# Web Services and Business Processes

## Infrastructure for agile SW deployment

M.I. Capel

ETS Ingenierías Informática y
Telecomunicación
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Granada
Email: manuelcapel@ugr.es
http://lsi.ugr.es/mcapel/

October, 30th 2020

Máster Universitario en Ingeniería Informática

1. Introduction to Middleware
2. Introduction to SOA
   - SOAP
   - XML
   - UDDI
   - SW Programming
3. Business process modeling
   - Business Processes characteristics
   - Business processes with UML
   - Initial Business View
   - Business Structure View
4. Web services composition
   - Orchestration
   - Choreography

# Middleware characteristics

## Definition of Middleware

software that offers the capability of integration and reuse of components at different scale (methods, protocols, software and systems) on demand, aimed to abstract the distribution and heterogeneity of the available services to the underlying computing environment

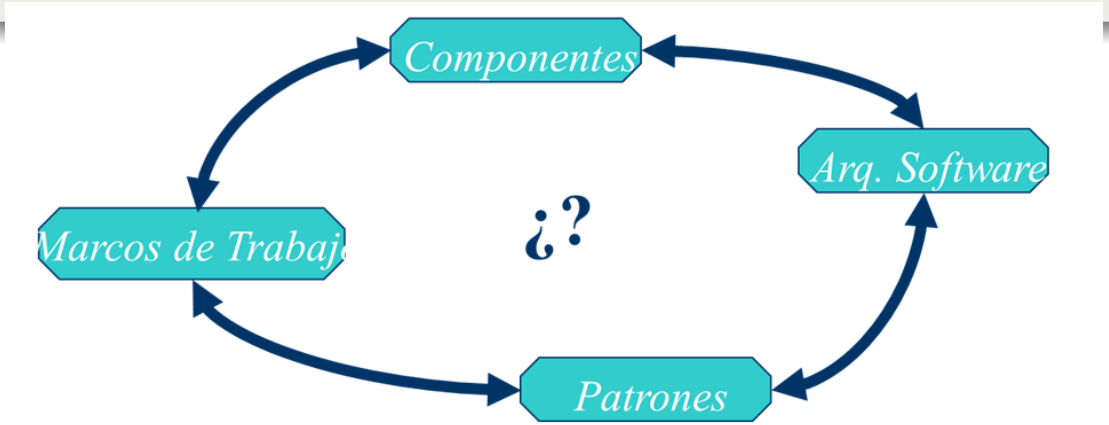



Figure: Middleware is something that's referred to by software developers as "software glue"

# Middleware characteristics-II

## Middleware objectives

To support the development, deployment, execution and maintenance of software applications, by saving the distance between the application layer itself and the underlying layers (operating system, network stack and hardware)

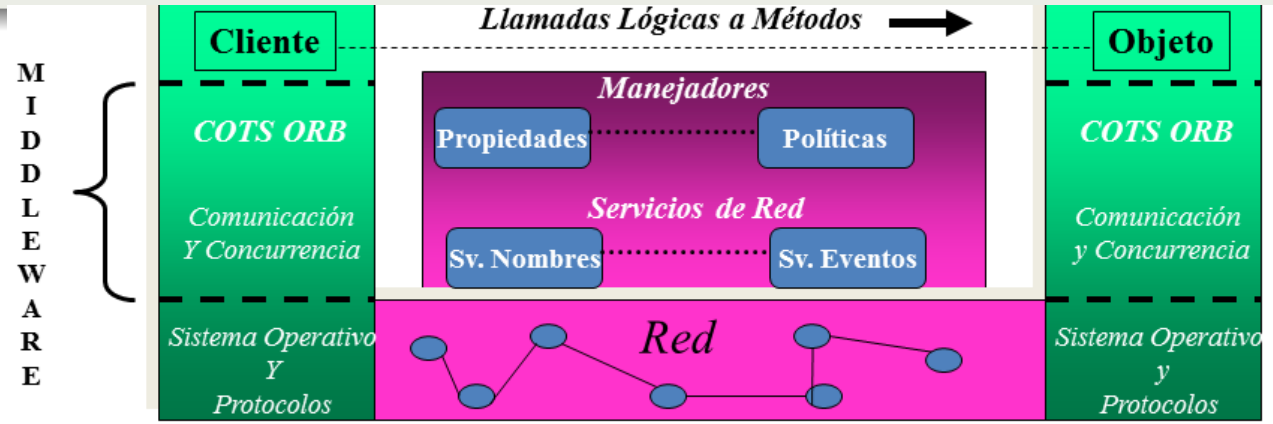



Figure: Resides between the application, operating systems, protocols and underlying hardware

# Middleware characteristics-III

## Middleware example

The functionality related to gesture or voice recognition is normally processed by a specific intermediary software and the results of that processing are transmitted to user applications

# Support technologies for component-based software and Systems development

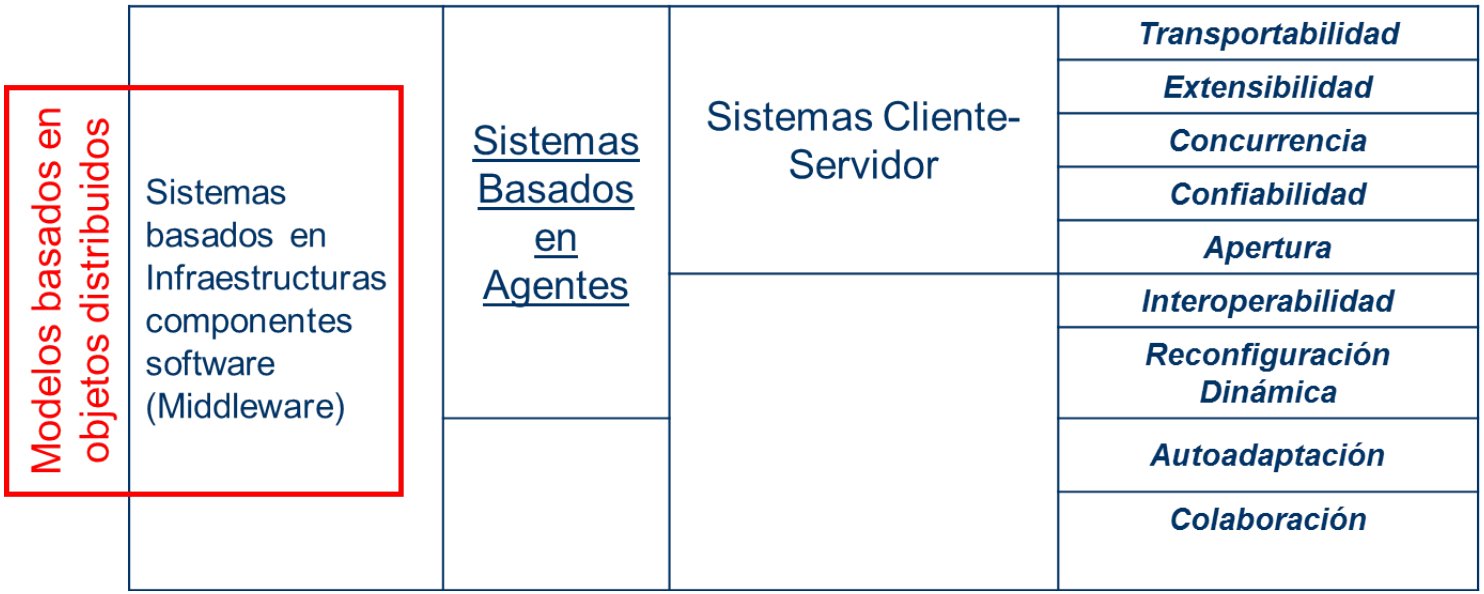



Figure: Covering degree of the required properties

# Middleware shortcomings

## Disavantages

- Loss of speed
  - Slow computer for application server
  - slow connection between applications server and database server
- Security
  - Can introduce new security holes
- Reliability
  - Have to worry about an extra computer going down

# Middleware shortcomings-II

## High requirements, difficult to fulfill by applications with time-criticality

- Any middleware-based solution must provide a runtime environment that is responsible for coordinating and supporting multiple applications capable of adapting to specific computing needs
- Design and development of middleware software that is successful for certain types of applications (real time, sensor networks, etc.) is far from being easy
- In addition to responding to the challenges of these applications, it must respond to the demands imposed by the execution context (punctuality, real-time restrictions,...)

# Middleware-based solutions “obsolescence”

## Fundamental idea

Currently, the development of applications for specific domains has evolved towards paradigms that involve **encapsulating more knowledge and independence** of processing and interoperability **in the own software components** rather than in middleware, the paradigms called “ **Service Oriented Architecture** ” (SOA) and Agent Oriented Programming arouse more interest today

# SOA characteristics

## What's SOA?

A standard and flexible software architecture that unifies business processes by structuring big applications as a collection of “services"

These applications can be used by end users inside or outside the company

## Architectural style

SOA is also an architectural style where all the functionalities, whether they are already at stake or are new, get grouped as services that communicate between themselves

# Web Services

**WS**

are services implemented using a set of standards: SOAP, WSDL and UDDI, so that they can be accessed through a network or the Internet, and get executed remotely

**Standard Protocol**

Service Oriented Architecture Protocol (SOAP)

A common protocol to all the architecture's components

**Services description language**

Web Service Description Language (WSDL)

**Discovery of available services**

Universal Description Discovery and Integration (UDDI)

Services get classified into categories, according to the offering and how can be called

# Fundamental


- We can talk about the first and the second generation of Web Services (WS)
- *Primitive* SOA corresponds to 1st Generation WS
- *Contemporary* SOA corresponds to 2nd Generation (WS 2.0)

# WS implemeentation framework

## Characterisation of *Base*-Web Services

- have to be *abstract*:
  - Defined by an international organisation of standards (ISO, ...)
  - implemented on all platforms
- Constructive blocks, service and message descriptions based on WSDL
- Message exchanging component based on SOAP and its concepts
- UDDI-based service description registry, which sometimes includes service discovery
- A well-defined software architecture with messages,

