Interface is available remotely, *e.g.* as a Web Service

Java Implementation Example: Component




```
package services.account;

import org.osoa.sca.annotations.*;

@Service(interfaces = Account.class)
public class AccountFacadeImpl implements Account {

    private String currency = "USD";
    private Data accountDataService;
    private StockQuote stockQuoteService;

    public AccountServiceImpl(
    @Property("currency") String currency,
    @Reference("accountData") Data dataService
    @Reference("stockQuote") StockQuote stockService) {
       this.currency         = currency;
       this.accountDataService = dataService;
       this.stockQuoteService  = stockService;
}   }
```

Annotation for the service offered by this class

Annotated Constructor to inject property and references

SCA Benefits

A Component Model…

- Hierarchic compositions
- Reconfigurable properties
- Provided/required ports
- Synchronous/asynchronous invocations
- Communication bindings
- Intent and policy supports

…for SOA

- Future OASIS standard
- Industry acceptance (Apache Tuscany, IBM…)

4 degrees of flexibility / adaptability

- Implementation language (Java, C++, BPEL, etc.)
- Interface description language (Java, WSDL, etc.)
- Communication protocol (SOAP, IIOP, etc.)
- Non-functional properties (security, transactions, etc.)

SCA Benefits

Use Case	Benefit of using SCA Standard
SOA does not always mean WS	• Neutral to communication technologies • Supports WS, JMS, JCA bindings • Wires internal to SCA domain use proprietary technology
Bridging QoS Models of heterogeneous platforms	• Modeling and configuring QoS aspects is handled by the platform neutral SCA Assembly layer •SCA defines QoS aspects in abstract terms ('intents') and allows their mapping to individual platform environments
Managing changes to service provider/location	• SCA component implementations are programmed to interfaces • Service endpoint information is not hardwired into client code • Wiring of components is a first class concept with elaborate support for common scenarios (internal, external, redeployment)
Support for testing, management	• By providing a holistic view of the solution, it becomes possible for management tools to capture service dependency information •Service testing tools can be more effective
Tolerance to new application runtimes and communication technologies	• Framework for bindings to different technologies makes it possible for developers to apply a consistent programming model

SCA Limitations

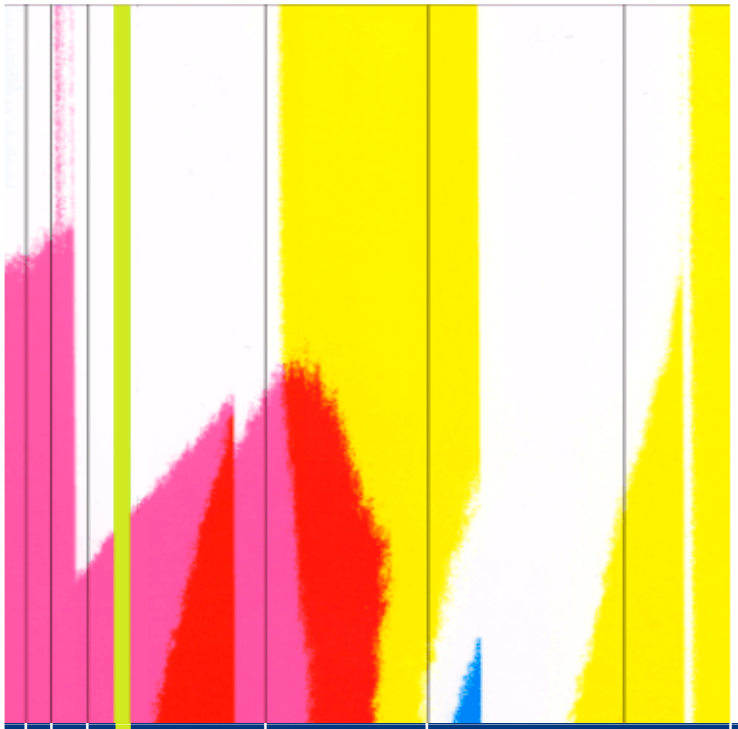
Static configuration & deployment

- XML file for describing composite components
- Lack of deployment API

No runtime adaptation & reconfiguration

- Lack of introspection API
- Lack of reconfiguration API

SCA is not a **reflective** component model



FraSCAti

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
LILLE - NORD EUROPE

Université
Lille1
Sciences et Technologies

lifL

cnrs
dépasser les frontières

INRIA
Team
A D A M

FraSCAti = SCA++

Dynamic deployment & configuration

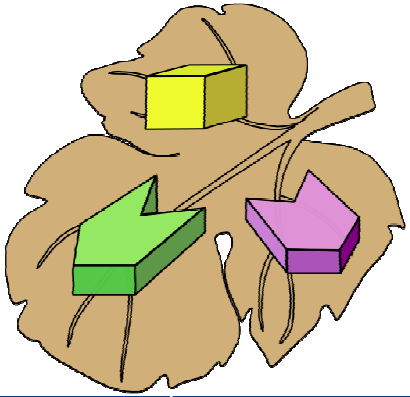
- Distributed deployment with FDF/Deployware

Runtime adaptation & reconfiguration

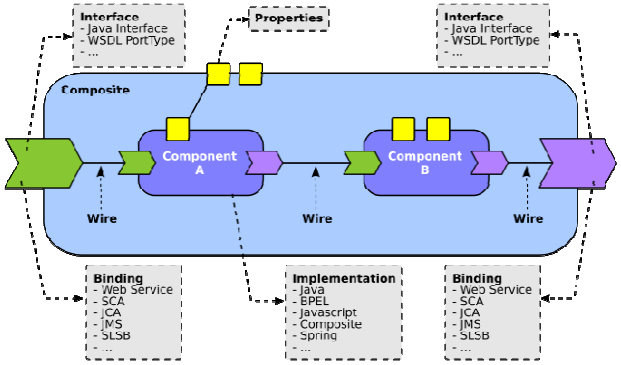
- Introspection & reconfiguration support via Fractal
- Reconfiguration of SCA components & FraSCAti itself

Reflective SCA platform

- Lightweight, efficient, predictable, scalable



FraSCAti: Mixing SCA & Fractal



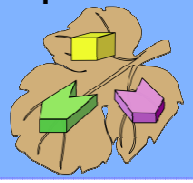
SCA

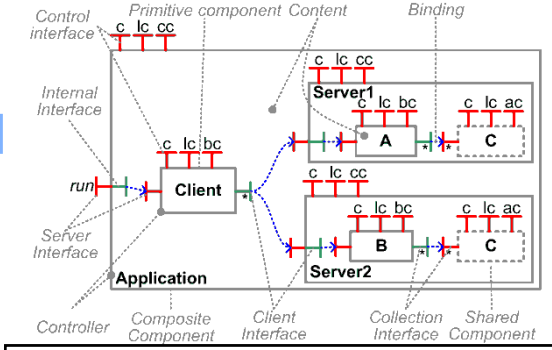
The standard component model for SOA

OW2 Consortium

FraSCAti

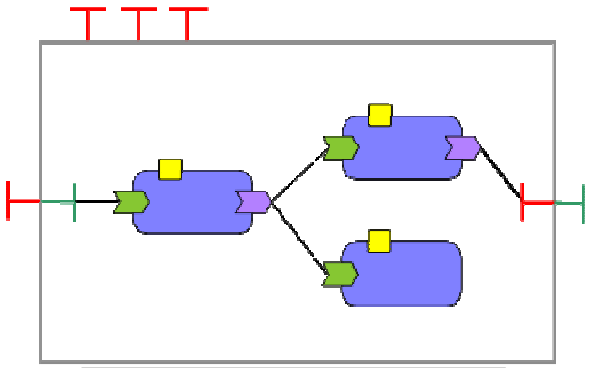
An open SCA runtime platform built on top of OW2 Fractal



