



USP - Universidade  
de São Paulo



IME - Instituto de  
Matemática e Estatística

# Web Services Choreography

Victoriano Alfonso Phocco Diaz

[alfonso7@ime.usp.br](mailto:alfonso7@ime.usp.br)

Glaucus Augustus G. Cardoso

[glaucus@ime.usp.br](mailto:glaucus@ime.usp.br)

Orientador: Prof. Marco Aurélio Gerosa

[gerosa@ime.usp.br](mailto:gerosa@ime.usp.br)

# Agenda

- Basic Concepts
- Web Services Composition
- Choreography
- Standards
- Choreography Languages
- WS-CDL
- Issues, Challenges and Trends
- Development and Tools
- Conclusions

# Agenda

- Basic Concepts
- Web Services Composition
- Choreography
- Standards
- Choreography Languages
- WS-CDL
- Issues, Challenges and Trends
- Development and Tools
- Conclusions

# Basic Concepts

# What is Web Services?

“A Web service is a software system designed to support interoperable machine-to-machine interaction over a network.”[1]

It was built to solve the problem of system integration

It is mainly based on:

- WSDL, which describes what types of data are exchanged and the available operations.
- SOAP, which specifies the pattern to call a specified service
- UDDI, which makes the services discoverable

# What is SOA?

“SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.” [1]

In SOA, systems are build by grouping services (not web services necessarily)

# SOA goals

It provides a way to use services as reusable software components

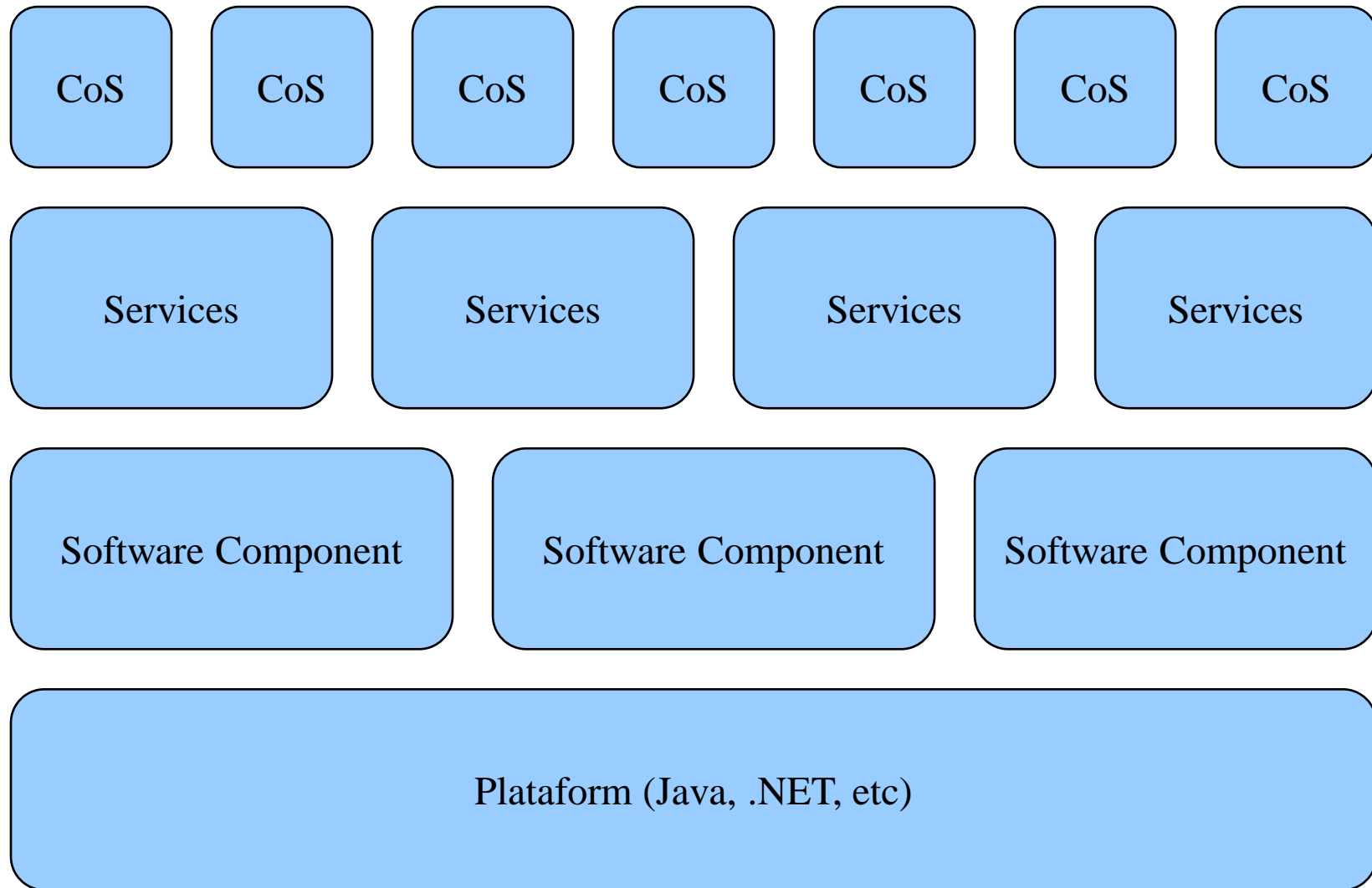
It improves manageability of large-scale systems

It makes easier the use of Internet advantages related to scalability, cutting costs of cooperation

It offers a scalable paradigm to organize large systems that require interoperability to meet its goals

It is possible to implement a system using SOA through distributed systems technologies such as CORBA, COM, and RMI

# SOA-based system



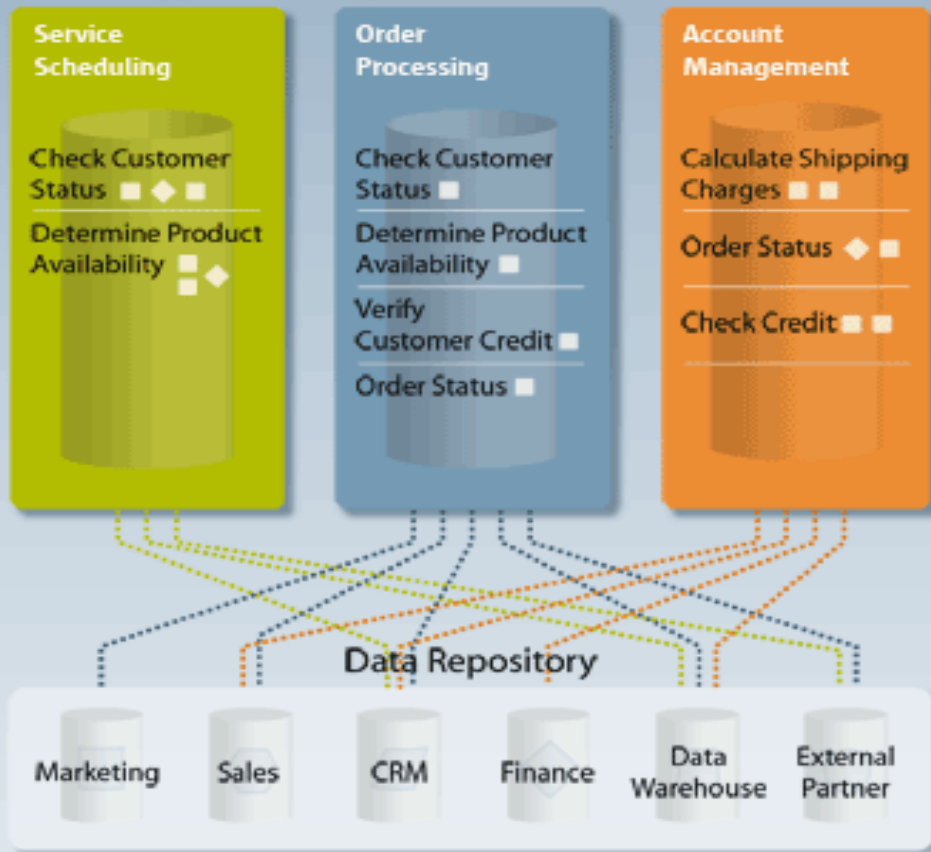
\* CoS (Composition of Services)



## Before SOA

Siloed · Closed · Monolithic · Brittle

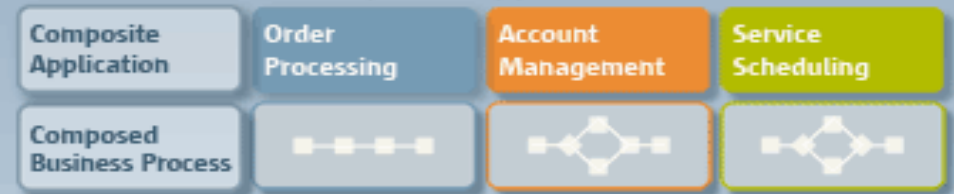
### Application Dependent Business Functions