# Implementation of SW

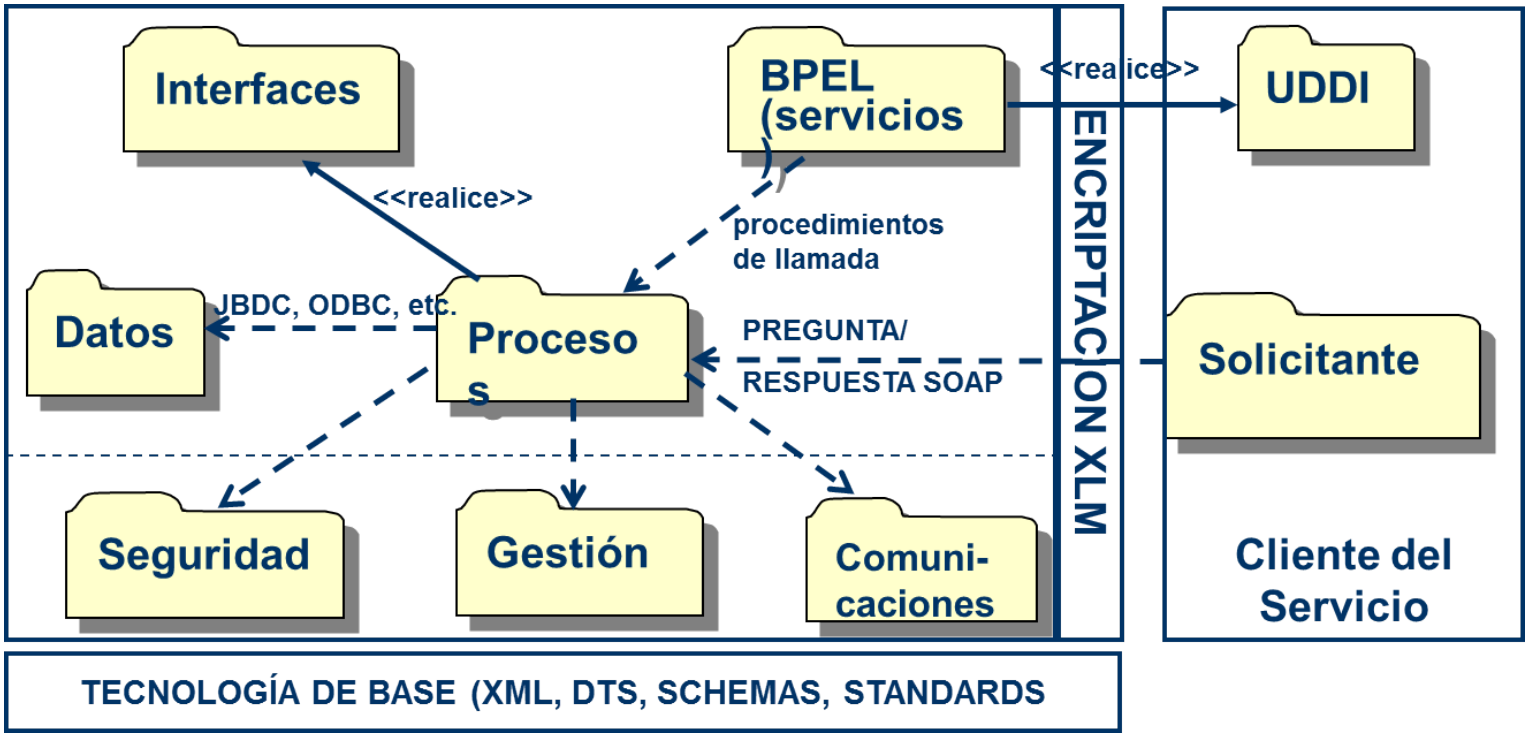
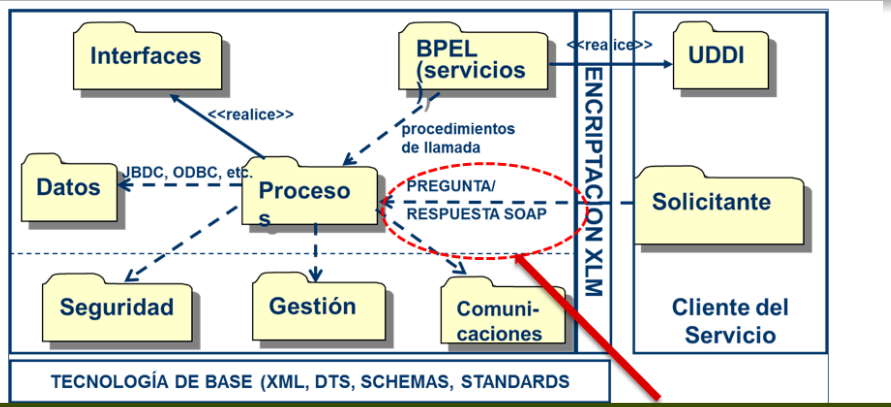



Figure: Implementation of a WS architecture

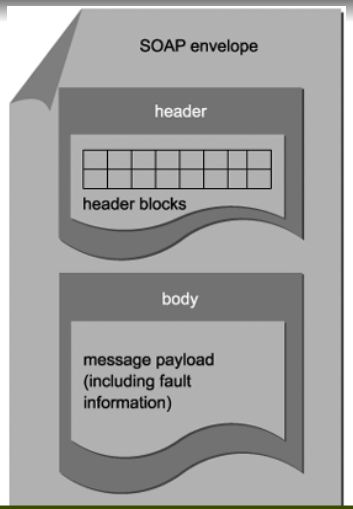
# SW and SOAP




## Message passing vs. RPCs

- Problems that SOA implementation with RPCs/RMI shows
- RPC realisation based on XML documents (as autodescriptive messages)
- XML–RPC
- SOAP –successor of XML-RPC– allows remote objects calls with HTTP

# SW and SOAP II




## What SOAP gives us:

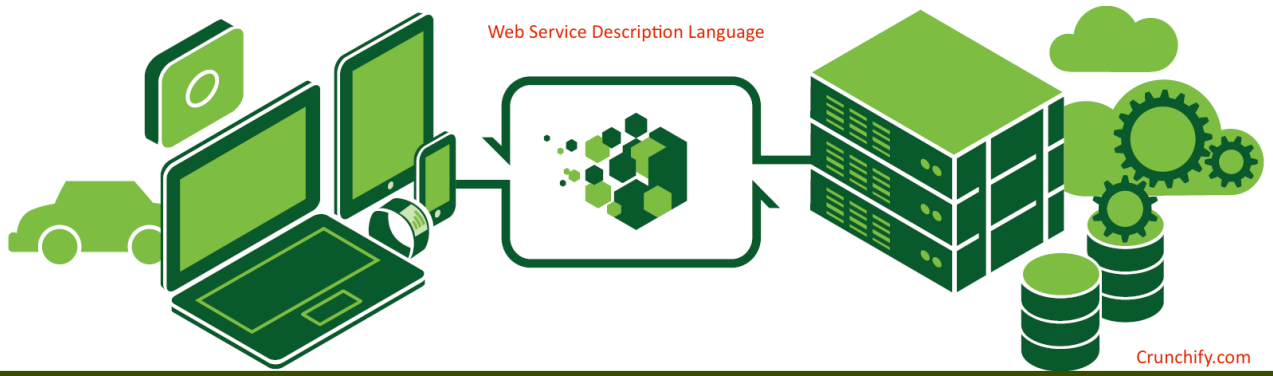
- To describe the (*"target"*) of a remote call
- Interoperable codification of data types within the 'remote-call' message
- To define relevant parts of each message
- Protocol independence (bindings instrumented with HTTP)

# SOAP characteristics

## Elements of the standard

- Namespaces for *wrappers*: `http:\schemas.xmlsoap.org/soap/envelope/`
- Namespaces for message contents codifying: `http:\schemas.xmlsoap.org/soap/encoding`
- Object references are allowed but only for objects that communicate through serialized communications (names used in messages are URLs)
- Passing of object references to a SOAP-receiver adapter
- Authentication of parts, information encryption, and compensation are not SOAP responsibilities

# SW and XML




## WS

- Services are normally offered by a Web server
  - “XML-capable" servers have the greatest interest
- Then, Service Level Agreements (“SLA”) are guaranteed

# XML Web Services

## Similarity with *software components*

- The client part has the same perception as if WS were software-components
- WS –'provided interfaces' are equivalent to the ones of software components

## Relevant differences with software components

- *Required interfaces* will be never disclosed by a WS
- Required interfaces can only be used in the context in which they are offered, since WS are 'self-contained'
- Internet weak-coupling is always assummed between WS
- Interface binding-based reconfiguration banned

# SW Descriptions

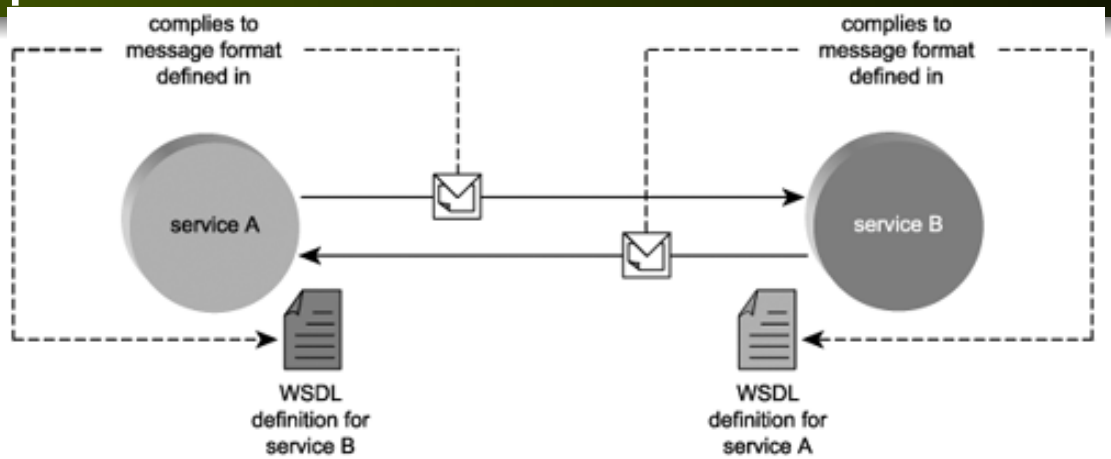
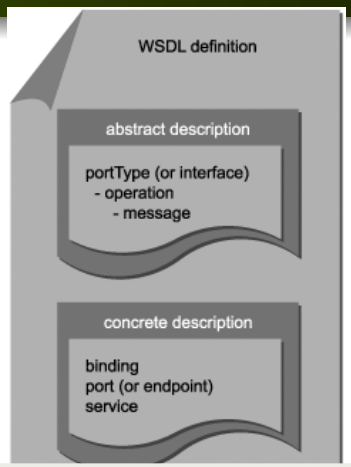