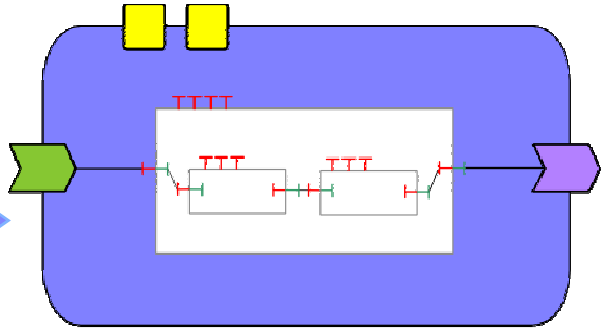


Fractal

A modular and reflective component model



Reconfigurable SCA Applications



SOA for Fractal

FraSCAti Principles

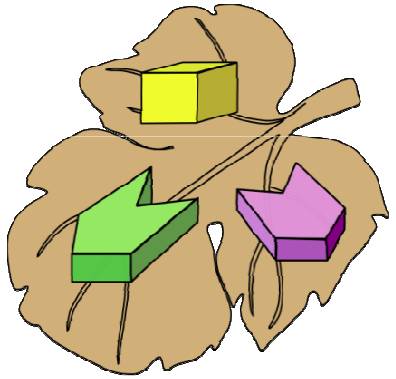
Designed with **adaptability/extensibility/flexibility** in mind

Component-based architecture to support *protocols* and *implementations*

- Communication protocols plugged within a binding factory
- Component implementation languages encapsulated as platform components

AOP-based mechanism to integrate *intents* and *policies*

- Non-functional services developed as regular SCA components
- Non-functional policies dynamically woven into the base architecture



Fractal-based runtime substrate (cf. http://fractal.ow2.org)

- Dynamic reconfiguration capabilities
- Java 5 @-based development style (dependency injection)
- XML-based architecture descriptors
- Structuring concepts (component personality, membrane, control interface, etc.)

2 execution modes for the FraSCAti platform

- Standalone application server (support for 2 backends)
- Integrated in the PEtALS JBI ESB (cf. http://petals.ow2.org)

FraSCAti Features

SCA component implementation

- Java POJO and SCA annotations
- Spring
- Fractal

SCA binding

- Web Services via Apache CXF
- Java RMI

Under development

- OSGi implementation and binding
- JMS, JSONRPC

FraSCAti and SCA Spec.

SCA Specification	FraSCAti	
	State	Component
SCA Assembly Model (v1.0)	☺	Assembly Factory
SCA Policy Framework (v1.0)	☺ / ☹	Assembly Factory
SCA Transaction Policy (v1.0)	☺	Transaction Service
SCA Java Common Annotations & APIs (v1.0)	☺	Tinfi
SCA Java Component Implementation (v1.0)	☺	Tinfi
SCA Web Services Binding (v1.0)	☺	Binding Factory

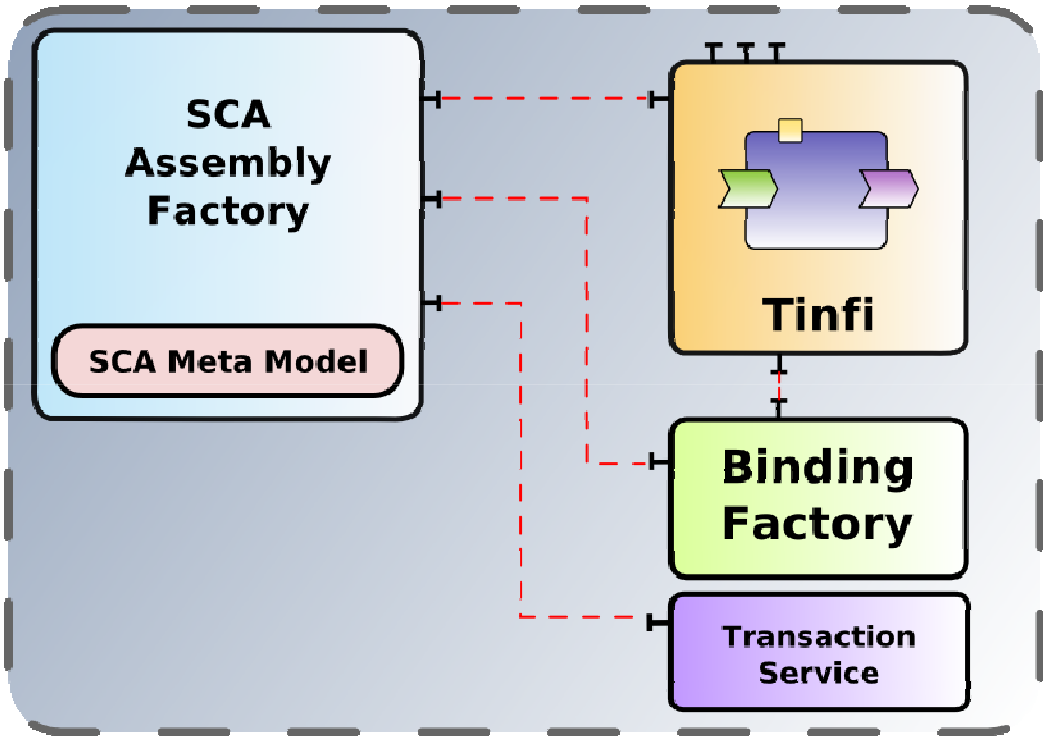
☺ = supported

☺ / ☹ = under development

FraSCAti and SCA Spec.

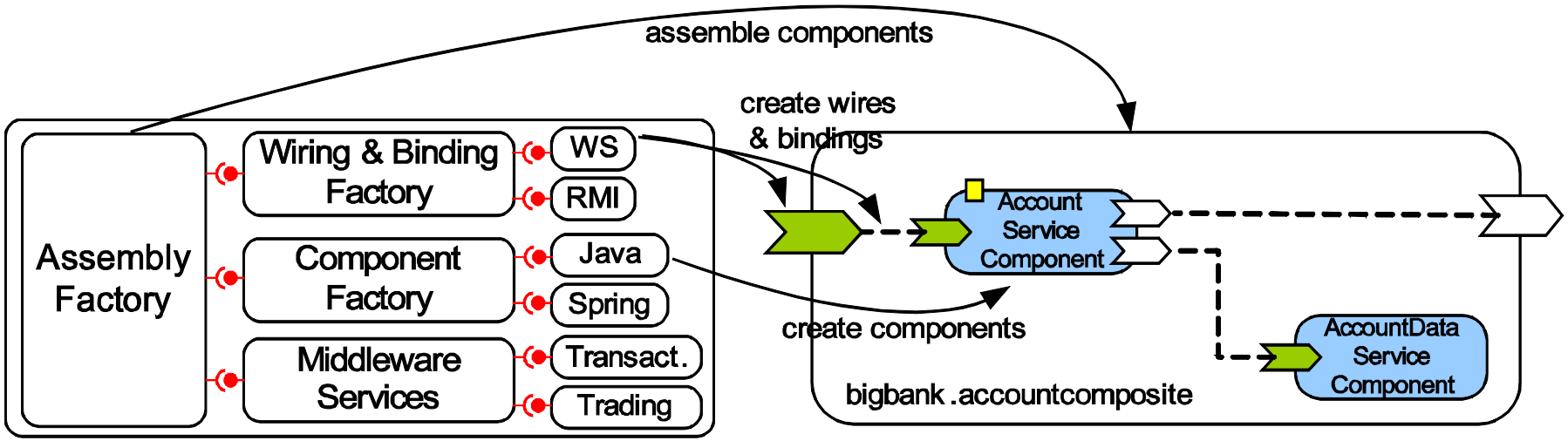
SCA Specification	FraSCAti	
	State	Components
SCA Spring Component Implementation (v1.0)	☺ / ☹	Plug-in Assembly Factory
SCA BPEL Client & Implementation (v1.0)	☹ ☹	Plug-in Assembly Factory
SCA C++ Client & Implementation (v1.0)	☹ ☹ ☹	
SCA C Client & Implementation (v1.0)	☹ ☹ ☹	
SCA COBOL Client & Implementation (v1.0)	☹ ☹ ☹	
SCA JMS Binding (v1.0)	☹ ☹	Plug-in Binding Factory
SCA EJB Session Bean Binding (v1.0)	☹ ☹	Plug-in Binding Factory
SCA JCA Binding (v1.0)	☹ ☹ ☹	Plug-in Binding Factory
SCA Java EE Integration (v0.9)	☹ ☹ ☹	