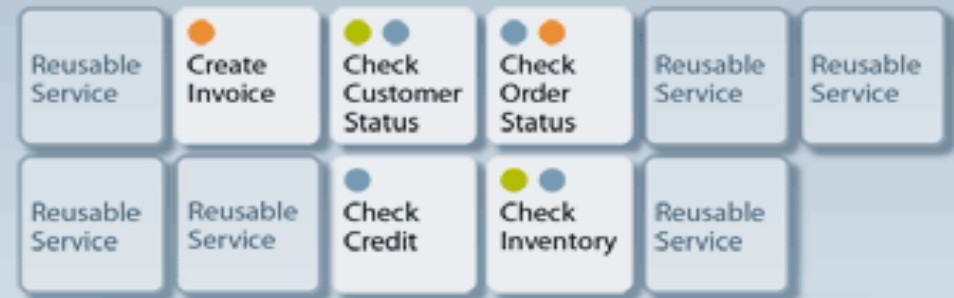
## After SOA

Shared services · Collaborative · Interoperable · Integrated

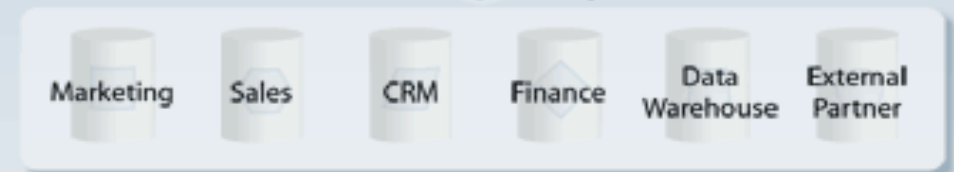
### Composite Applications



### Reusable Business Services



### Data Repository



**Fig.: SOA architecture**

from [http://ie.sun.com/practice/software/soa/images/ig\\_soa\\_before.gif](http://ie.sun.com/practice/software/soa/images/ig_soa_before.gif)

# Web Services Composition

# What is it?

It is the grouping of services to produce other services (more complex)

Compositions can be complex and aggregate a lot of services that may be within different domains

Composition is mostly done manually, although a few ways to do it automatically have been studied

The service composition is made possible mainly due to SOA

“The full potential of Web Services as an integration platform will be achieved only when applications and business processes are able to integrate their complex interactions by using a standard process integration model. The interaction model that is directly supported by WSDL is essentially a stateless model of request-response or uncorrelated one-way interactions.”[1]

[1] - <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>

# Approachs

## **Problem**

SOAP+WSDL standards only supply atomic interactions and do not keep the state of the interation

## **Solution**

Orchestrations and Choreographies supply a dynamic interaction and integration with state maintenance (stateful).

# Choreography

# What is it?

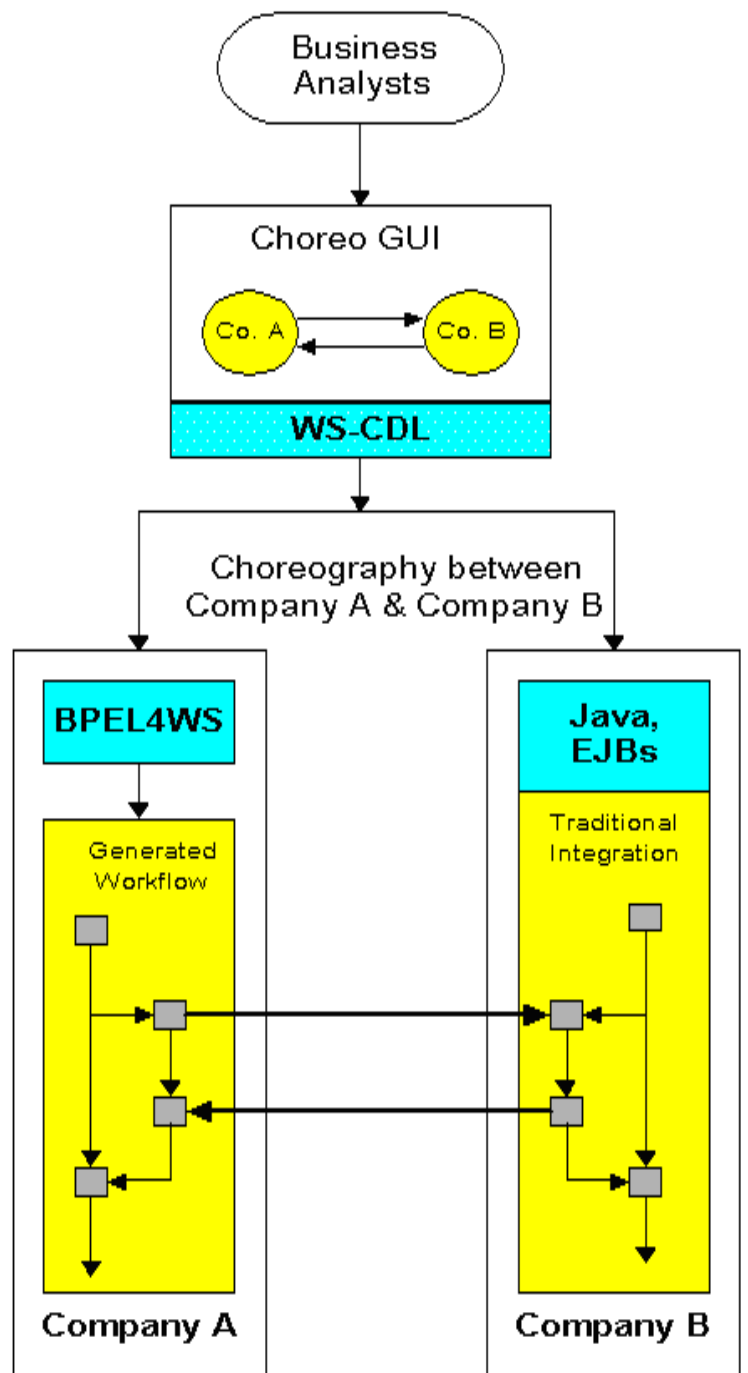
Choreography is the process of execution of business logic where each participant involved are responsible for execution of its own task (there is no central controller)

In choreography, the participants are able to identify a sequence of messages, being stateful

Choreography languages define rules to services intercommunication while orchestration languages are executable and define operators to execute, catch exceptions, do parallelism, etc

# How is Choreography Used?

- Precisely defines the sequence of interactions between a set of cooperating WS
- Generates the necessary code skeletons that can be said to implement the required external observable behavior of a WS