Figure: Low coupling between services by using WSDL

## Similarity with component descriptions:

- Software component's interfaces need metadata or a special annotation for their semi-formal description
- "*Web Services Description Language (WSDL)*" is the appropriate standard to solve the above issue
- Extensible framework definition

# SW Descriptions with WSDL – II




- Service descriptions are fundamental for obtaining weak-coupled communication
- and also for getting a development infrastructure of software based on WS

- Abstract description
- Concrete description

# Abstract description of services with WSDL

Sets the characteristics of the service interface without any reference to its implementation technology

## Abstract description structure

- *PortType* (WSDL 2.0 “Interface”): message ordering in groups of operations
- *Operation*: set of input/output messages ($\equiv$ operation’s parameters)
- *Message* patterns that govern transmission of message sequences

# Concrete description of services with WSDL

## Objective:

Connects an abstract interface definition with a real technology that can be used for service implementation, i.e., for deployment into a given physical network's transport protocol environment

## Concrete description structure

- *Binding*: sets the conditions needed to establish the "*physical*" connections of the service with the solicitors
- *Port*: "physical" direction through which any service can be accessed by using a specific protocol renaming, similar to the *end-point* concept in WSDL 2.0
- *Service*: groups of *end-points* that are the *correspondents* for providing a service, which is demanded at the other end

# Example: abstract WSDL description of a simple WS

```
<definitions name="StockQuote" targetNamespace="http://example.
    com/stockquote.wsdl"xmlns:tns="http://example.com/
    stockquote.wsdl"xmlns:xsd1="http://example.com/stockquote.
    xsd"xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"xmlns
    ="http://schemas.xmlsoap.org/wsdl/">
  <types>
   <schema targetNamespace="http://example.com/stockquote.xsd"
           xmlns="http://www.w3.org/2000/10/XMLSchema">
          <element name="TradePriceRequest">
           <complexType>
<all><element name="tickerSymbol" type="string"/></all>
           </complexType>
          </element>
          <element name="TradePrice">
           <complexType>
            <all>
             <element name="price" type="float"/>
            </all>
           </complexType>
          </element>
   </schema>
```

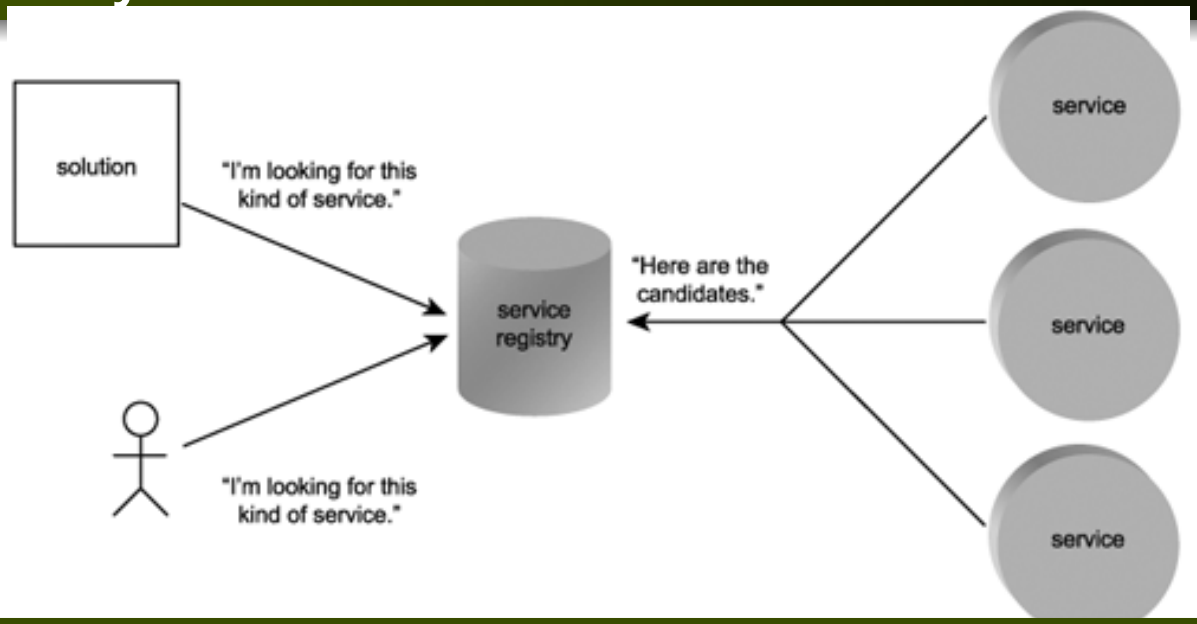
# Example: abstract WSDL description of a simple WS –II

```
1  ...
2    <message name="GetLastTradePriceInput">
3       <part name="body" element="xsd1:TradePriceRequest"/></
            message>
4    <message name="GetLastTradePriceOutput">
5       <part name="body" element="xsd1:TradePrice"/></message>
6    <portType name="StockQuotePortType">
7       <operation name="GetLastTradePrice">
8         <input message="tns:GetLastTradePriceInput"/>
9         <output message="tns:GetLastTradePriceOutput"/>
10      </operation>
11   </portType>
```

# Example: concrete WSDL description of a simple WS

```
1   <binding name="StockQuoteSoapBinding" type="tns:
        StockQuotePortType">
2       <soap:binding style="document" transport="http://schemas.
            xmlsoap.org/soap/http"/>
3       <operation name="GetLastTradePrice">
4           <soap:operation soapAction="http://example.com/
                GetLastTradePrice"/>
5           <input><soap:body use="literal"/></input>
6           <output><soap:body use="literal"/></output>
7       </operation>
8   </binding>
9   <service name="StockQuoteService">
10      <documentation>My first service</documentation>
11      <port name="StockQuotePort" binding="tns:
            StockQuoteBinding">
12          <soap:address location="http://example.com/stockquote"
                />
13      </port>
14  </service>
15 </definitions>
```

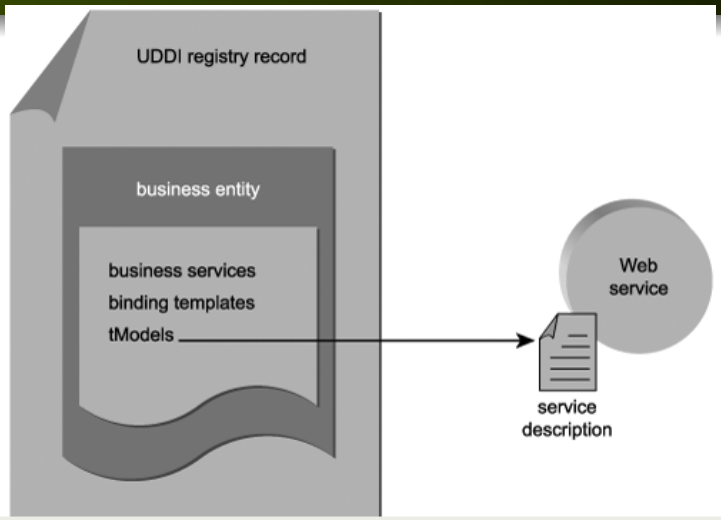
# SW discovery




## WS discovery mechanism is a “must"

Any Web site that implements a WS does not need any type of support to be discovered by applications, but an implementation of a WS -by any other means- needs a *remote directory* in order to be found

# SW discovery-II




- To be able to access to any SW, the only requirement is to have access to the WS description
- Service repositories are needed,
  - To locate the last versions of available WS
  - To discover new WS according to a set of objectives of a given application
  - UDDI is available since the 1st generation of WS came out

# WS registry with UDDI

## Registry contents:

1. Business entity
2. Business Services
3. Business input service skeletons
4. `tModels`, i.e., the 'technical specification' of each type of service

- UDDI keeps a global registry of `tModels`, identified by their *universal IDs*
- A UDDI service is also a regular WS, so each query to UDDI is a XML message wrapped by a SOAP one

# Example of UDDI code

```
POST /safe_business HTTP/1.1
Host: www.XYZ.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "safe_business"
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas/xmlsoap.org/soap/envelope/">
   <Body>
     <safe_business generic="2.0" xmlns="urn:uddi-org:api_v2">
     <businessKey="">
     </businessKey>
       <name>
       Enagas, S.L.
      </name>
```

# Example of UDDI code UDDI –II

```
1        <description>
2           The company is interested in providing state-of-the-art
                in ....
3        </description>
4        <identifierBag> ... </identifierBag>
5           ...
6        </safe_business>
7     </Body>
8  </Envelope>
```

# WS protocol's stack

| | | |
|---|---|---|
| Discovery | UDDI | It's also a Web Service and works as one Web Services directory |
| Description | WSDL, WSFL/XLANG other for showing up | Given 1 or more WS, these notations describe the properties in a meta-level |
| Access | SOAP, SOAP with attachments | Given a Web Service instance, its access is based on messages |
| Transfer | HTTP,SMTP, other | Transfer SOAP messages, attachments included |
| Transport | TCP/IP, UDP, other | Transport data |

Table: Fundamental WS-protocol stack

# WS and programming models

- WSDL ports vs. component interfaces methods
- OOP model incompatibility
- Asynchronicity ($\neq$ RPC-based mechanisms)
- Round-trip call latency is unpredictable for WS requests
- Unpredictable delays caused by message storage in queues
- Concurrency constraints (caused by a great number of threads) in the client side
- Deployment of an asynchronous model for WS requests, similar to the one based on message queues

# WS-I

## Web Services Interoperability Interface

- Industrial consortium initiated for promoting *interoperability* amongst the stack of WS specifications
- "*Guidelines*" definition isn't any new standard
- Recently, became part of OASIS
- http://ws-i.org/Profiles/BasicProfile-1.2-2010-11-09.html
- *Self-certification* guidelines inclusion

# WS-I

## Web Services Interoperability Interface

WS-I
WEB SERVICES INTEROPERABILITY ORGANIZATION

### Basic Profile Version 1.2

**Final Material**

**2010-11-09**

*This version:*
http://www.ws-i.org/Profiles/BasicProfile-1.2-2010-11-09.html

*Latest version:*
http://www.ws-i.org/Profiles/BasicProfile-1.2.html

*Errata for this Version:*
http://www.ws-i.org/Profiles/BasicProfile-1.2-errata-2010-11-09.html

*Editors:*
Robert Chumbley , IBM
Jacques Durand , Fujitsu
Micah Hainline , Asynchrony Solutions
Gilbert Pilz , Oracle
Tom Rutt , Fujitsu

*Administrative contact:*
secretary@ws-i.org



**Abstract**

This document defines the WS-I Basic Profile 1.2, consisting of a set of non-proprietary Web services specifications, along with clarifications, refinements, interpretations and amplifications of those specifications which promote interoperability. It also conta
set of executable test assertions for assessing the conformance to the profile.