FraSCAti Architecture

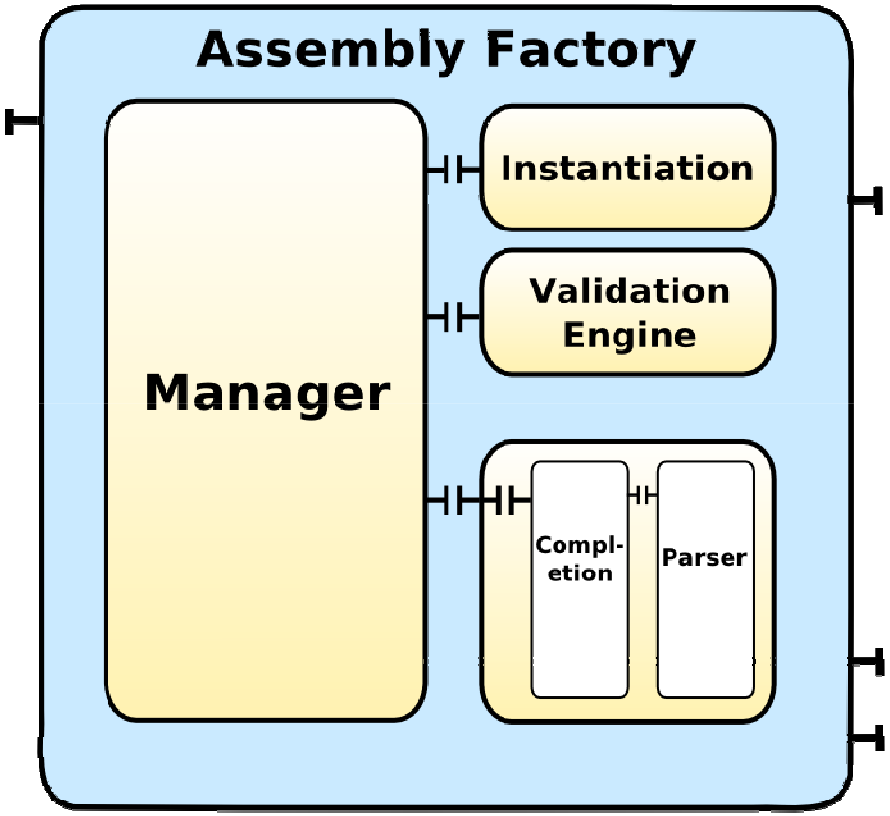


- **Tinfi** generates the SCA components' glue code and create component instances
- **Binding Factory** imports & exports the SCA components via specific communication protocols
- **Transaction** controls local & distributed transactions between the SCA components
- **Assembly Factory** processes and deploys SCA assembly models

FraSCAti Architecture



FraSCAti Assembly Factory



- **Manager** loads resources and invokes sub components
- **Parser** creates a model instance from composite definition and implementation.
  Use Eclipse STP SCA model
- **Validation Engine** validates additional model constraints.
  Implementation in progress
- **Instantiation** creates new component instances.
  Use Tinfi & Binding Factory

Tinfi Is Not a Fractal Implementation

- Builds membranes which provides a control semantics to achieve a SCA personality for a component
- A scaPrimitive component
  - is a Fractal component
  - can be assembled with other Fractal components
- OSOA 1.0 API coverage by Tinfi — Implemented
  - 5 interfaces (+ 1 for constants) — 5/5
  - 4 exceptions
  - 25 annotations
  - 17 general-purpose — 17/17
  - 2 remote communications — 0/2
  - 6 policy intent sets — 2/6
  - To be done : @Remotable, @AllowsPassByReference, @Authentication, @Confidentiality, @Integrity, @Qualifier

Tinfi

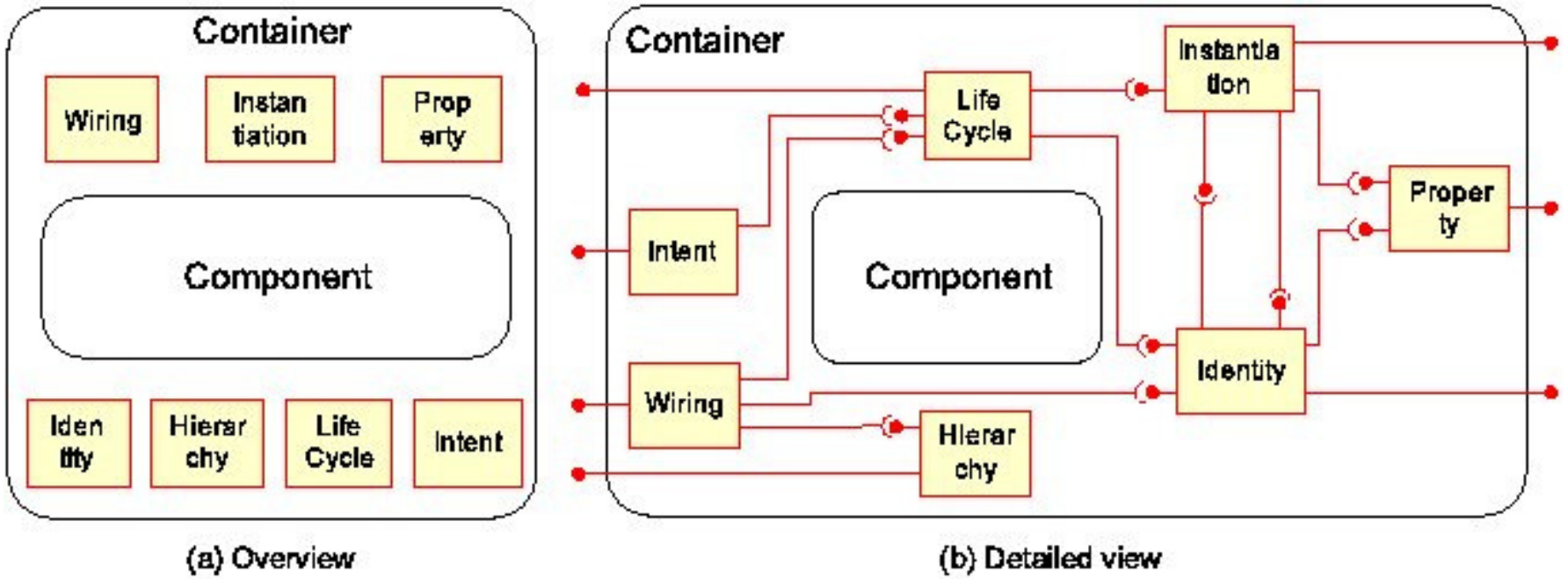
6 controllers

- **SCAComponent** : component identity dedicated interface (ComponentContext) and implementation
- **SCAContentController** : component instantiation policy dedicated implementation, private interface (no need to export it)
- **SCAIntentController** : intent handlers management
- **SCAPropertyController** : component properties management
- **SCALifeCycleController** : component initialization (@EagerInit) same interface as Fractal LC, dedicated implementation
- **SCABindingController** : component bindings same interface as Fractal BC, dedicated implementation