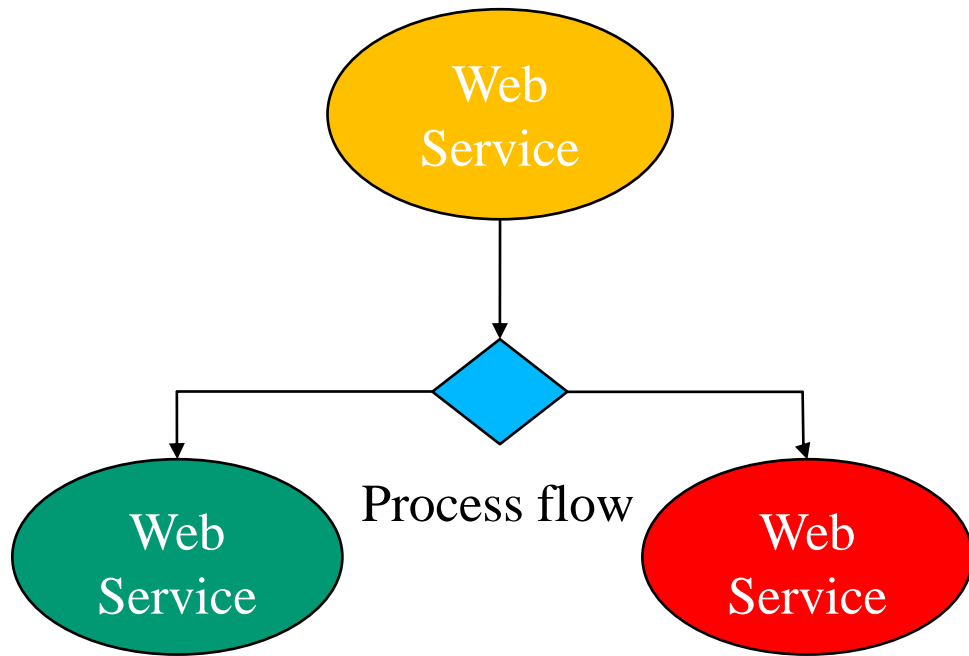


# Benefits of Choreography Language

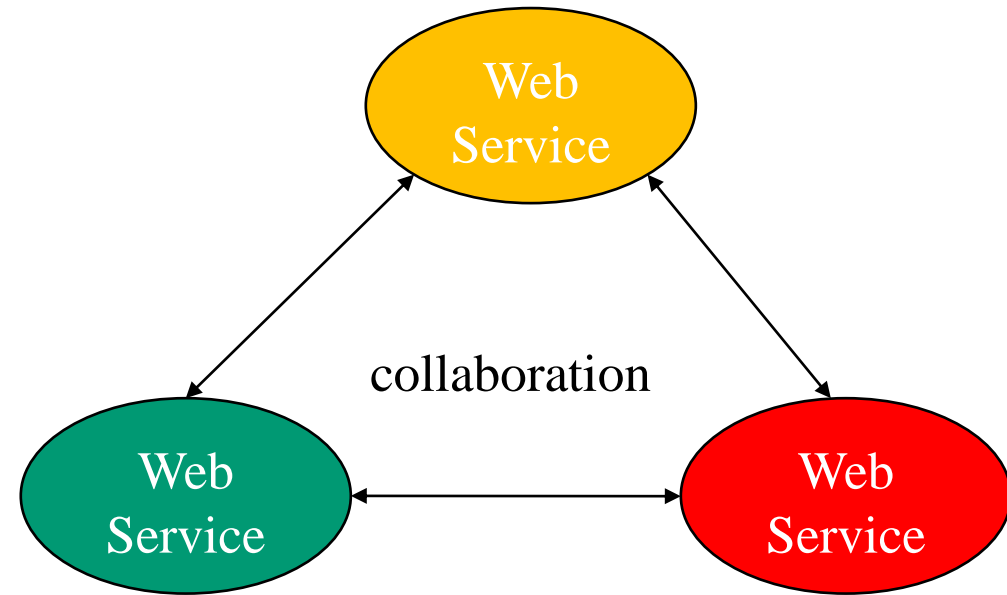
- All uses of a choreography description need a standardized language
- Enables the construction of more robust WS
- More effective interoperability
- Reduces the cost of implementing WS
- Increases the utility of WS

# Orchestration Vs Choreography

- Different aspects to create business process from integrated web services, but are complementary.
- Sometimes look like a synonymous.
- **Orchestration:**
  - Single agent that controls and coordinates interactions
- **Choreography:**
  - Descriptions of observable behavior
  - Observable message exchanges between a collection of services



**Web Services  
Orquestration**



**Web Services  
Choreography**

**Fig. : Orchestration and Choreography**

# Standards

# Major Consortia

## **World Wide Web Consortium (W3C)**

Was created in October 1994 to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability

## **Organization for the Advancement of Structured Information Standards (OASIS):**

Is a not-for-profit, global consortium that drives the development, convergence, and adoption of e-business standards

## **Web Services Interoperability Organization (WS-I)**

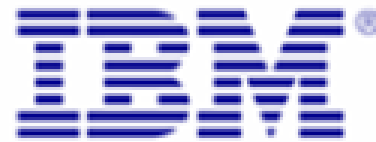
Is an open, industry organization chartered to promote Web services interoperability across platforms, operating systems, and programming languages

FUJITSU

SOA | software™



ORACLE®



Microsoft®



invent



STERLING  
COMMERCE

PIVOTAL

autodesk

COMPAQ

TOSHIBA

IBM

SAP

accenture

MCAFFEE.COM

UNITED

ORACLE

intel

Peregrine SYSTEMS  
Partners in Business

Microsoft

FUJITSU

KPMG Consulting

GRAND CENTRAL COMMUNICATIONS

DAIMLERCHRYSLER

flamenco networks

Ford Motor Company

amcracker

LOUDCLOUD

hp invent

IONA

POSC

epicentric

bea

CommerceQuest

KANA

CAPE CLEAR

epicor

REUTERS

BUSINESS OBJECTS

Rational  
the software development company

corechange

SYBASE

plumtree

REED ELSEVIER

FrontRange SOLUTIONS

Borland

COMMERCE ONE

sas

Sabre

FileNET

macromedia

groove NETWORKS

Qwest  
ride the light

Akamai

webMethods

JDE EDWARDS

DASSAULT SYSTEMES

RealNames

VeriSign

versata  
Automating e-Business

# Standards Evolution

- **WSFL** by IBM.
- **XLANG** by Microsoft .
- **WSCI** – Web Services Choreography Interface  
Sun, SAP, BEA, and Intalio (august 2002).
- **BPEL4WS** (a.k.a. BPEL) – Business Process Execution Language for Web Services - IBM and Microsoft (April 2003).
- **BPML** – Business Process Management Language.  
BPMI.org (chartered by Intalio, Sterling Commerce, Sun, CSC, and others), Now is **BPMN**.
- **BPSS** – now is **ebXML** of OASIS (started in 1999).



# Standards Evolution

- **XLANG**, **WSFL** and **WSCI** are no longer being supported by any standard organization.
- **WS-BPEL** (by OASIS) replace XLang.
- **WSFL** and **WSCI** was superseded by **WS-CDL** (by W3C, November 2005).
- **BPMN** 2.0 (september 2009) introduces diagrams for specifying service choreographies.
- The academic field has put forward other service choreography languages, for example **Let's Dance** (2006), and **BPEL4Chor** (from July 2007).

# Comparing BPEL with CDL

## **BPEL**

- Orchestration implies a centralized control mechanism.
- Recursive Web Service Composition.
- Executable language.

Requires Web Services.

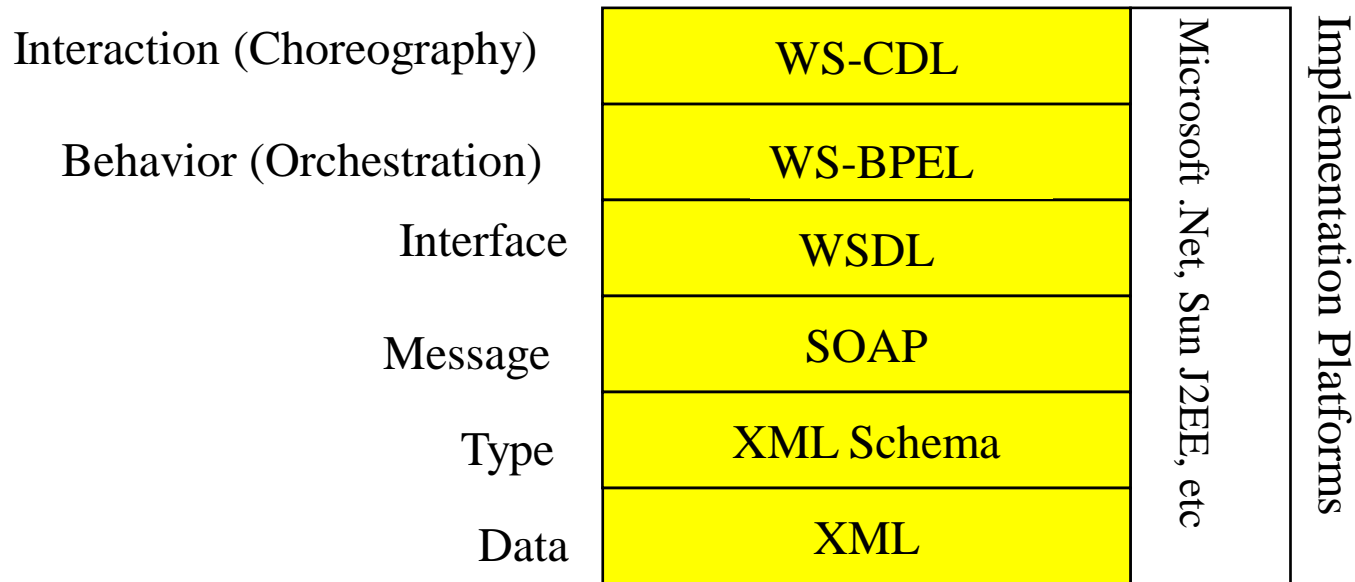
## **.CDL**

- Choreography has no centralized control. Instead control is shared between domains.
- Description language.
- Does not need Web Services but is targeted to deliver over them.