**Status of this Document**

# WS-II

## Web Services Interoperability Interface

### Table of Contents

# SW and software components models standards

## Current proposals in the software components market

- CORBA, is in the market for a long time: business-oriented computation and legacy applications
- Java (prior Sun Systems) of Oracle: the most important language/platform for application and Web servers development
- COM+ and .NET of Microsoft: takes advantage of the quasi-“monopolistic” Windows operating system within the low-cost laptop market
- The new models of WS have become a new software layer, which are currently essential in the business model, with a software integration potential unprecedented!

# Business modeling

## "Business Modeling *buzzword*"

- Dissemination:
  - More than 5.9M of links to English written documents,
  - More than 1.1M of links to Spanish written documents
- In specialized digital libraries :
  - *ACM digital library:* 20,000 hits to "business process modeling"
  - *IEEE digital library:* 750 hits to "business modeling"

# Business modeling-II

## Business Model (BM)

Abstraction about how a business works. It depends on the expert(s) who made it.
Simplified view of the business structure to spread or improve it.
It is considered part of the definition of the information system's requirements, from which the information system will take place

# Business modeling-III

## Business

Set of activities of a company that owns or uses resources, have several goals and if it is carried out correctly and creates value (usually economic) for the company

# Motivation of using BP technology

## Benefits

- BBPP are the most valuable asset of a company
- Propitiates a better understanding of the key mechanisms of a business or company
- Acts as the basis for the creation of appropriate information systems that support the business
- Supports the improvement of the structure and operation of the current business
- Shows the structure of an innovative business
- Experiments with new business concepts or copy or study a concept used by a competing company
- Identifies outsourcing opportunities for products or supplies

# Characteristics

## Business processes:

- Very large and complex
- Highly dynamic
- Distribution
- Depends on economic activity or on the company's business rules
- Partially automatable

# Views

## Objective:

They allow highlighting only the important information, at a certain stage, to describe more easily the functions performed by a business process

## Types:

- Product/services
- Data
- Process
- Function
- Organizational

# Differences with other process types

| **Types of processes** | Industrial | Information | Business |
|---|---|---|---|
| **Focus** | Things | Data | Relationships |
| **Purpose** | Transform and assembly materials and components in other components and final products using resources | Process and transmit structured data and non- structured and knowledge | Meet conditions that satisfy the needs of participants clients or users |
| **Characterization** | Traditions of Industrial Engineering | Traditions of Informatics and Computer Engineering | Based on structures of communication and coordination of humans through languages |
| **Actions** | Assembly<br>Transform<br>Transport<br>Store<br>Inspect | Send, Invoke<br>Record, Recuperate<br>Consult<br>Classify | Solicit, Promise<br>Offer, Turn down<br>Propose, Cancel<br>Measure |

# Business Process based Systems

## Development elements

- BPS are based on information technologies (IT)
- Manage all phases of the business process life cycle
- Transparent deployment of computer systems
- Provide visibility and control
- Integrate ICT technologies as they have reached a level of maturity
- Process-guided management
- They allow to establish a common language among all the stakeholders

# Structure of Business Process Modeling (BPM)

## Process schemata

- More general/abstract perspective than carrying out system development based on OO/WS
- Generalize and integrate development processes focused on data and functional decomposition

| **OO/WS** (before) | **BPM** (now) |
|---|---|
| Applications | Manufacturing processes tools & services |
| Application server | Processes |
| Data Base | Fundamental process |

# Relationship between BPM and WS

| Standard BPM stack | | |
|---|---|---|
| Business Process Modeling Notation (BPMN) | | |
| Semantic Business Process Model (SBPM) | | |
| Business Process eXtension Layers (BPXL) | | |
| Web Services stack (WSDL,UDDI, etc.) | | |
| WS-CDL | BPEL | BPQL |

# UML extension for business

## Eriksson-Penker extension

- Provide elements, not included in standard UML, to model BBPP
- To model: *events*, *resources*, *relationships*, *rules* y *objectives*, typical of BPM
- UML elements are used to represent them: *stereotypes*, *tagged values*, *restrictions*

# Business Process conceptualization with Eriksson-Penker notation

## Business Process-EPBE

Transforms and refines the *resources* of the model and, complying with the business objectives, gets to *create value*

# EPBE Notation

## Structural concepts

- Resources
- Processes
- Business goals
- Business rules

# EPBE Notation-II

## Notational concepts

- Stereotype
- Tagged value
- Constraint

# EPBE Notation-III

## Business concepts

- Purpose, documentation
- Propietary, actor
- Priority, risks, cost, ...

# Business views

## Concept of View

- Similar –from the modeling point of view– to the *use cases* of UML: they start the modeling and constitute a grouping element of the rest of the diagrams
- A view is not in itself a separate business model
- Each view defines its own types of diagrams

# EPBE Views

## Types:

- Initial Business View (BVV)
- Business Process View (BPV)
- Business Structure View (BSV)
- Business Behavior View (BBV)

# Initial Business View (BVV)

## Utility

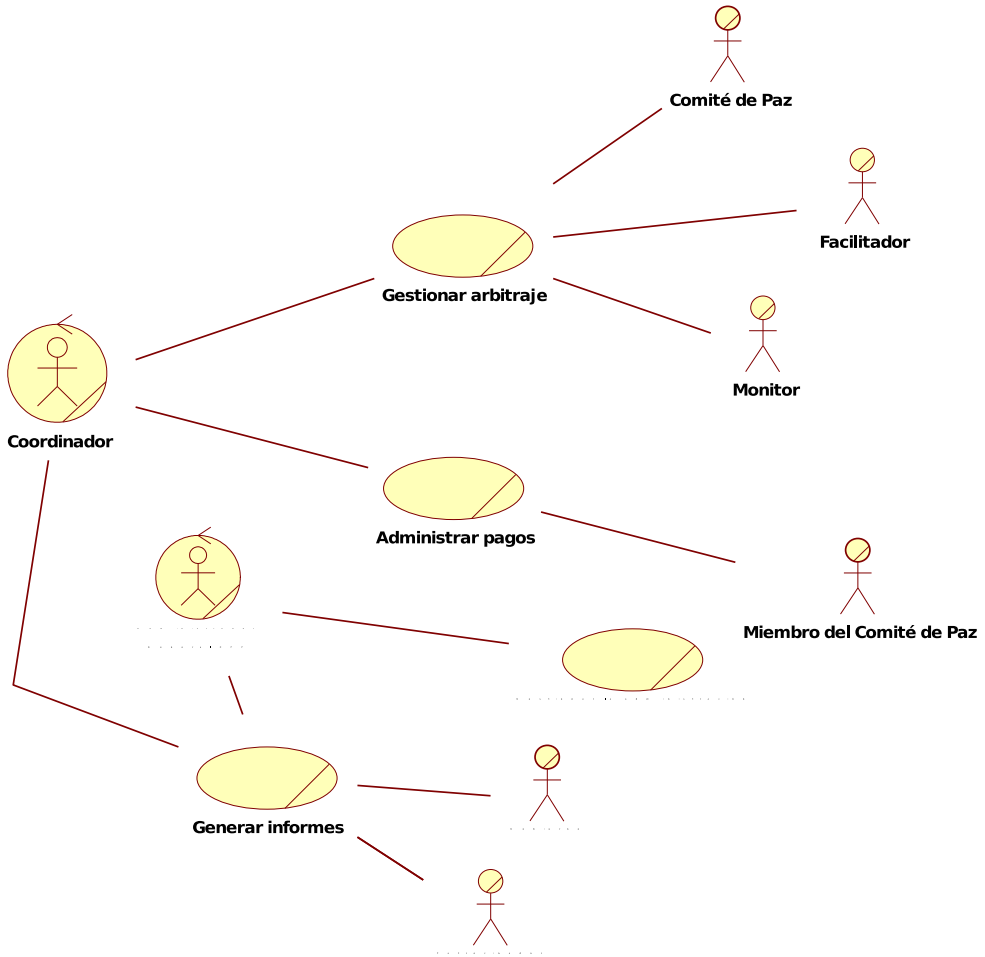
To represent the objectives
To model the other views
To make decisions for the final Information System (IS)