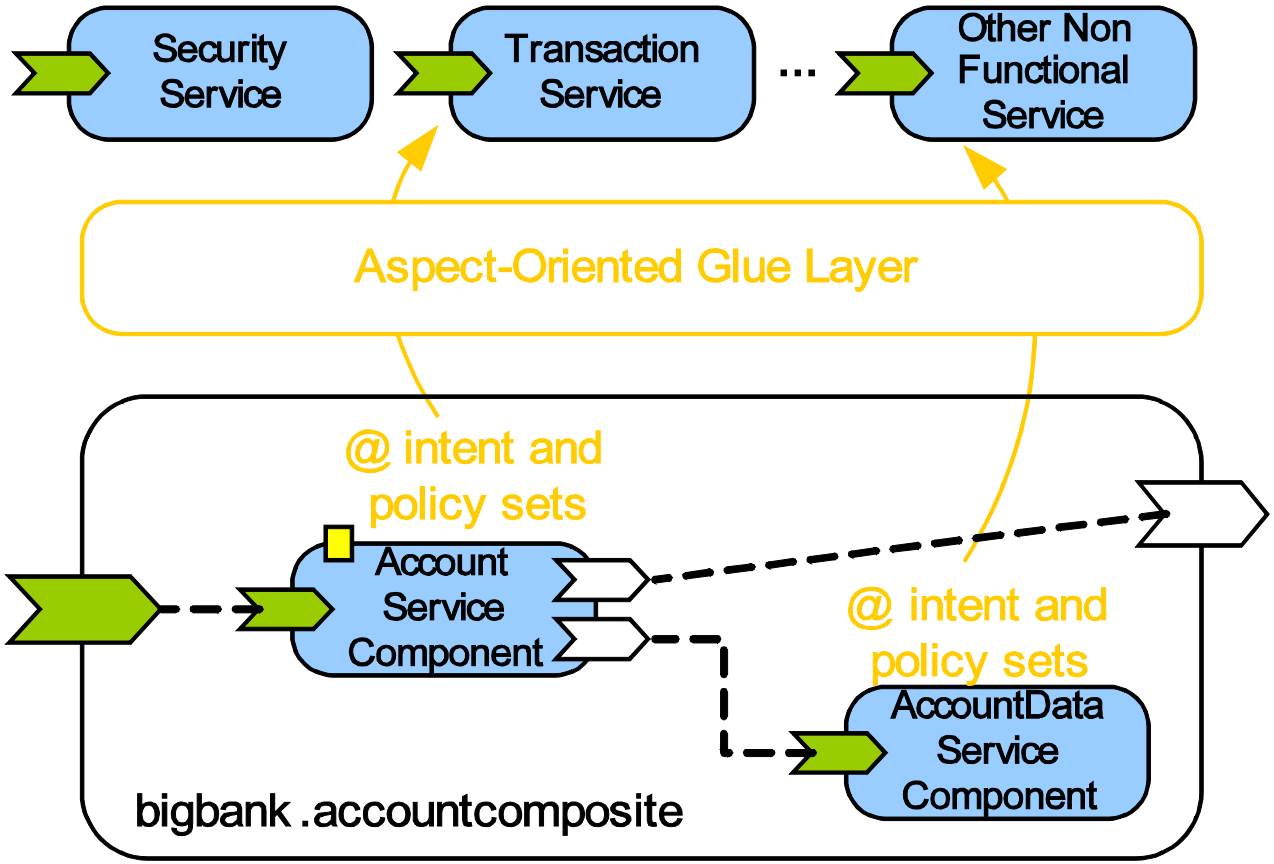
Interceptors

- lifecycle management
- component instantiation policy
- intent dispatch

Tinfi architecture



Tinfi
> weaving intents & policies

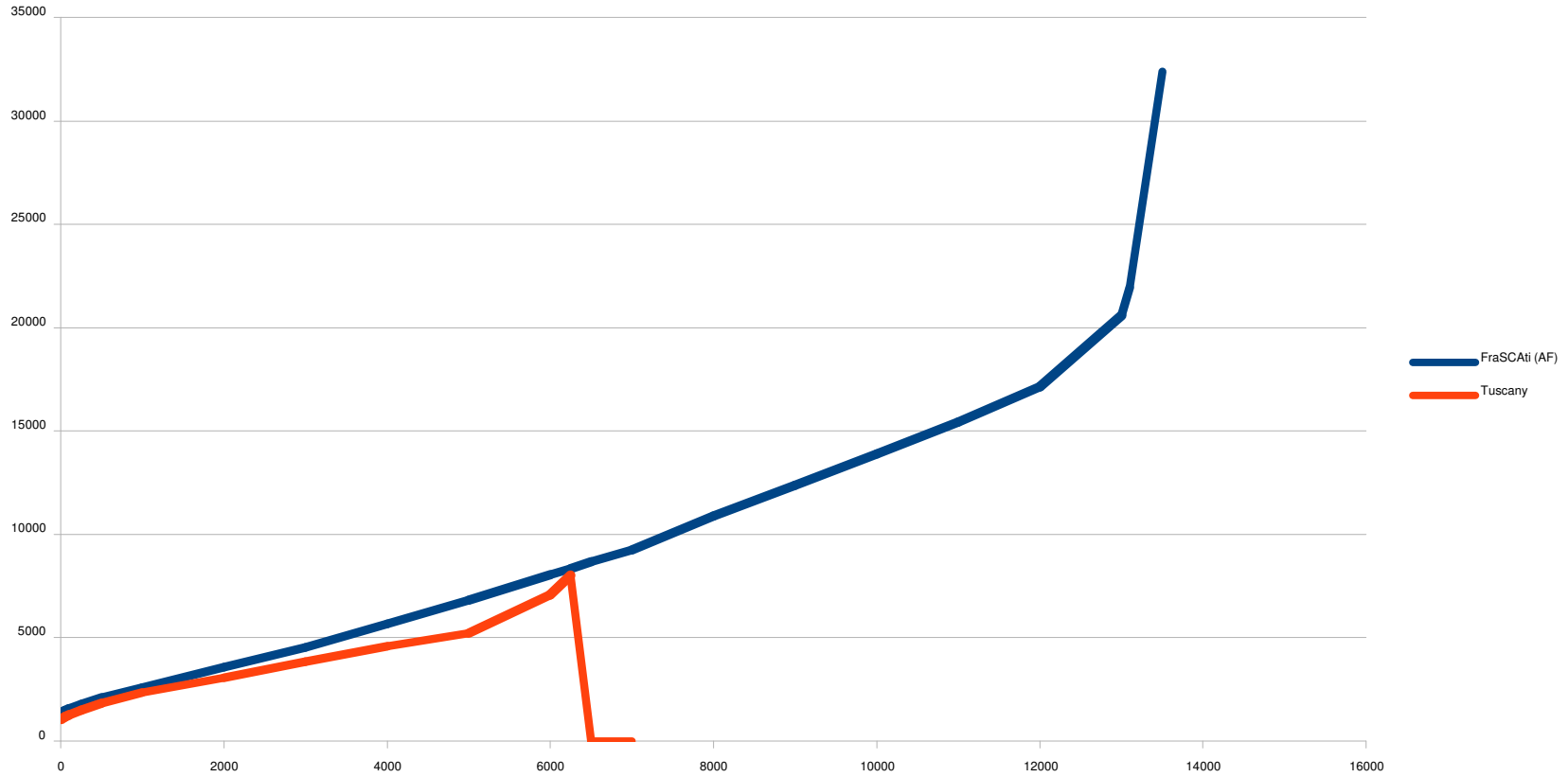


FraSCAti Vs. Apache Tuscany

- ☹ *Less SCA features* supported
  - Less implementation languages and binding protocols
- ☹ Smaller ecosystem
  - Less sponsoring companies, developers, and users
- ☺ **Better continuum** from SCA tooling to runtime platform
  - Share the same SCA metamodel with Eclipse STP SCA project
- ☺ **Better footprint** to target embedded systems
  - Smaller disk and memory footprints
- ☺ Ready for **dynamic runtime reconfiguration**
  - Based on OW2 Fractal component model and associated tools