CDL doesn't see BPEL is unique or different to any other end-point language. Hence you can generate to BPEL just as easily as you can do to Java

# Protocols stack

- **Loosely coupled**, interaction through standardized interfaces
- **Standardized** data transmission via XML
- **Asynchronous messaging**
- **Platform independent** (.NET, J2EE)



Web Service Standards

# Choreography Languages

# WSCI (Web Service Choreography Interface)

- The specification supports message correlation (like state full), sequency rules, treatment of exceptions, transactions and dinamic colaboration
- Describes the observable (view of an external observator) behaviour of the web services

# WSCI some features

- **Messages choreography:** Describes the order and the period of message exchange
- **Transactional control:** Describe the transactional operations and the "undo" steps
- **Treatment of exceptions:** Describe how the service will treat the exception and the possible flow sequence
- **Thread management:** Describe how the service can manage multiple interactions with multiple services

# WS-CDL

- Describes the behavior observable and common to a group of participants who cooperate to execute one activity.
- Specifies specially the exchange of information and the conditions of ordination that must be meet.
- It is supported by The W3C Web Services Choreography Working Group, actually is a Candidate Recommendation.
- Latest version: W3C Working Draft 19 June 2006.

# WS-CDL - features

Local choreographies can be generated from WS-CDL also skeletons for orchestration

**Reusability** : The same condition must be reusable by other participants in other contexts

**Cooperation** : Choreographies must define the sequence of message exchange between the participants

**Multi-collaboration**: Choreographies can have any number of participants and process



# WS-CDL - features

**Composition** : Choreographies can be combined to make a new one, that can be executable in other context

**Modularity** : Choreographies can be defined just including others choreographies

**Information-oriented collaboration** : Choreographies describes how their participants go through the execution based in the exchanged information

**Synchronization of information** : Choreographies allow information share between their participants

# WS-CDL - features

- **Exceptions treatment** : Can define how to treat the exceptions
- **Transactions**: Allow transactional work and exception control during a long execution
- **Pattern compositions** : Allow complementary work with other specifications like WS-Reliability, WSCAF, WS-Security, WS-BPEL and many others
- **Semantics** : Choreographies must have semantic description for all of its participants.

**WS-CDL**

# WS-CDL Usage Picture

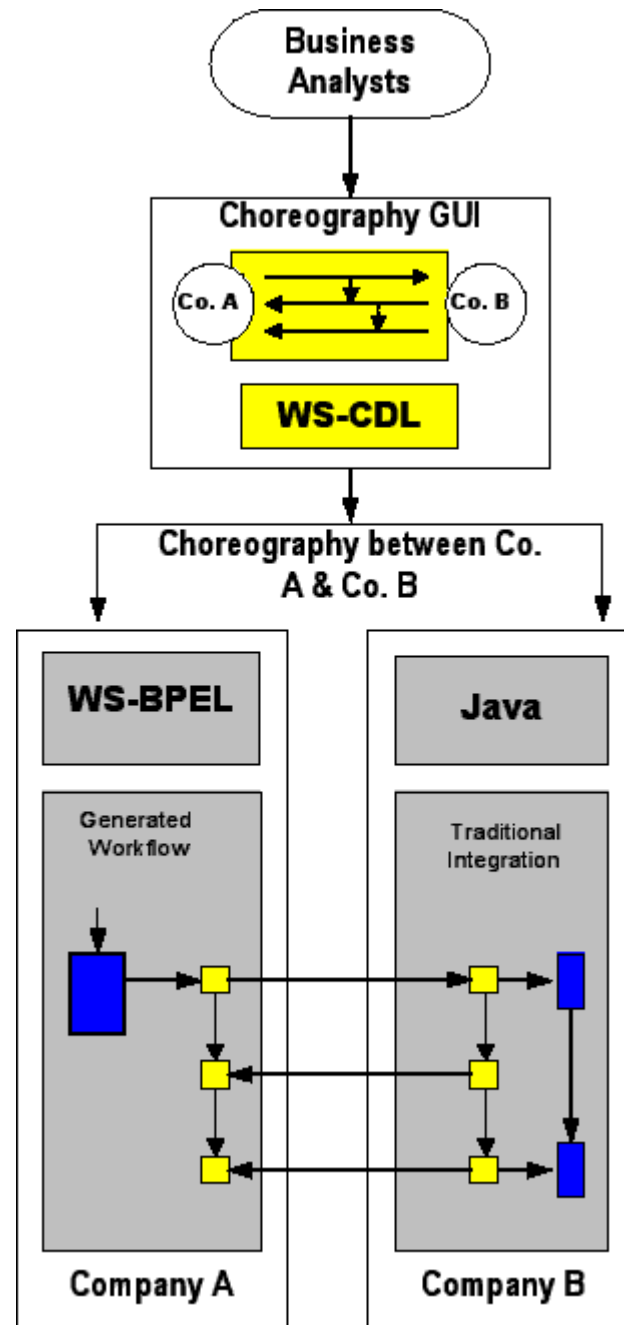


Fig. : Integrating Web Services-based applications using WS-CDL

# Why Process Calculus?

Model	Completeness	Compositionality	Parallelism	Resources
Turing Machines	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Lambda	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Petri Nets	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
CCS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
$\pi$	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Table: pi-calculus characteristics

# WS-CDL and pi-calculus

Operation	Notation	Meaning
Prefix	$\pi.P$	Sequence
Action	$a(y), a(y)$	Communication
Summation	$a(y).P + b(x).Q$ $\sum \pi_i(P_i)$	Choice
Recursion	$P=\{\dots\}.P$	Repetition
Replication	$!P$	Repetition
Composition	$P   Q$	Concurrency
Restriction	$(vx)P$	Encapsulation

Collapse send and receive into an interact on channels

Table: ws-cdl and pi-calculus

# WS-CDL and the pi-calculus

- Static checking for livelock, deadlock and leaks
  - Session types and causality
- Robust behavioral type system
  - Session types

# WS-CDL Package

- Package root element
- Sets target namespace
- Allows importing
- Contains all further definitions
- Especially one or more choreographies

```
<package  
  name="ncname"  
  author="xsd:string"?  
  version="xsd:string"  
  targetNamespace="uri"  
  xmlns="http://www.w3.org/2004/04/  
  ws-chor/cdl">
```

```
  importDefinitions*  
  informationType*  
  token*  
  tokenLocator*  
  role*  
  relationship*  
  participant*  
  channelType*  
  Choreography-Notation*  
</package>
```

exchanged Information

Collaborating parties

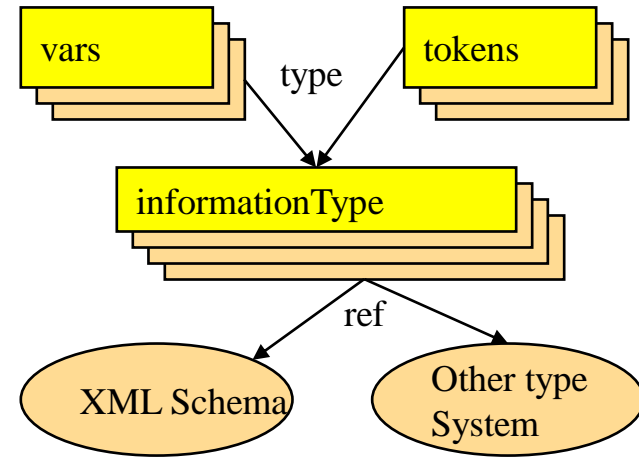
Way for exchange

Interactions between parties

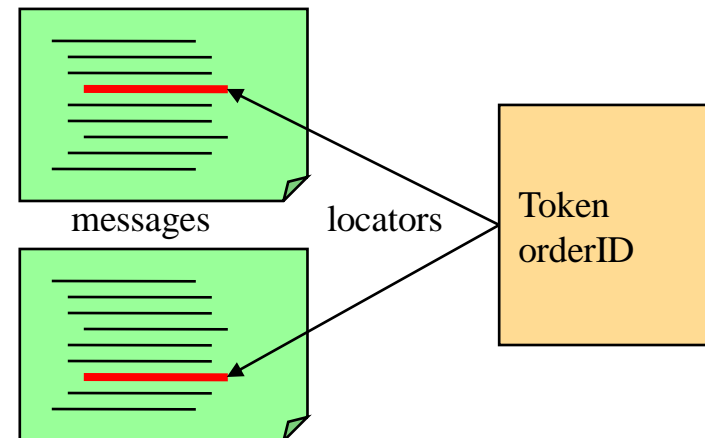


# Data Concepts

- Information type
- Independence of schema



- Token, token locators
- Accessing same information in diverse sources



# Data Concepts

- **Channel**
  - For communication with a participant
  - Dynamic, actual endpoint in data
  - Can be passed around
- **Variables**
  - Information exchange, state, channel variables
  - Reside in Roles