# Initial Business View (BVV)–II

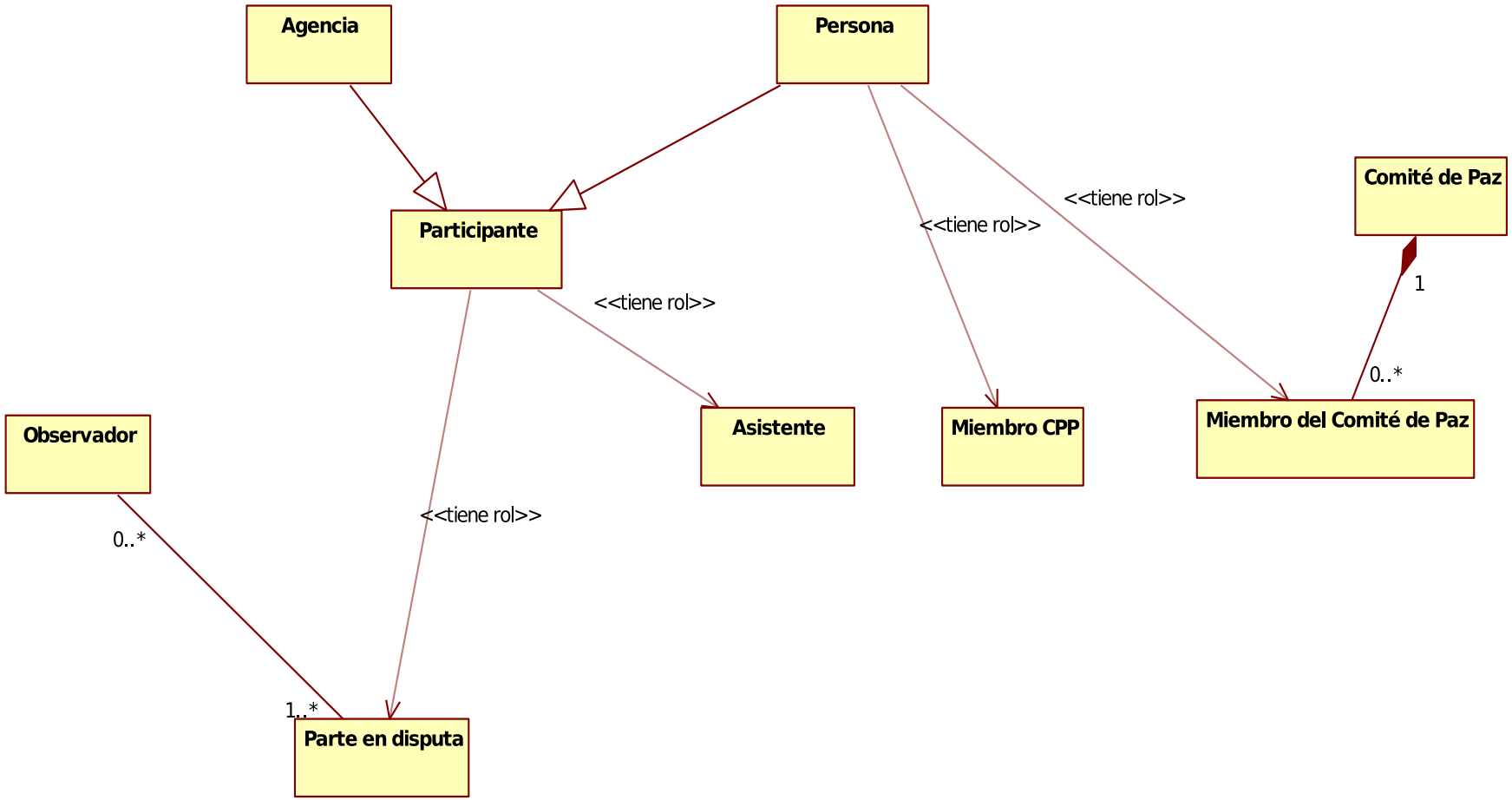
## Activities to realize

- Conceptual modeling:
  -Define the important concepts used in the business and their relationships
  -Initial Class Diagram: capture concepts and their relationships
  -Fix all attributes and operations
- Modeling of problems and objectives:
  -Business objectives (why, what and how)
  -Problems to solve in order to achieve the objectives
  -Qualitative/quantitative goals
  -Dependency and association modeling

# Example: CPP Business Use Cases Diagram



# CPP: Initial class diagram

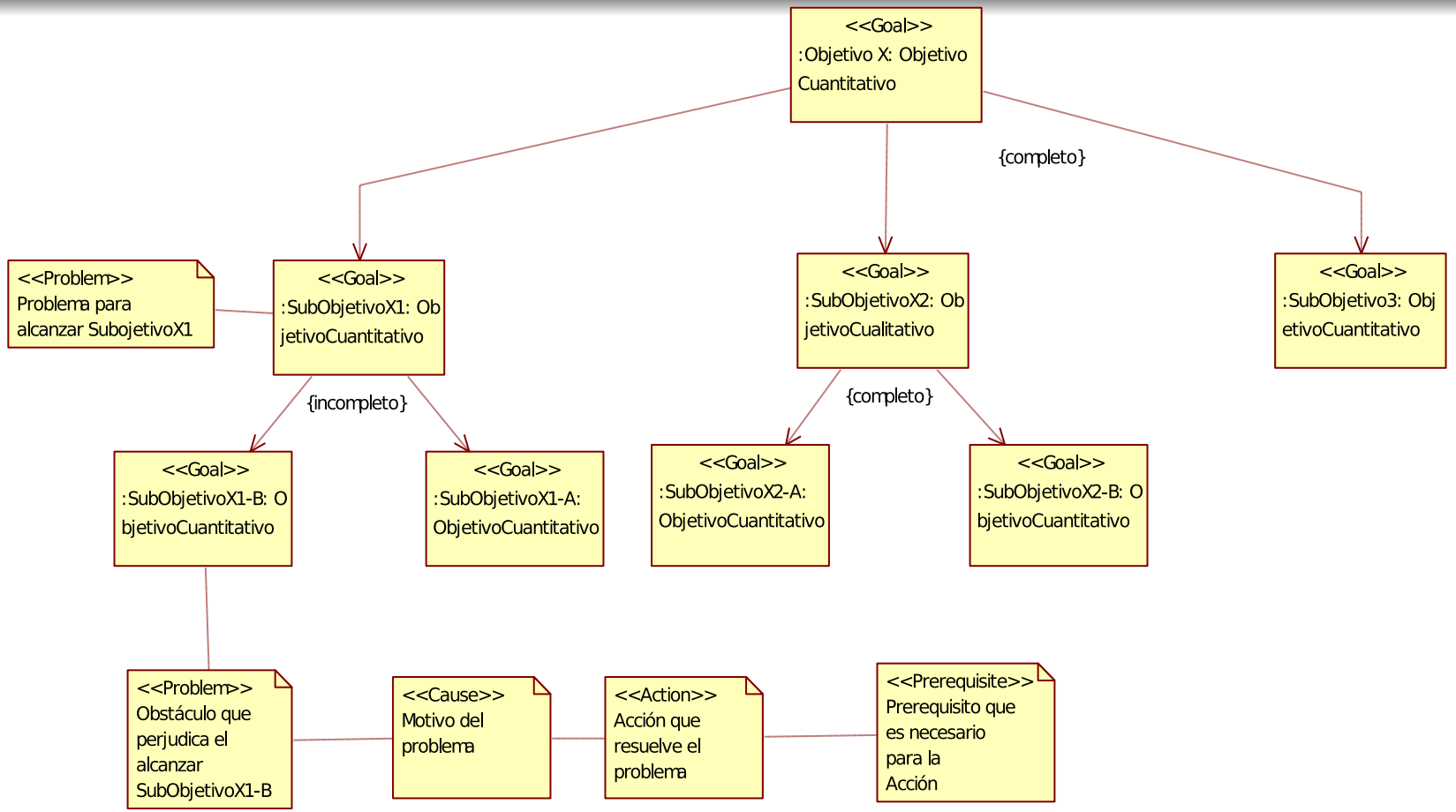


# Modeling of business objectives

## UML Object Diagrams

- Decomposition of high-level objectives in sub-objectives
- Objective classes:
  - Quantitative
  - Qualitative
- Object hierarchy as diagram objects
- Relations between objectives (dependencies and associations of UML)
- Annotations:
  - *{complete }*, *{incomplete }*, *{contradictory }*

# CPP Objectives Diagram



# Business Process View (BPV)

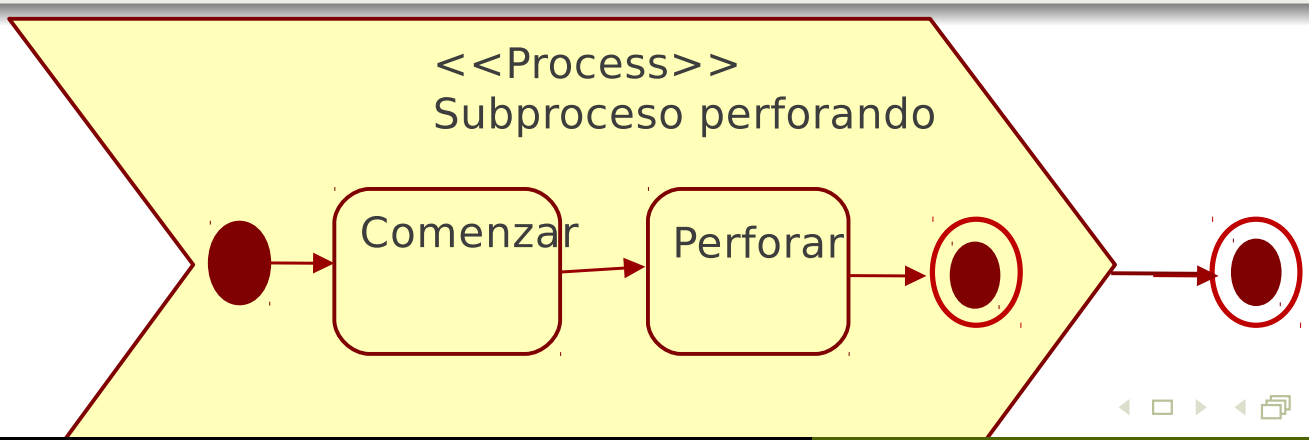
## BPV elements

- At the center of Business Modeling activity with the EPBE methodology
- Object types in diagrams:
  - *objectives, input / output, suppliers, controllers*
  - *pools, swimlanes*
- Stereotypes for processes and resources are added:
  `<< Goal >>`, `<< Process >>`, `<< Information >>`, `<< People >>`, `<< Physical >>`, `<< Abstract >>`

# Business Process View–II

## Subproceses and activities

- Subprocesses:
  - Same representation as processes
  - At the lowest *decomposition level*, they are called activities
  - They must have a beginning and an end
- Activities:
  - Direct / Indirect
  - Product quality assurance