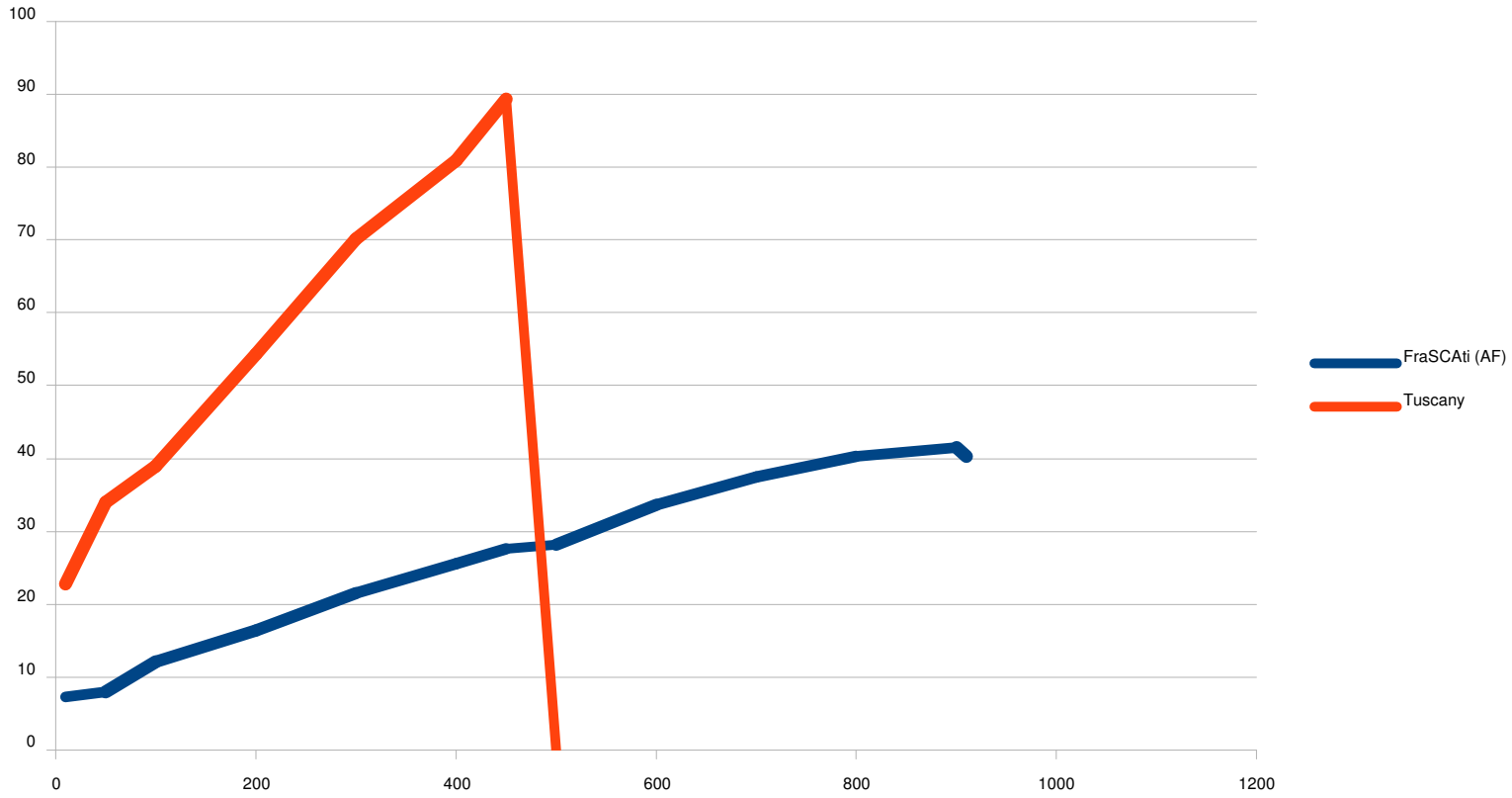
FraSCAti Vs. APACHE TUSCANY

Deployment performance evaluation



FraSCAti Vs. Tuscany

Runtime performance evaluation



Other OSS Competitors

- Fabric3 
  - ☹/☺ Fork from the Apache Tuscany project
  - ☹ Developed by fewer contributors
- The Newton Project 
  - ☺ Distributed runtime framework based on OSGi, Jini, and SCA
  - ☺ SCA bindings for OSGi and Jini
  - ☹ Does not target a fully-compliant SCA framework
    - ☹ No support for SCA Java annotations
    - ☹ No Web Service binding
- The Mule Project - MuleSCA activity 
  - ☺ Some Web pages
  - ☹ No open source code currently available

FraSCAti Perspectives

- INRIA ADT galaxy – Agile SOA Platform
  - SCA management at runtime
    - FraSCAti Explorer
    - Eclipse STP/SCA Composite Designer
  - SCA scripting with FScript
  - SCA monitoring with WildCat
  - SCA BPEL Client and Implementation (v1.0)
- CAPPUCINO – eCommerce
  - FraSCAti in mobile devices (PDA & smart phones)
- ANR ITEmIS – Marriage of IT and embedded systems
  - PEtALS/FraSCAti & OSGi
  - FraSCAti & JMS/JORAM
  - Formal specification of SCA
- IST SOA4All – A Web of billions of services
  - Large-scale PEtALS/FraSCAti deployment
  - SCA binding for SOA4All Semantic Spaces

FraSCAti Explorer

Load SCA composites

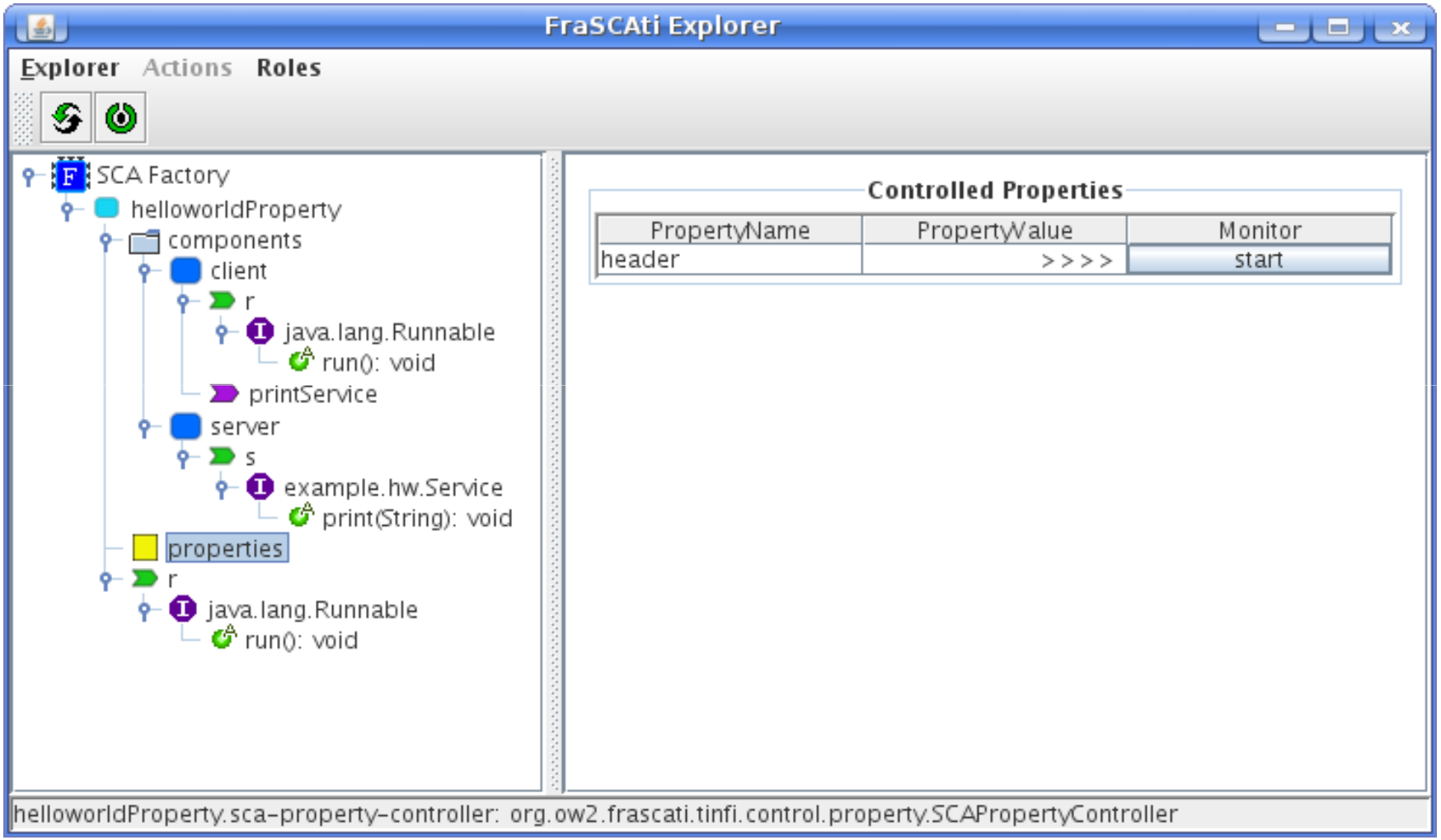
Visualize, introspect, navigate within SCA composites