# Participants & Roles

- **Roles**

- Enumerate behaviors, optionally linked to WSDL interfaces

- **Participants**

- Play one or more roles
  - Apparently not used in WS-CDL

- **Relationships**

- Associate specific behaviors of two roles “for a purpose”
  - Complex relationships broken down to pairs

# Participants & Roles

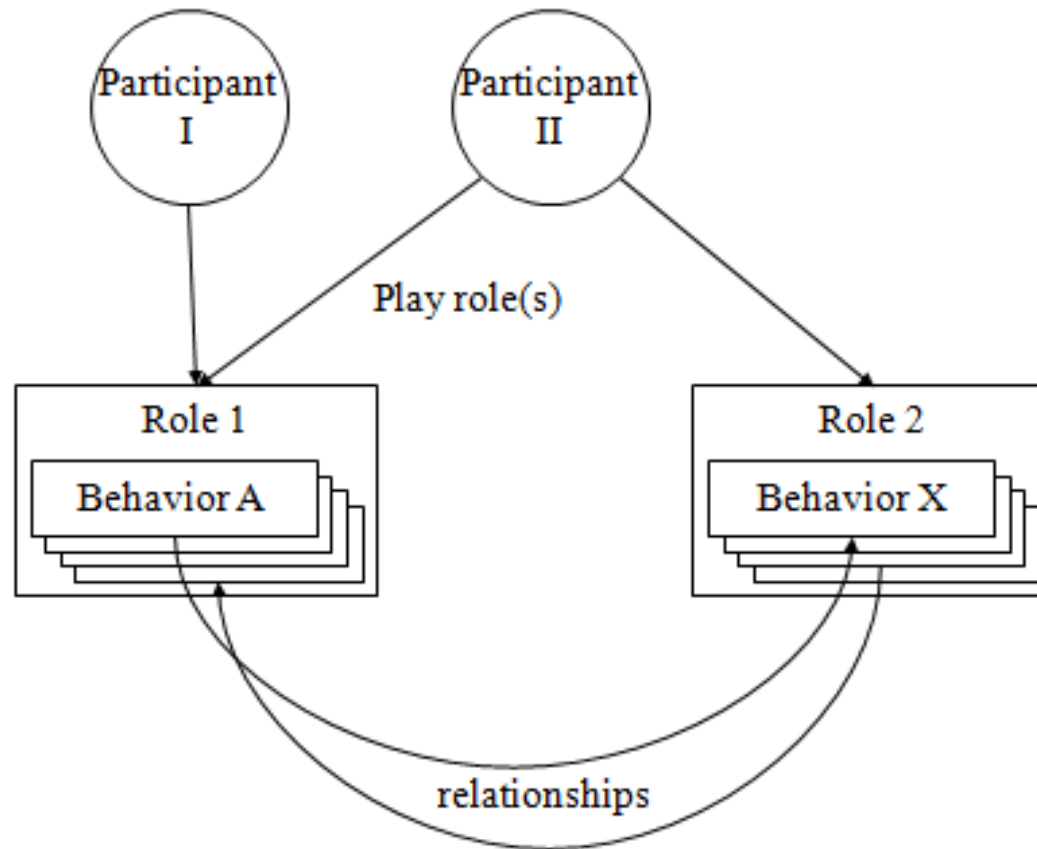


Fig. : Participants and Roles

# Choreography Syntax

```
<choreography name="ConsumerRetailerChoreo" root="true">  
  <relationship ... /> *  
  <variableDefinitions>  
    <variable ... /> *  
  </variableDefinitions> ?  
  ... <!-- local choreographies -->  
  
  <interaction initiateChoreography="true">  
    ...  
  </interaction>  
  <exception> ... </exception> ?  
  <finalizer> ... </finalizer> ?  
</choreography>
```

# Activities

- Perform actual work
- **Ordering structures** (complex activities)
  - Sequence, parallel
  - Choice – implicit selection by incoming message
- **WorkUnits** (guarded activities)
  - Condition and repetition for activity
  - Optionally blocking on data availability
- **Variable assignment**  
Create or update a variable in a role
- **NoAction**  
non-observable effects

# WorkUnit Example

```
<workunit name="POProcess"  
    guard="cdl:getVariable(  
        `POAcknowledgement` ,  
        `tns:customer` )"   
    block="true">  
  
    ... <!--some PO processing activity -->  
</workunit>
```

# Activities

- **Interaction**
  - Roles within a relationship communicate/align information and state
  - Time-to-complete timeout mechanism
  - Can initiate choreography
- **Performing a choreography**
  - Choreography composition
  - Including variable and role aliasing



# Interaction Example

```
<interaction channelVariable="tns:retailer-channel"
  operation="handlePurchaseOrder" align="true">
  <participate relationship="tns:ConsumerRetailerRelationship"
    fromRole="tns:Consumer" toRole="tns:Retailer" />

  <exchange messageContentType="tns:purchaseOrderType"
    action="request">
    <use variable="cdl:getVariable(tns:purchaseOrder, tns:Consumer)"/>
    <populate variable="cdl:getVariable(tns:purchaseOrder,
      tns:Retailer)"/>
  </exchange>
  <exchange messageContentType="purchaseOrderAckType"
    action="respond">
    ...
  </exchange>

  <record role="tns:Retailer" action="request">
    <source variable="cdl:getVariable(tns:purchaseOrder, PO/CustomerRef,
      tns:Retailer)"/>
    <target variable="cdl:getVariable(tns:consumer-channel,
      tns:Retailer)"/>
  </record>
</interaction>
```

# Perform Syntax

```
<perform choreographyName="qname">  
  <alias name="ncname">  
    <this    variable="XPath-expression"  
      | role="qname" />  
    <free    variable="XPath-expression"  
      | role="qname" />  
  </alias>  
</perform>
```

# Issues, Challenges and Trends



# Service orchestration

- Service interaction at message-level
- Point to point compositions from perspective & control of a single endpoint.
- Executable business processes

# Service Choreography

In maturity process, actually: WS-CDL,  
BPEL4Chor (academic)

← → ↻ 🏠 ☆ <http://www.w3.org/2002/ws/chor/> ▶



[About Web services](#) · [Web Services Activity s](#)  
[Administrative page](#) · [Web Ser](#)

## Web Services Choreography Working Group

[Charter](#) · [Drafts](#) · [Meeting records](#) · [Discussion lists](#) · [Participants \(IPR declarations\)](#) · [implementations](#) · [Related resources](#)

The Working Group was closed on the 10th July 2009.

### Discussion lists

Technical discussions take place on the public [public-ws-chor@w3.org](mailto:public-ws-chor@w3.org) mailing list ([public archive](#)).

**Leaving WS-CDL as a Candidate Recommendation**

# Choreography Challenges

- Composability analysis for repleceability, compatibility & conformance.
- Autonomic composition of service.
- QoS aware service composition.
- Business-driven composition.
- Composition of resources, human & organizations in the form of services.
- Highlight and improve the semantic characteristic that are present in the current standards

# QoS utilization

- Integrating QoS into choreography is valuable for designing and analyzing the choreography.
- Calculate the overall quality of a composed system, give QoS information of its individual elements.
- Another is how to choose some services from a group of candidates to implement an abstract composition while ensuring some QoS criteria.



# Validation and verification

- Formal approaches are useful in analyzing and verifying web service properties:
  - There is a model-based approach to verify web service compositions.
- Prevent deadlocks, live-locks, functional requirements (e.g. the purchase order will be either accepted or rejected, but not both accepted and rejected).

# Validation and verification

- Many investigations in this issue, create models for a CDL-subset.
- Model-checking technique to verify the correctness of specified systems.
- Given a system, we might check its consistency, and
- Various properties (e.g. no deadlock), and the satisfaction with business constraints.


# Development and Tools


# Available Tools



## WS-CDL Implementations

This list is not a complete list of all the implementation around.