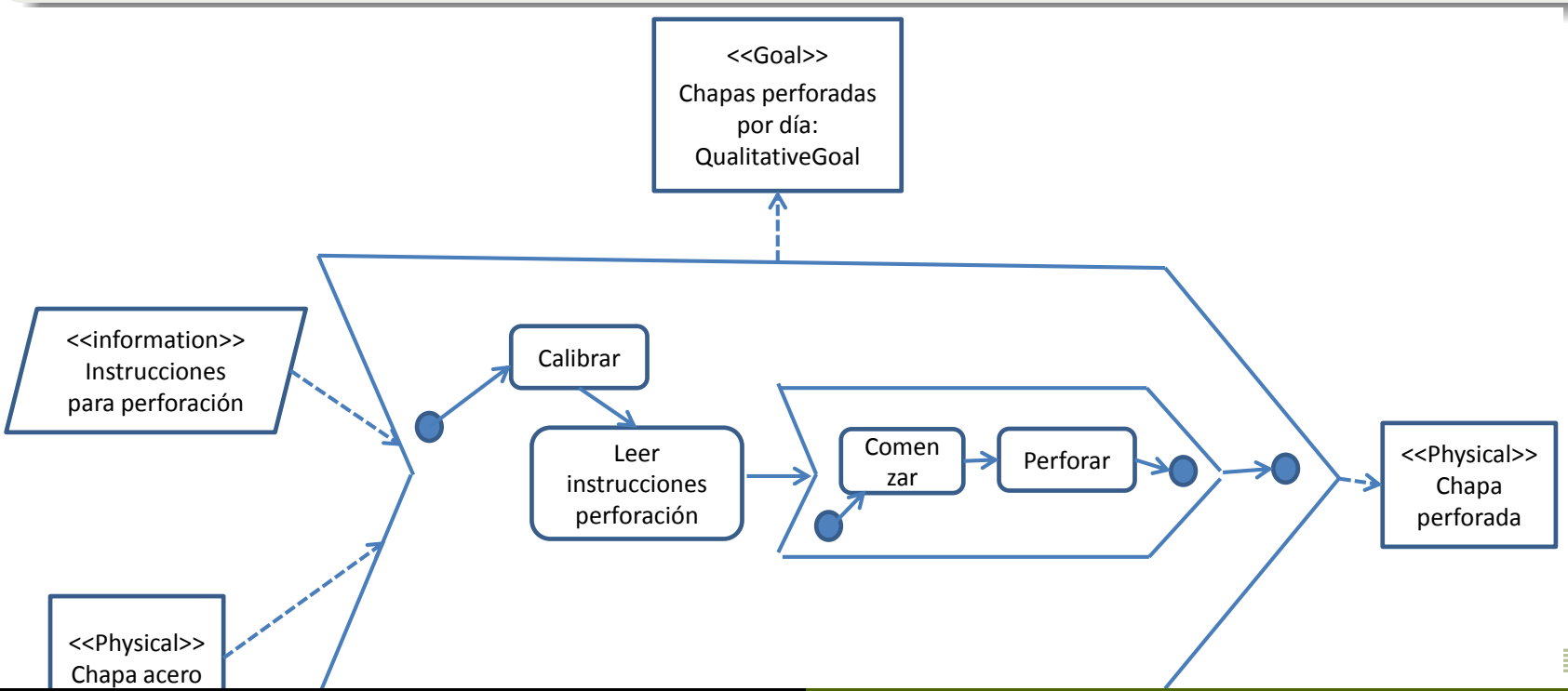


# Interaction of processes with objects–resources

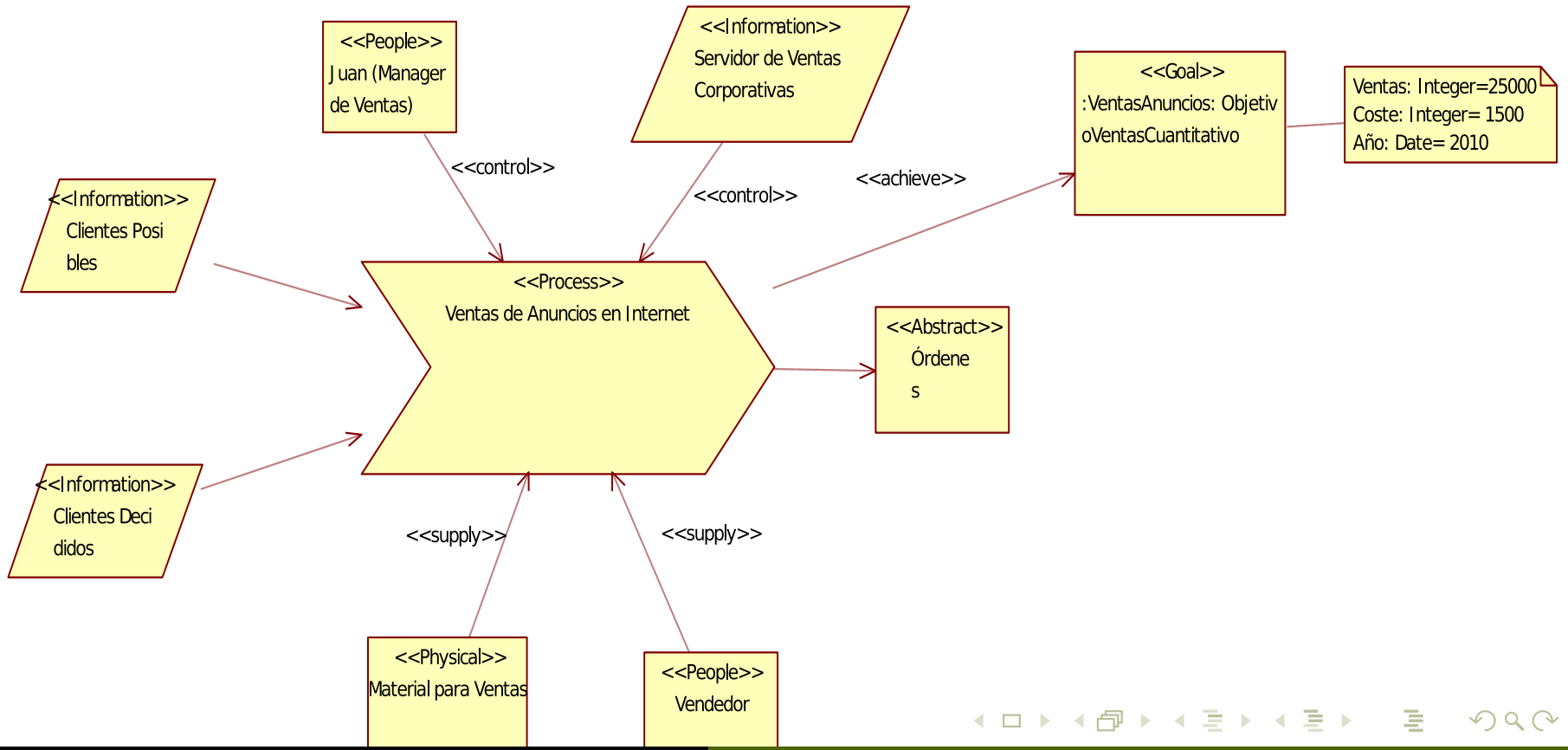
## Stereotypes

BBPP interact with resources:

`<< physical >>`, `<< information >>`, `<< abstract >>`, `<< people >>`



# Process diagram–EPBE

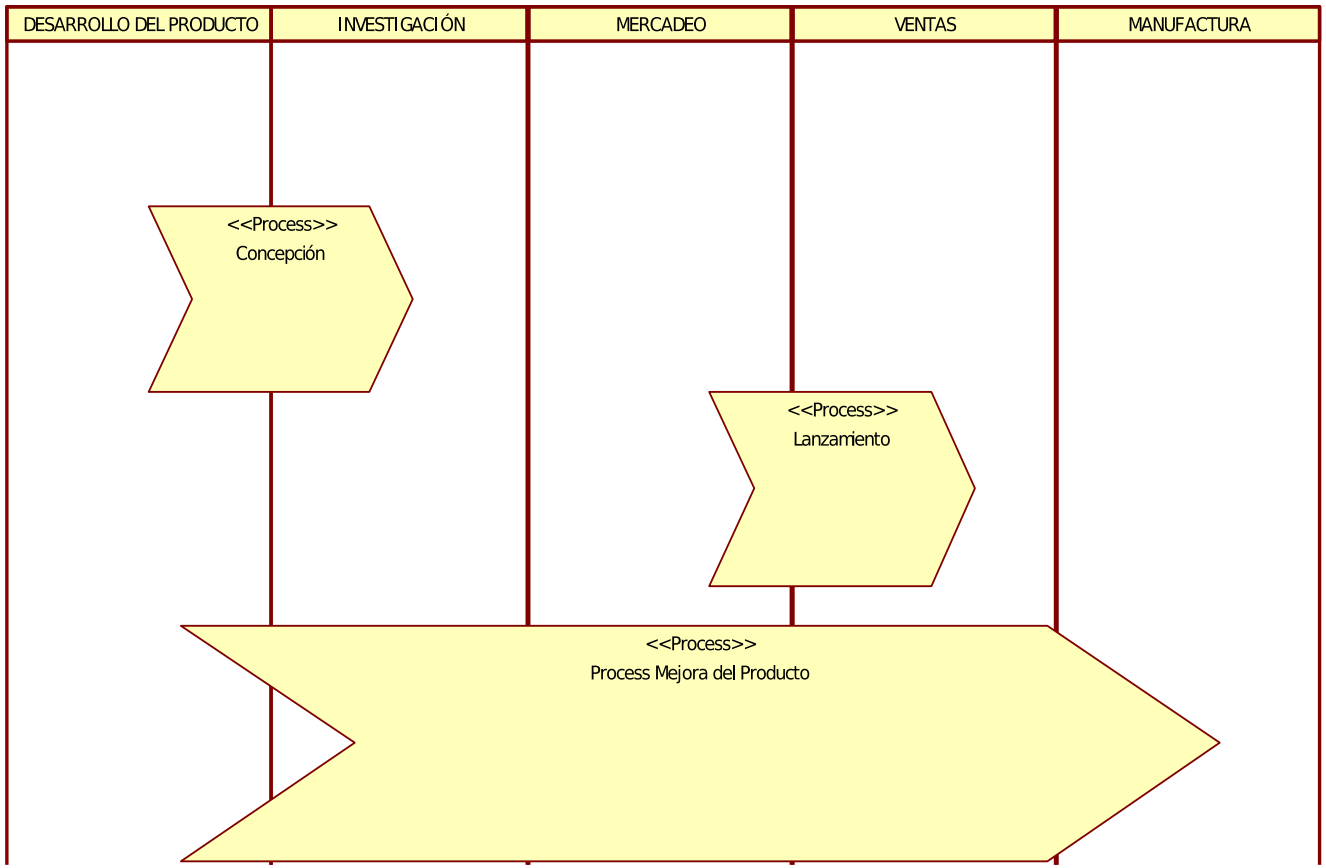


# Business Process View–III

## *Swimlanes* definition

- BBPP can span over several organizational limits of the company
- Decomposition into subprocesses is independent of departments, . . .