- Components
- Services / References / Bindings / Wires
- Properties
- Intents

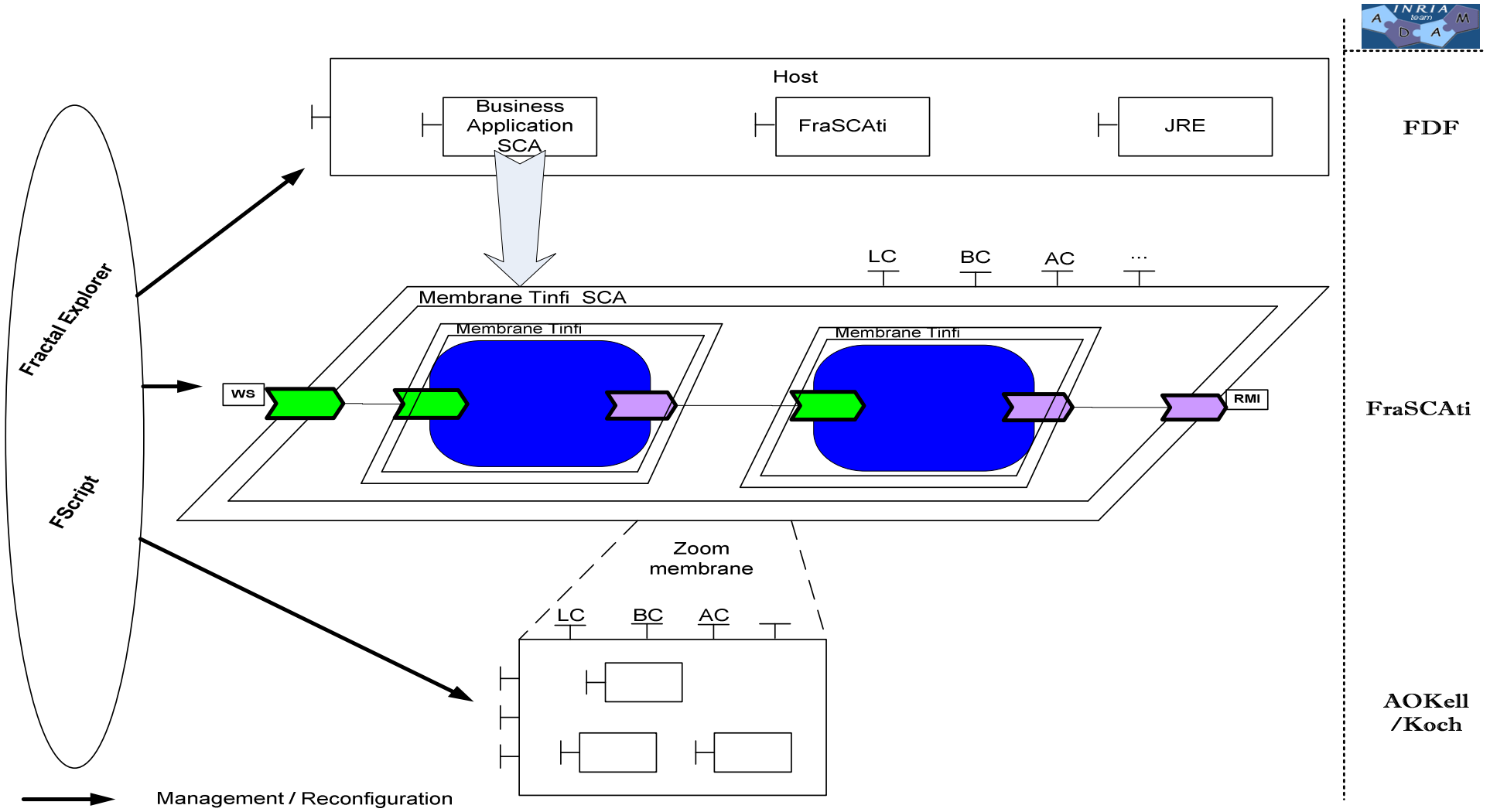
Reconfiguration actions

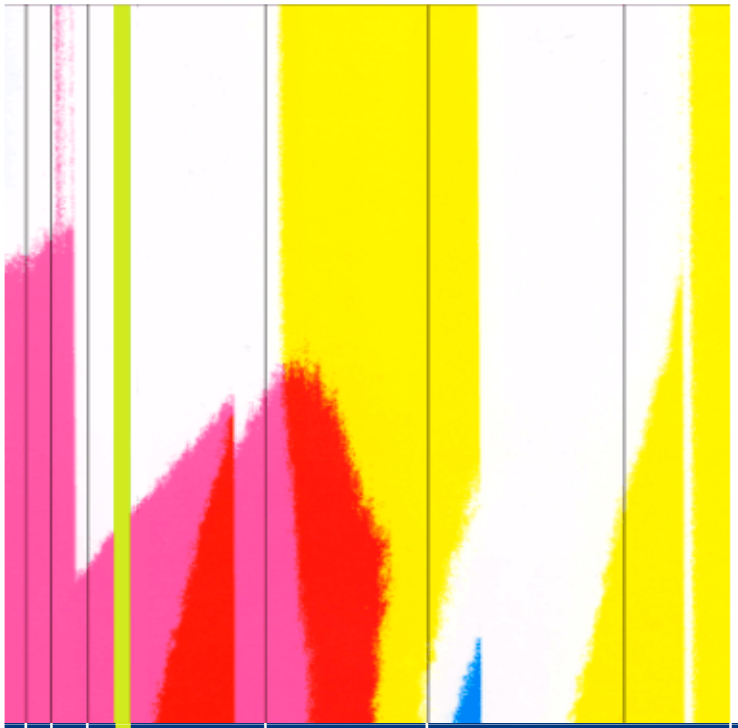
- Start/stop SCA components
- Add/remove SCA components / wires / bindings / intents
- Update SCA properties / intents
- Update SCA bindings
  - Web service address
  - Java RMI port and exported name

FraSCAti Explorer



Reflective SCA Systems





SCOrWare

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
LILLE - NORD EUROPE

Université
Lille1
Sciences et Technologies

lifL

cnrs
dépasser les frontières

INRIA
Team
A D A M

FraSCAti Ecosystem

FraSCAti platform

- Open-source implementation of the SCA specifications
- Developed in the context of the ANR SCOrWare project
  - 2-year "precompetitive" project

•Industrial partners	•Academic partners
Artenum, Paris Science & Groupware Edifixio ebm WEBSOURCING beo Open Wide Architecte Open Source	INRIA ADAM OW2 SARDES INP ENSEEIHT IRIT UMR 5505 – CNRS – INP ENSEEIHT – UPS INT

Project leader: Philippe Merle (INRIA ADAM)

ANR SCOrWare Objectives

- Promote the development of SCA-based applications
- Provide an integrated development environment for SCA-based development
- Provide an **open and flexible platform for SCA**
- Bring **dynamicity and reconfiguration to SCA** applications
- Leverage the integration of JBI and SCA
- Contribute to the **open-source ecosystem**

ANR SCOrWare Keystones

Demo

Designer

Deployment

Runtime

Applications

Modeling tools for
- Eclipse STP SCA assembly definition
- TUNe / **DeployWare**

Deployment
- TUNe (autonomy)
- **DeployWare**

Runtime support : **FraSCAti**
- Assembly Factory + Binding Factory + Tinfi + Transaction