	Name	Platforms	Last Update	Availability	Screenshots	Notes
	Pi4SOA CDL Editor	<a href="#">Eclipse</a>		<a href="#">Free</a>		
	WS-CDL Eclipse	<a href="#">Eclipse</a>	2005-05-19	<a href="#">Free</a>		
	LTSA WS-Engineer	<a href="#">Eclipse</a>	2005-11-30	<a href="#">Free</a>		

 **BPEL4CHOR – editor (Tools4BPEL)**

# Outdated Tools

- **LTSA WS-Engineer Eclipse Plug-in.**
  - Broken link of the install site for LTSA Eclipse plug-in.
  - Support for WS-CDL 1.0
- **WS-CDL Eclipse:**
  - Hosted on Sourceforge
  - Support for WS-CDL 1.0
  - Discontinued 5 years ago

# PI Calculus for SOA

Pi4 Technologies Foundation open source tool suite:

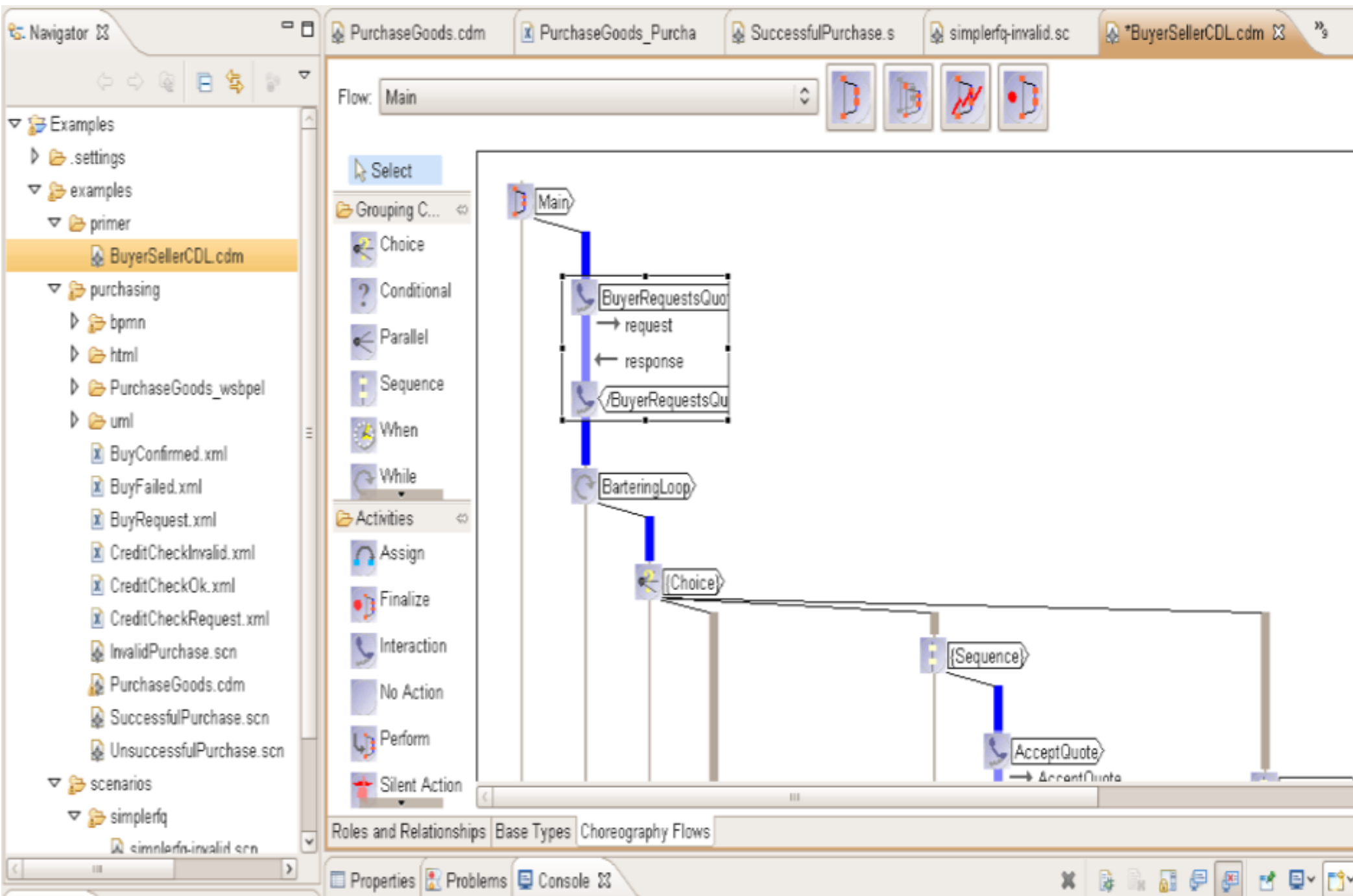
[www.pi4tech.org](http://www.pi4tech.org)

Eclipse plugins

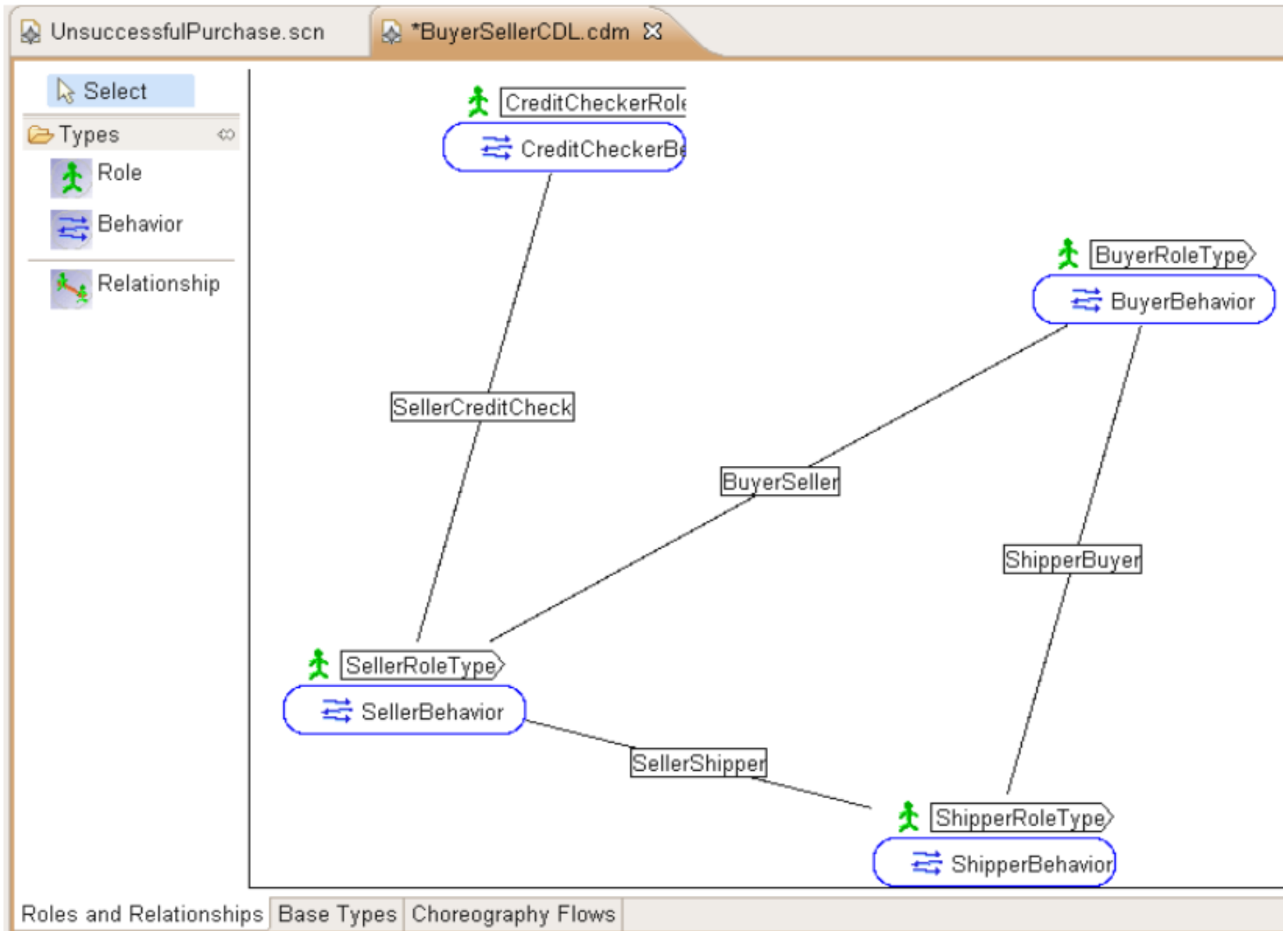
Graphical modeling environment

Static type checking (completeness, lock free, race condition free)

- **Export**
  - WS-CDL
  - WSDL
  - HTML
  - BPMN
- **Generate**
  - UML artifacts
  - Java (for J2EE)
  - Java (for Axis)
  - BPEL
- **Monitor**
  - Against WS-CDL description
  - Legacy and/or generated