# Representation of swimlanes of a process



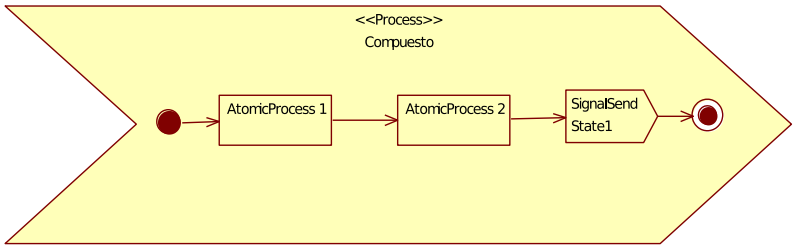
# Relationships between entities

## Types of relationships

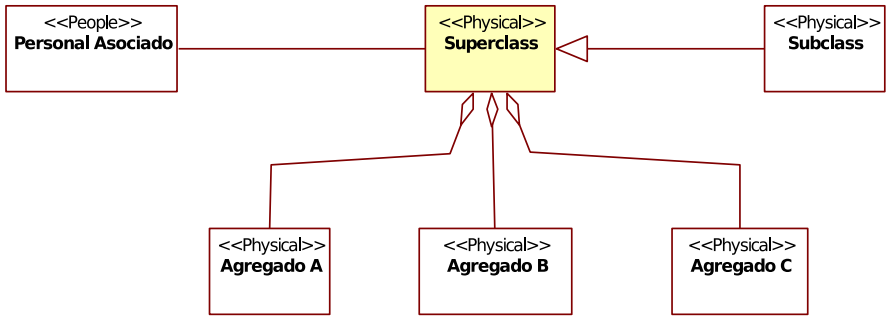
- The order of activities (I) is normally implied within the relationship defined between threads with each other and with the process that contains them
- Generalization, association and aggregation (II) are the same relations that exist between classes in UML
- Refinement (III) can be understood as a relationship of updating or implementation between a class and one interface; it also includes the relations of subtype or inheritance of UML
- Reference notes, similar to those used in UML diagrams

# Representation of relations in EPBE diagrams

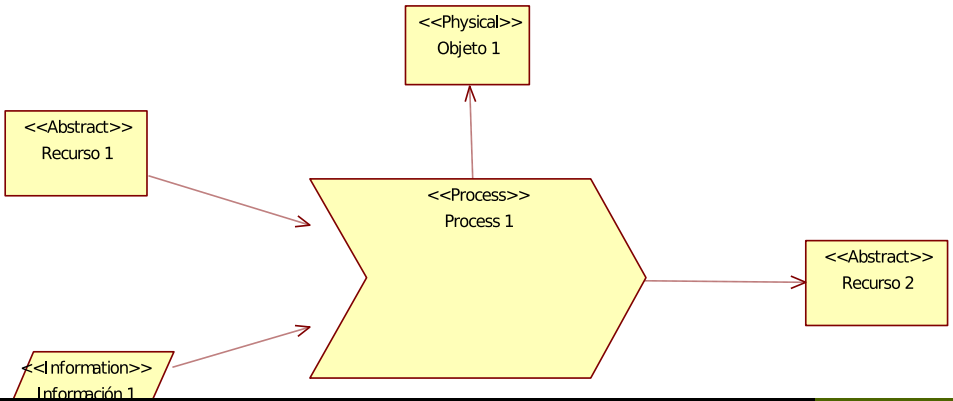
## I - Orden de actividades




## II - Generalización, asociación, agregación




## III - Dependencia



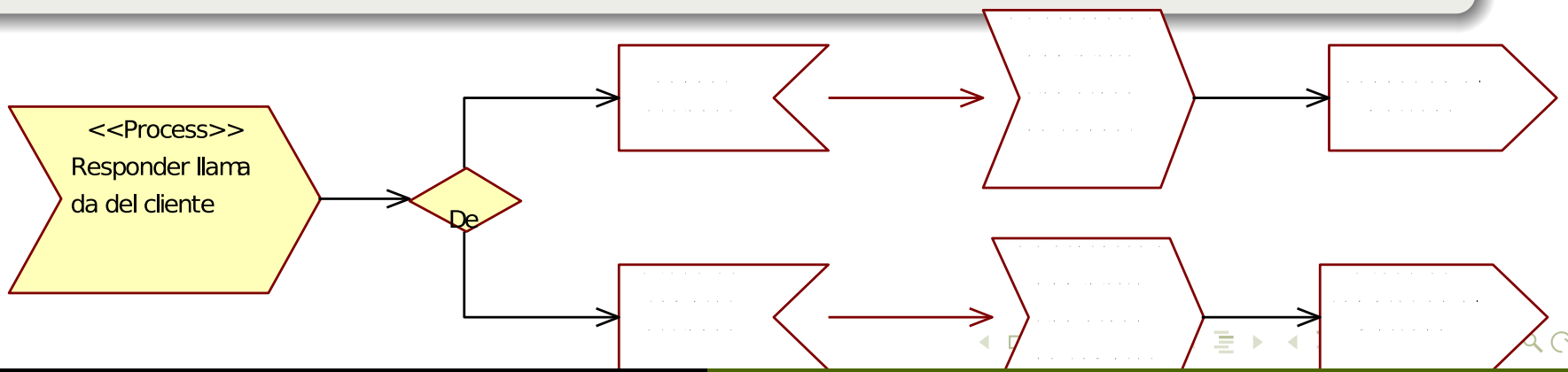

## IV - Refinamiento


## V - Nota de referencia

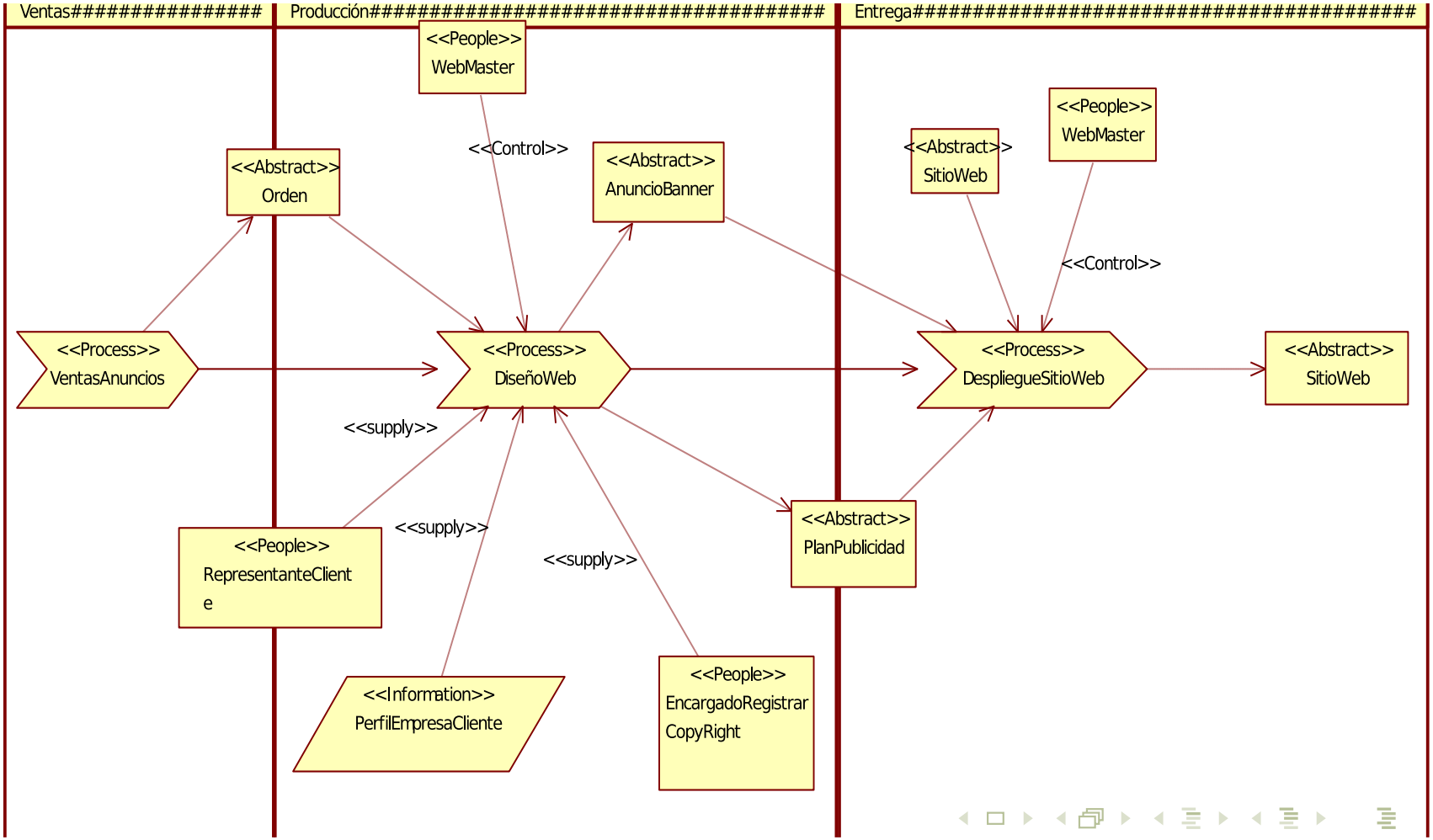
# Events between processes

## Treatment characteristics of events in EPBE

- A process can be itself affected by events from its environment or generated by other processes
- Initiate activities in the processes or conclude their execution
- They correspond to the opposite EPBE notation symbol, which is included in another diagram