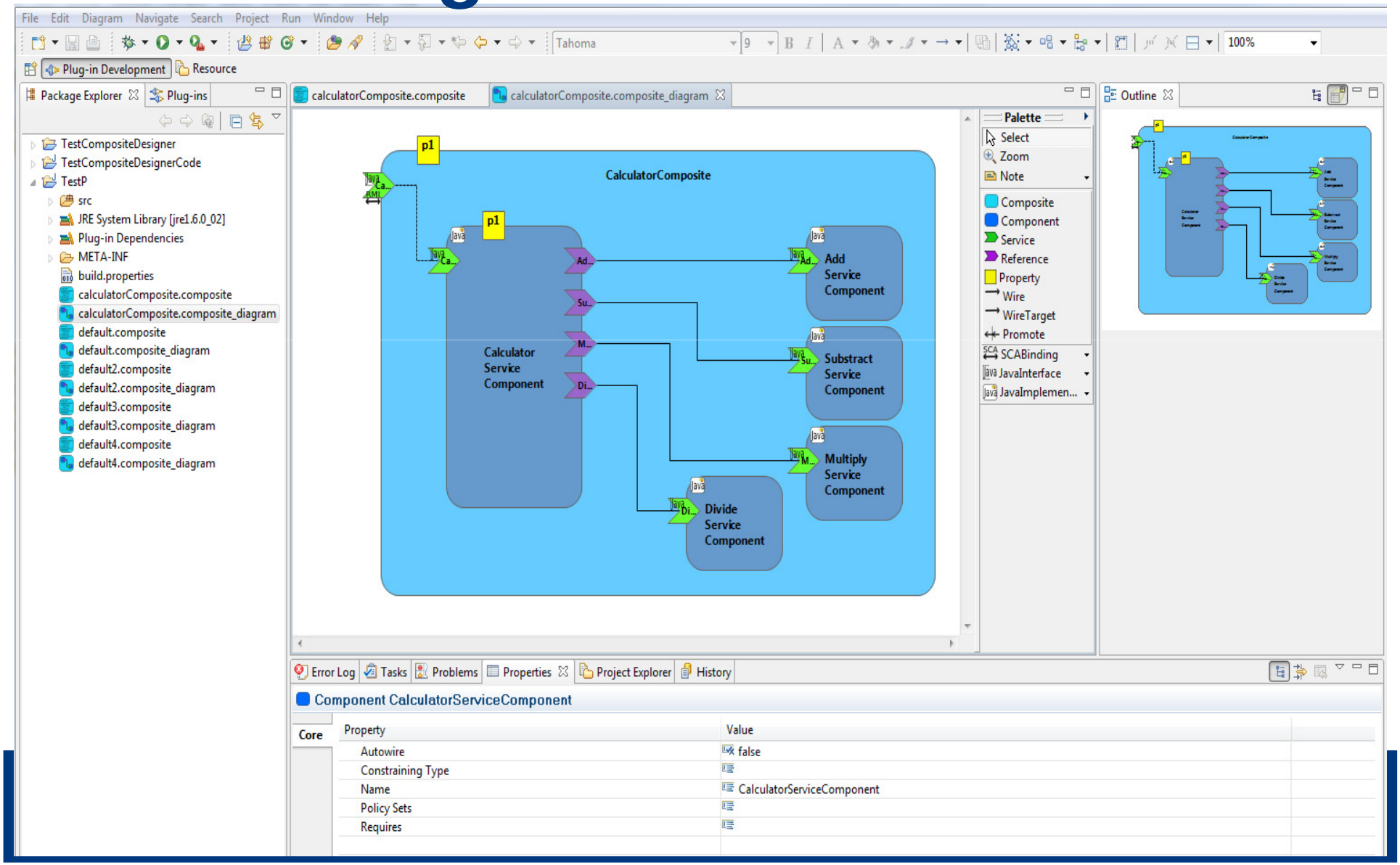

Software Developer


Software Architect


System Admin

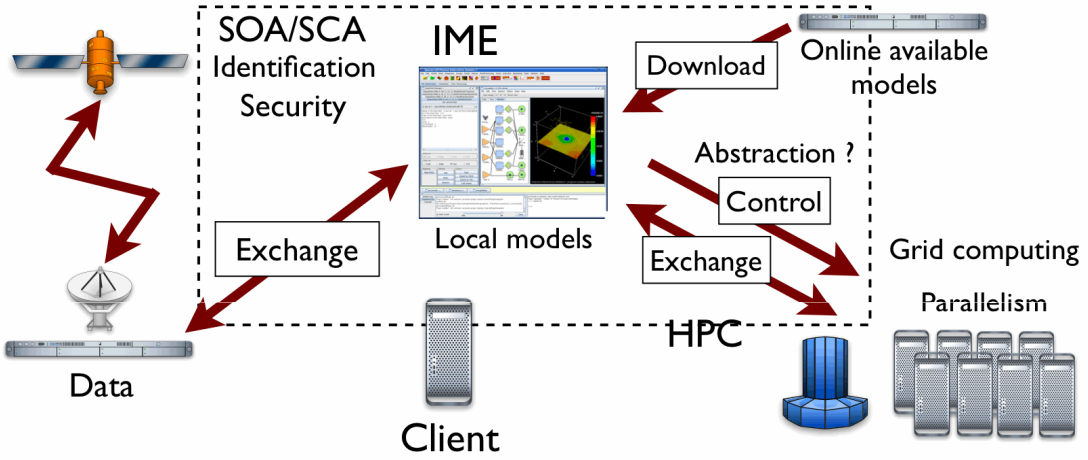

Platform Provider

Modelling with Eclipse STP SCA tooling

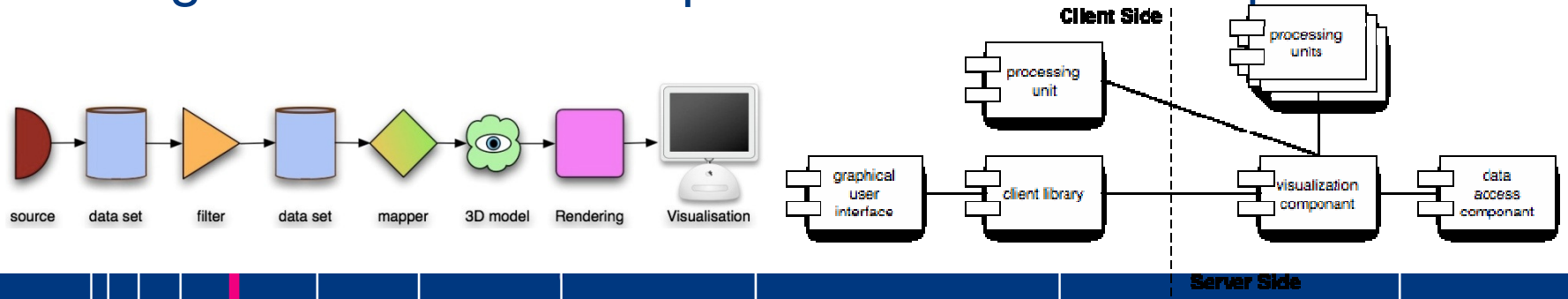


ANR SCOrWare Use Case (Artenum)

- Trend: Service-oriented scientific computing (***Computing On Demand***)



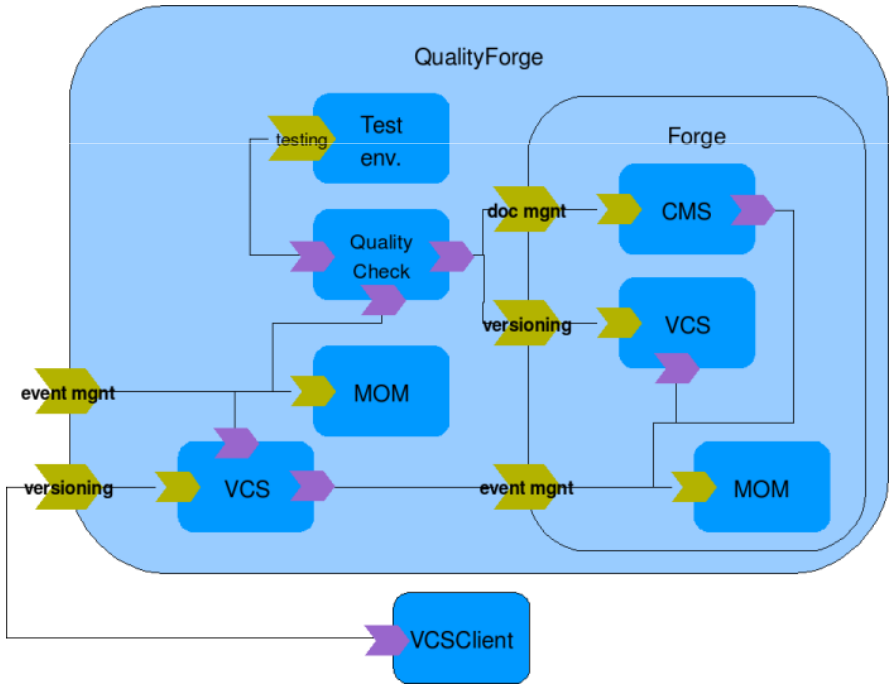
- Defining SCA-based interoperable scientific components