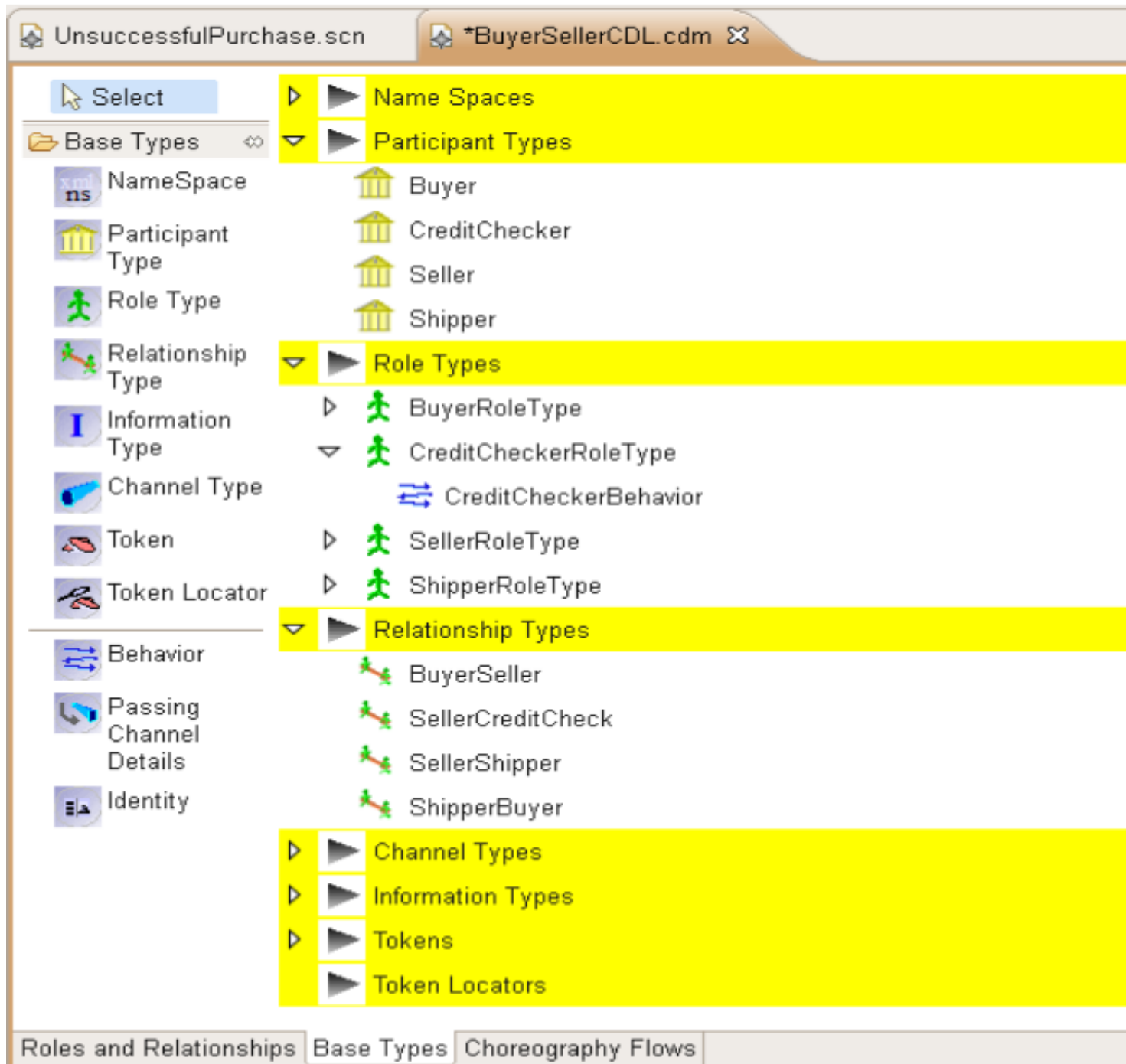


**Fig. : Overview of pi4SOA plugin**



**Fig. : Roles and Relationships**





**Fig. : Base Types**

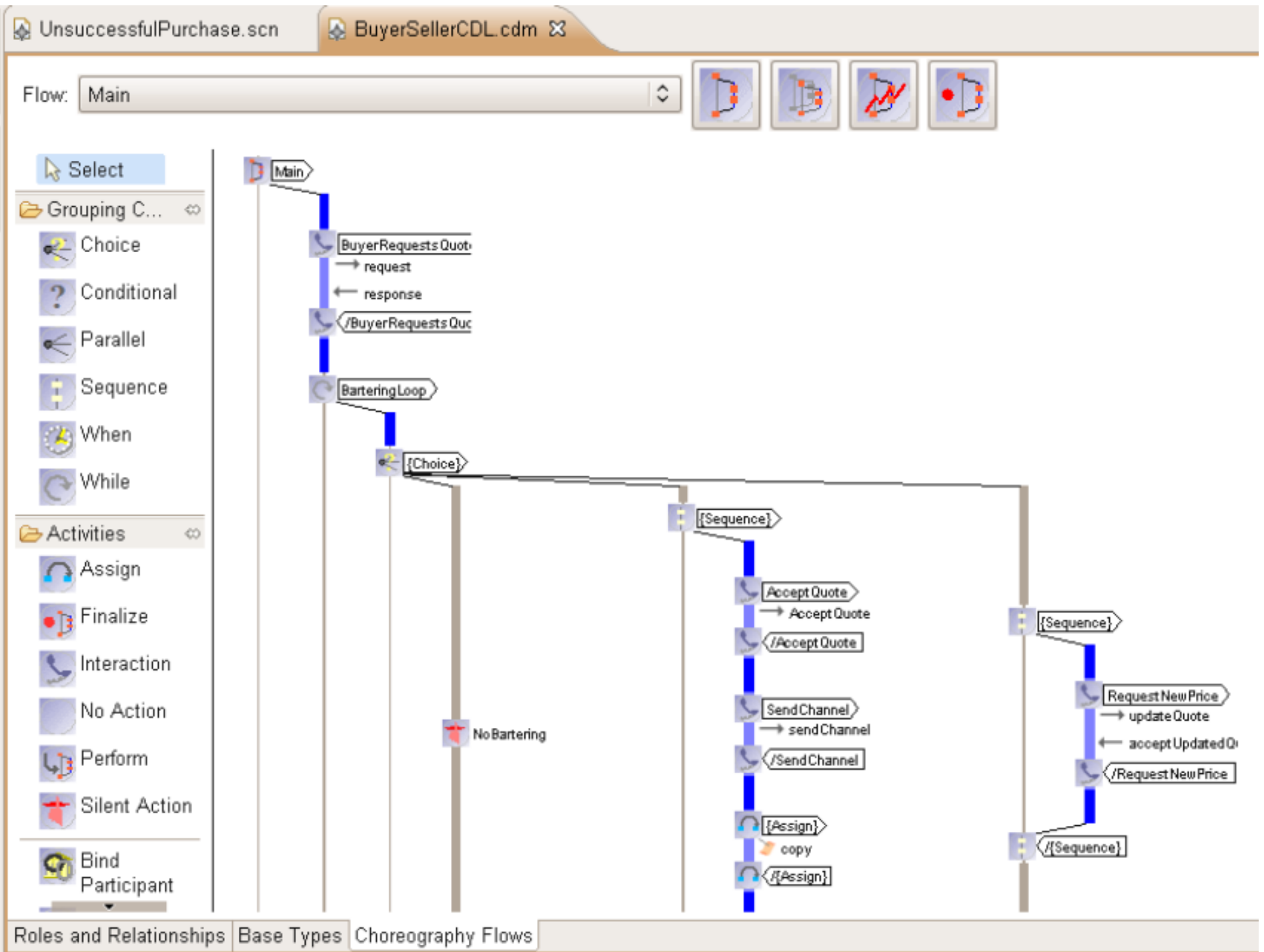
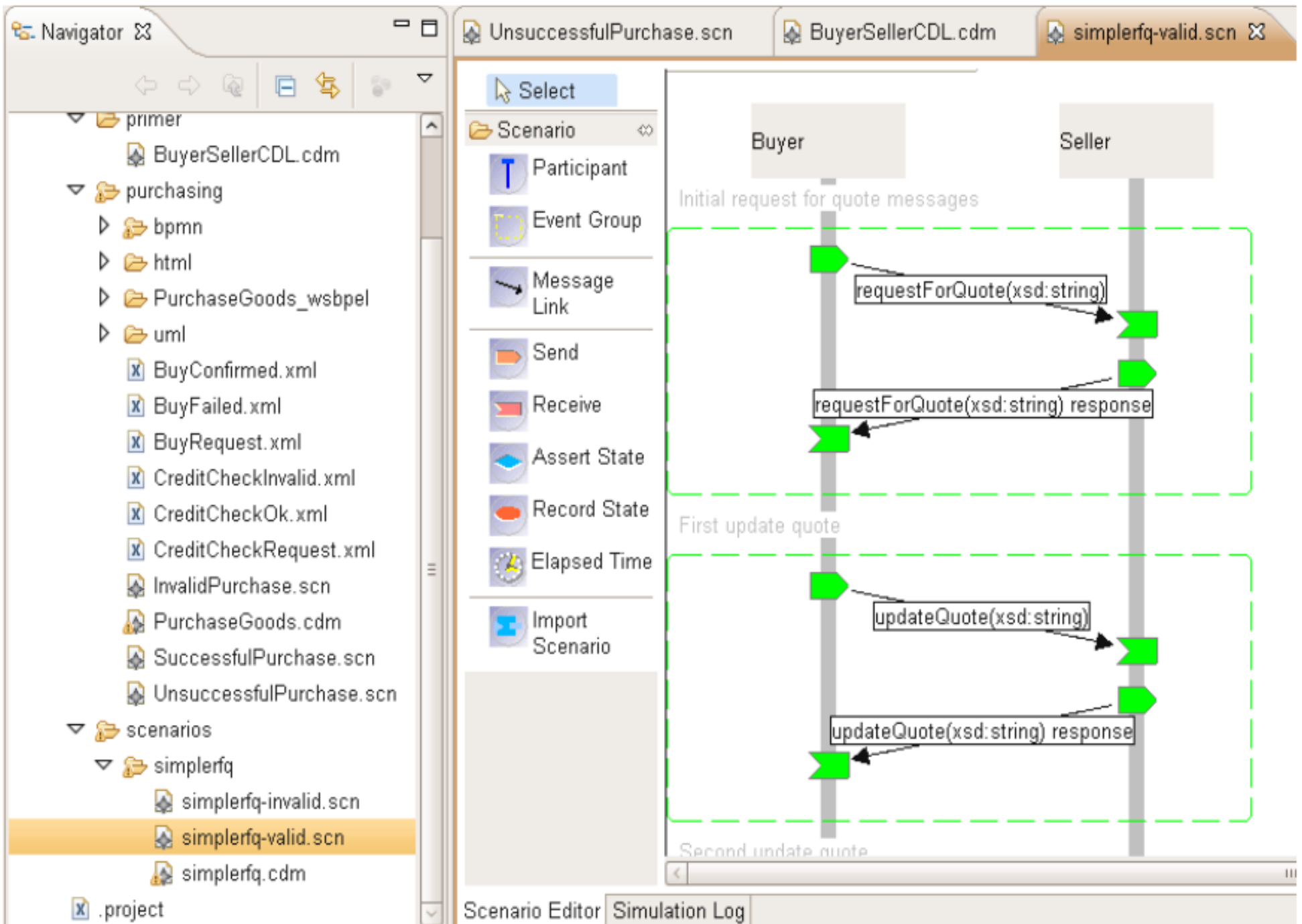


Fig. : Choreography flows



**Fig. : Scenario Editor**

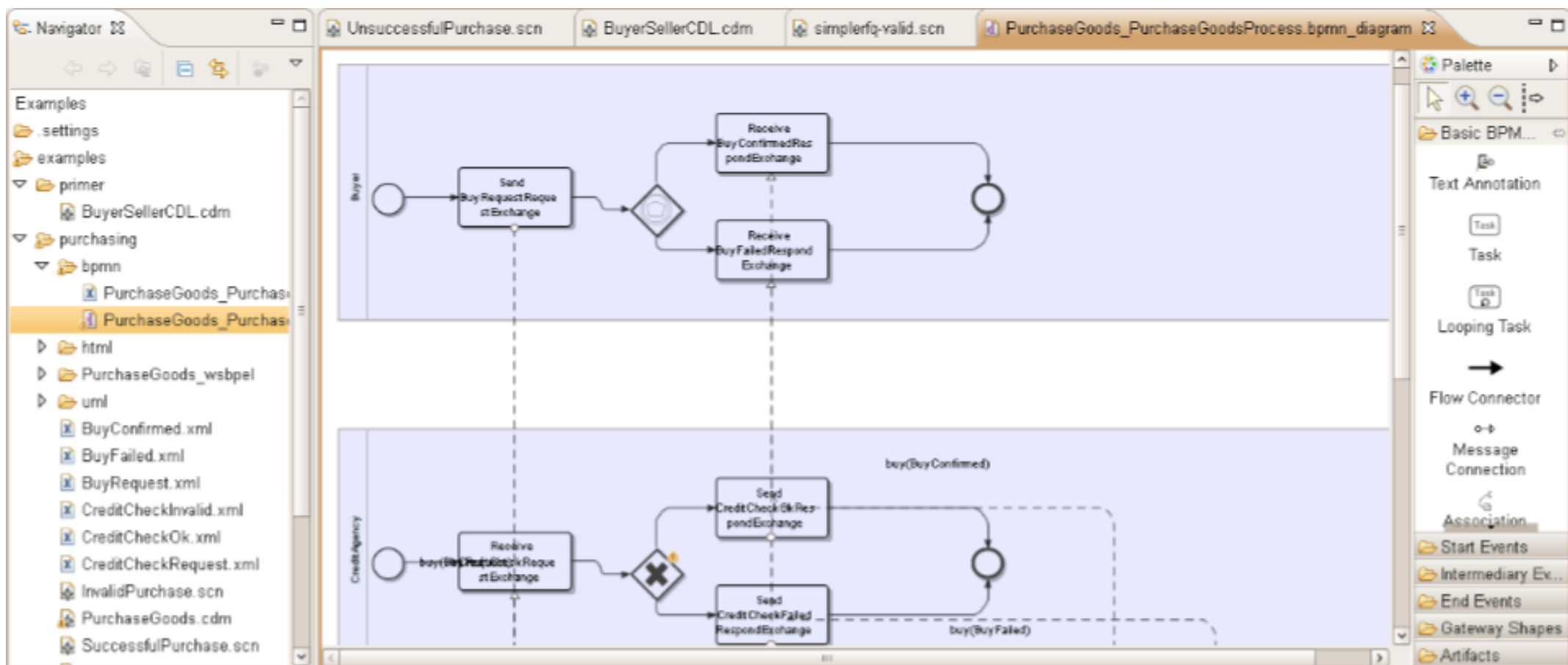


Fig.: Export to BPMN model

# SOA in Numbers

“SOA markets at \$3.5 billion in 2009 are anticipated to reach 8.2 billion by 2016. Market growth is a result of demand for automated business process that permits flexibility in response to changing business conditions. SOA provides this as application middleware that permits IT to manage change.”

“IBM is the market leader in SOA application middleware with 75% market share.”

Some enterprises such as: IBM, Microsoft, Google, Oracle, Red Hat, Hewlett Packard (HP), Computer Associates (CA), have interests in SOA

# Conclusions

- The goal of the WS-CDL language is to propose a declarative XML based language that concerns about global, multi-party, peer-to-peer collaborations in the web services area.
- Although WS-CDL borrow terminologies from Pi-Calculus, the link to this or another formalism is not clearly established.
- Design of web services appears to have a new approach comparing with traditional software development.

# Bibliography

Barker, A. , Walton C.D., Robertson D. "*Choreographing Web Services*". Services Computing, IEEE Transactions on , vol.2, no.2, pp.152-166, April-June 2009.

Cambroner, M.E.; Diaz, G. Martinez, E. Valero, V. , "*A Comparative Study between WSCI, WS-CDL, and OWL-S*" e-Business Engineering, 2009. ICEBE '09. IEEE International Conference on , vol., no., pp.377-382, 21-23 Oct. 2009.

Alistair B. , Marlon D., Philipa O. "*A Critical Overview of the Web Services Choreography Description Language (WS-CDL)*". BPTrends, March 2005.

Stephen Ross T. "*Orchestration and Choreography: Standards, Tools and Technologies for Distributed Workflows*". Pi4 Technology, London, UK and W3C, Geneva, Switzerland.

# Bibliography

Paulo Henrique M. "*Coreografia de Serviços WEB (Uma abordagem para a integração de serviços Web)*". Bachelor's work of Universidade Federal de Santa Catarina, June 2007.

Adam Barker, Christopher D. Walton, and David Robertson. "*Choreographing Web Services*". IEEE Transactions on Services Computing, Vol. 2, N° 2, April-June 2009.

Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, Frank Leymann, "*Service-Oriented Computing: State of the Art and Research Challenges*". *Computer*, vol. 40, no. 11, pp. 38-45, Oct. 2007, doi:10.1109/MC.2007.400.



**Any questions ?**