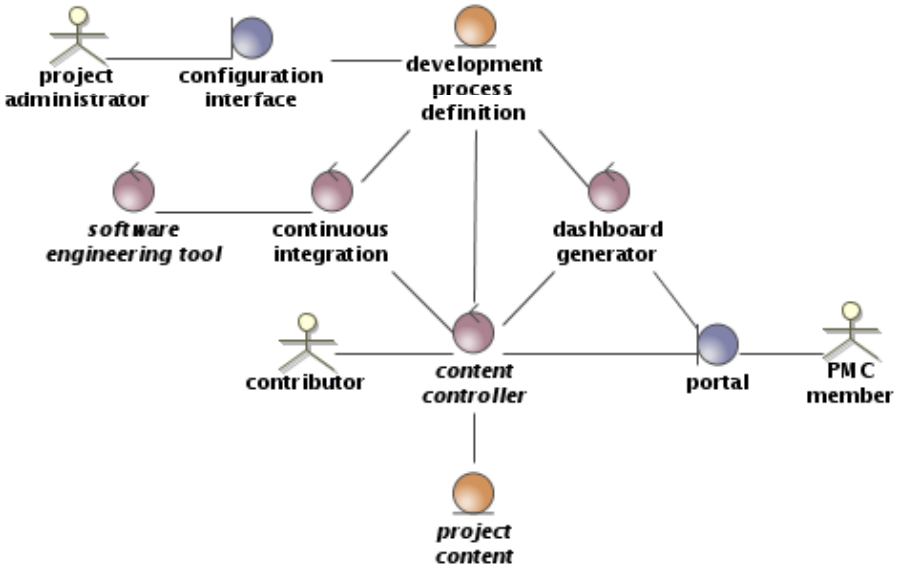


ANR SCOrWare Use Case (INRIA OW2)

New generation source forge

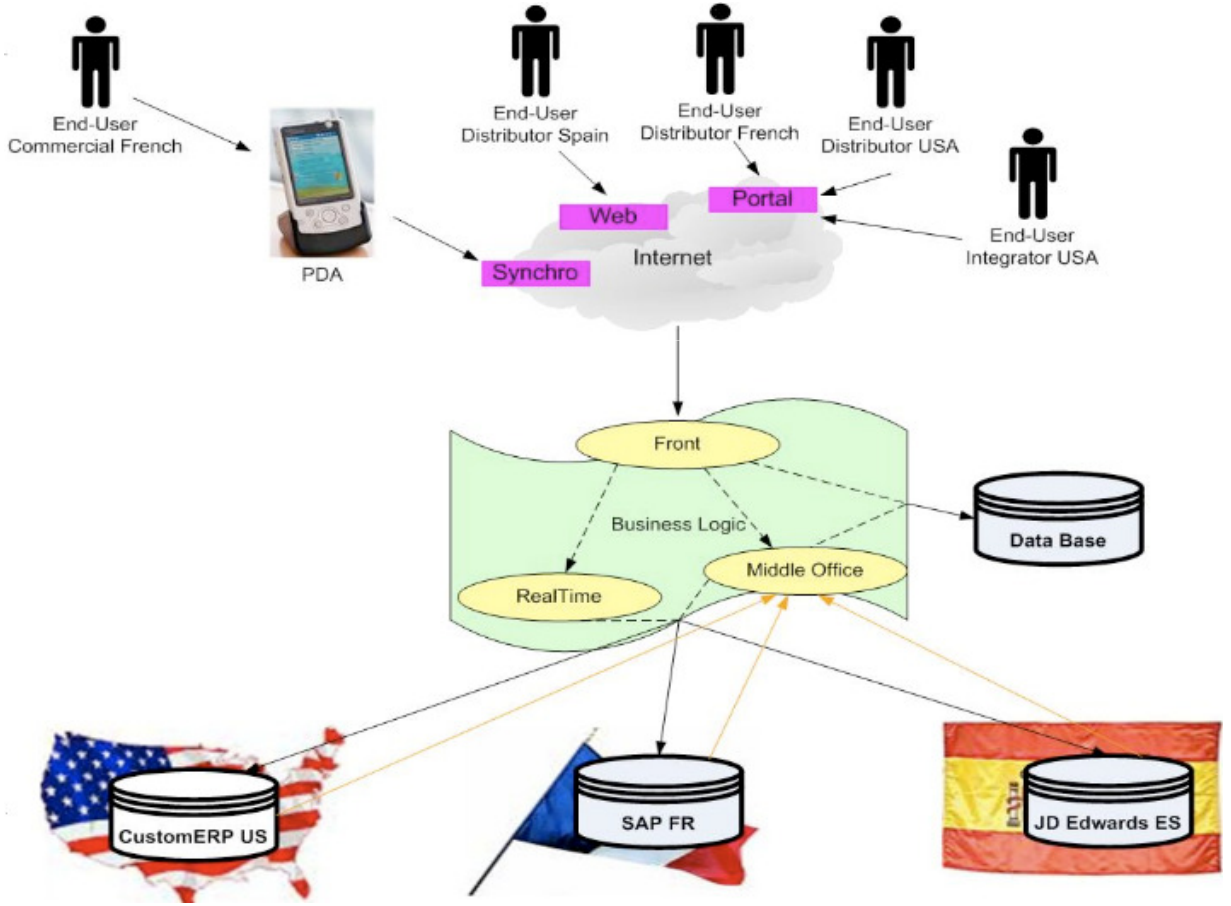
- Service-oriented components for content management, continuous build, release management…





ANR SCOrWare Use Case(Edifixio)

e-Business integration with SCA

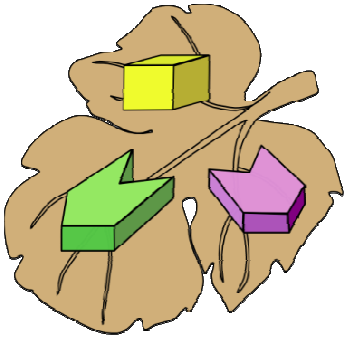


Conclusion

FraSCAti

- an open and extensible implementation of the SCA specifications
  - continuum from tooling to runtime (common SCA metamodel shared with STP)
  - reconfigurable SCA applications
  - lightweight version for embedded devices currently being developed
- based on OW2 code blocks
- developed by the ANR SCOrWare project (ending 04/2009)

FraSCAti Contact



Website

- http://frascati.ow2.org
- http://www.scorware.org

Project heads

- Philippe Merle: Philippe.Merle@inria.fr
- Lionel Seinturier: Lionel.Seinturier@univ-lille1.fr

Development team (core)

- INRIA ADAM & SARDES

Acknowledgements

- Damien Fournier, Valerio Schiavoni, Nicolas Dolet, Vivien Quéma, Jean-Bernard Stefani, Alain Boulze, Adrian Mos, Christophe Demarey, Adrien Louis, Stéphane Bagnier, Daniel Hagimont, Etienne Juliot, Gaël Blondelle, Jean-Pierre Lorre, Marc Dutoo, Marc Pantel, Mickael Istria, Mohammed Eljai, Nicolas Salatge, Samir Tata, Roland Naudin, Samuel Quaireau, Stéphane Drapeau, Thomas Darbois
- and all SCOrWare project members (past and present) that I may have forgotten…

SCA References

SCA Specifications

- OpenSOA http://www.osoa.org
- OASIS OpenSCA http://www.oasis-opencsa.org

OSS Implementations

- Tuscany http://tuscany.apache.org
- Newton http://newton.codecauldron.org/site/index.html
- Fabric3 http://xircles.codehaus.org/projects/fabric3
- FraSCAti http://www.scorware.org

SCA Resources

- http://www.osoa.org/display/Main/SCA+Resources
- http://www-128.ibm.com/developerworks/library/specification/ws-sca
- http://www.davidchappell.com/articles/Introducing_SCA.pdf
- http://www-128.ibm.com/developerworks/websphere/techjournal/0510_brent/0509_brent.html
- http://events.oasis-open.org/home/sites/events.oasis-open.org.home/files/Flexible_Agile_Composition_01.ppt [Mike Edwards]
- http://www.osoa.org/download/attachments/250/Power_Combination_SCA_Spring_OSGi.pdf?version=3