# Process diagram with swimlanes

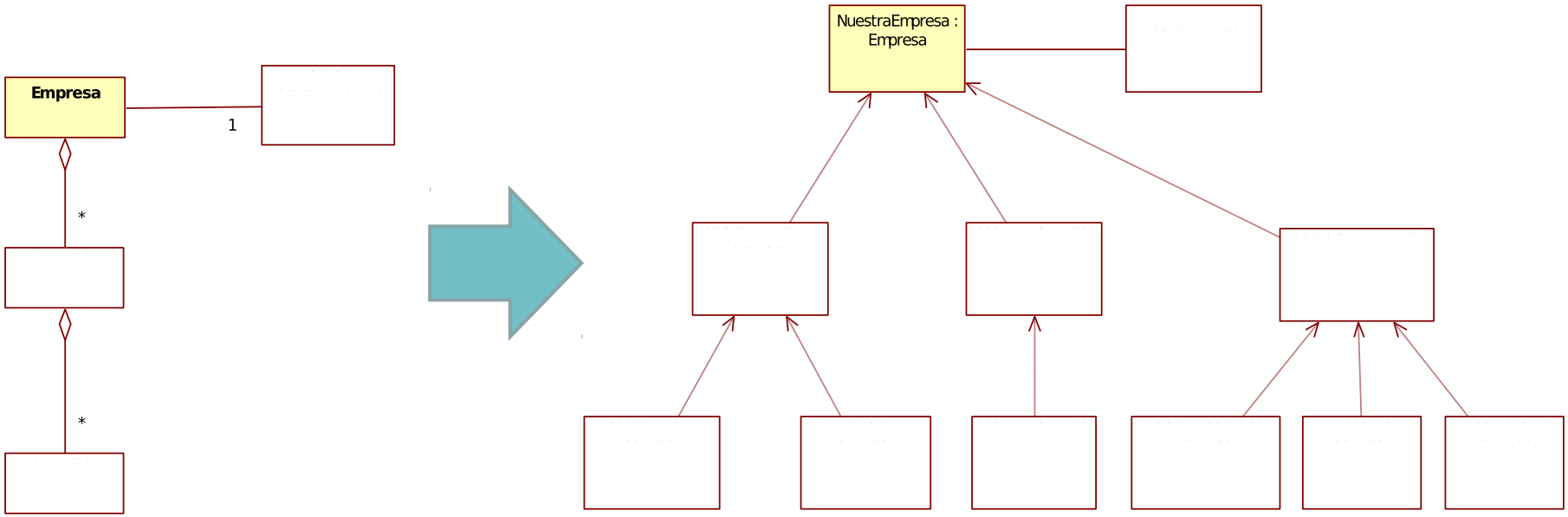


# Business Structure View (BSV)

## Expression of the structure of an organization.

- Class diagrams: show basic structures and rules of business organization
- Object diagrams: shows the actual organization, currently in use, in a company
- The organization's resources are allocated to each process
- The completion of the organizational class diagram will result in the company object diagram

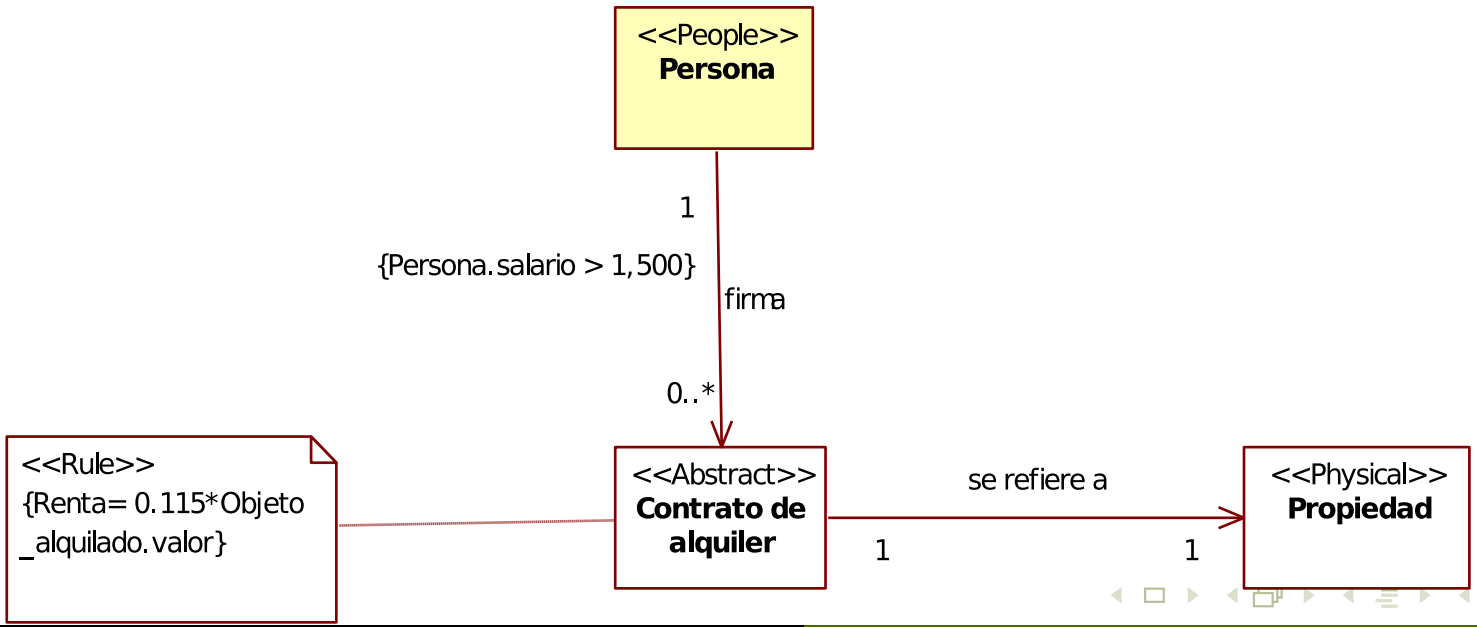
# Business structure



# Business Structure View–II

## Business rules

- The expression of the business rules in this diagram complements the previous diagrams and, together, they contain the information on how to conduct the business



# Business Structure View–III

## Business rules (conted.)

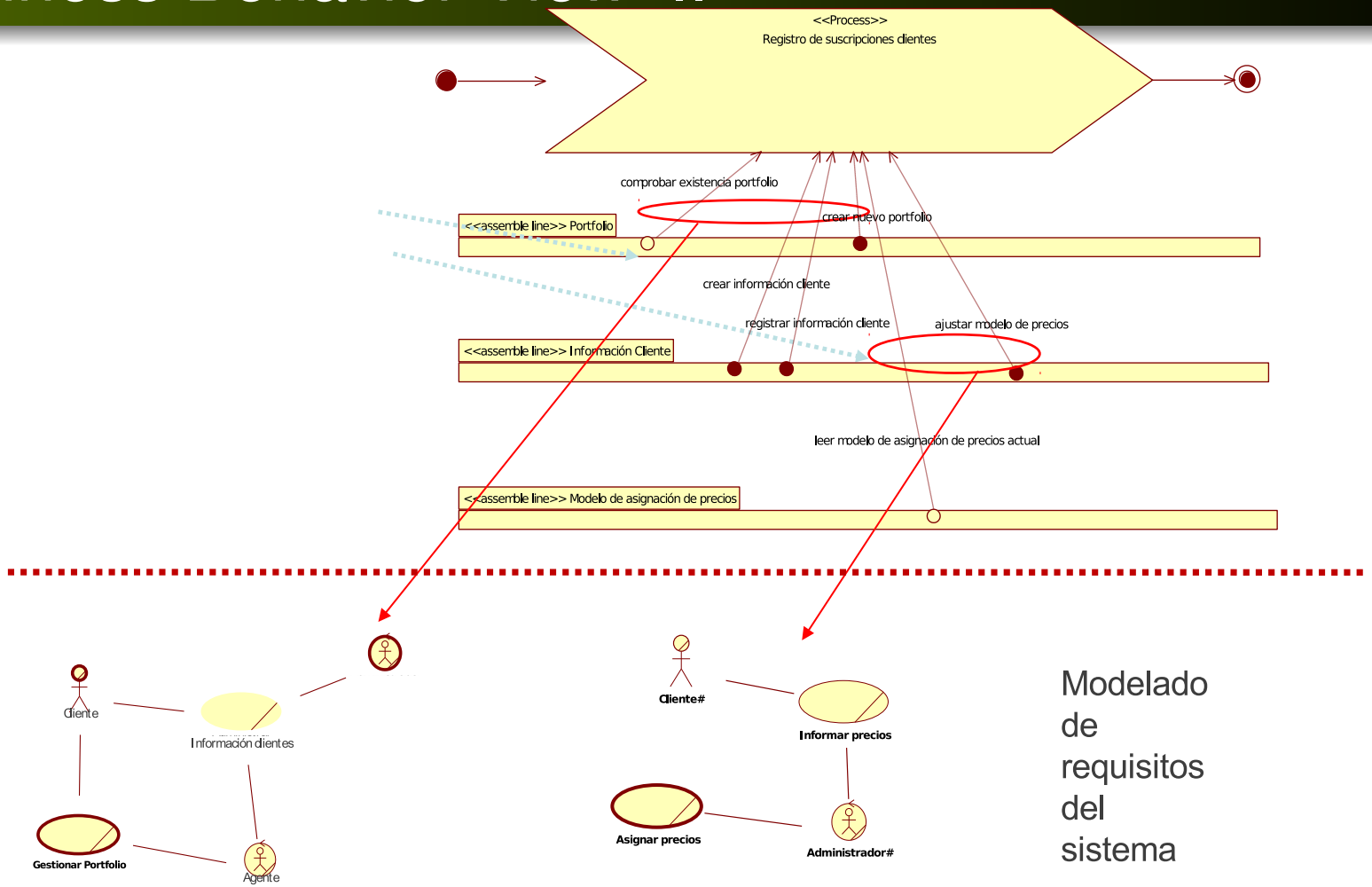
- Rules in the diagram affect the execution of business processes and resources structure
- Types of rules:
  - derivation
  - restriction
  - existence
- UML provides support to implement these rules

# Business Behavior View (BBV)

## Objectives:

- To describe behavior of resources in greater detail than in BPV, i.e.:
  - their status,
  - their behavior in each state,
  - and possible state transitions
- To describe system activities, transformations and their functionality, while concentrating on the interactions between resources, objectives and on business rules

# Business Behavior View–II




Modelado de requisitos del sistema

# Business Behavior View (BBV)–III

## Types of UML diagrams that are used:

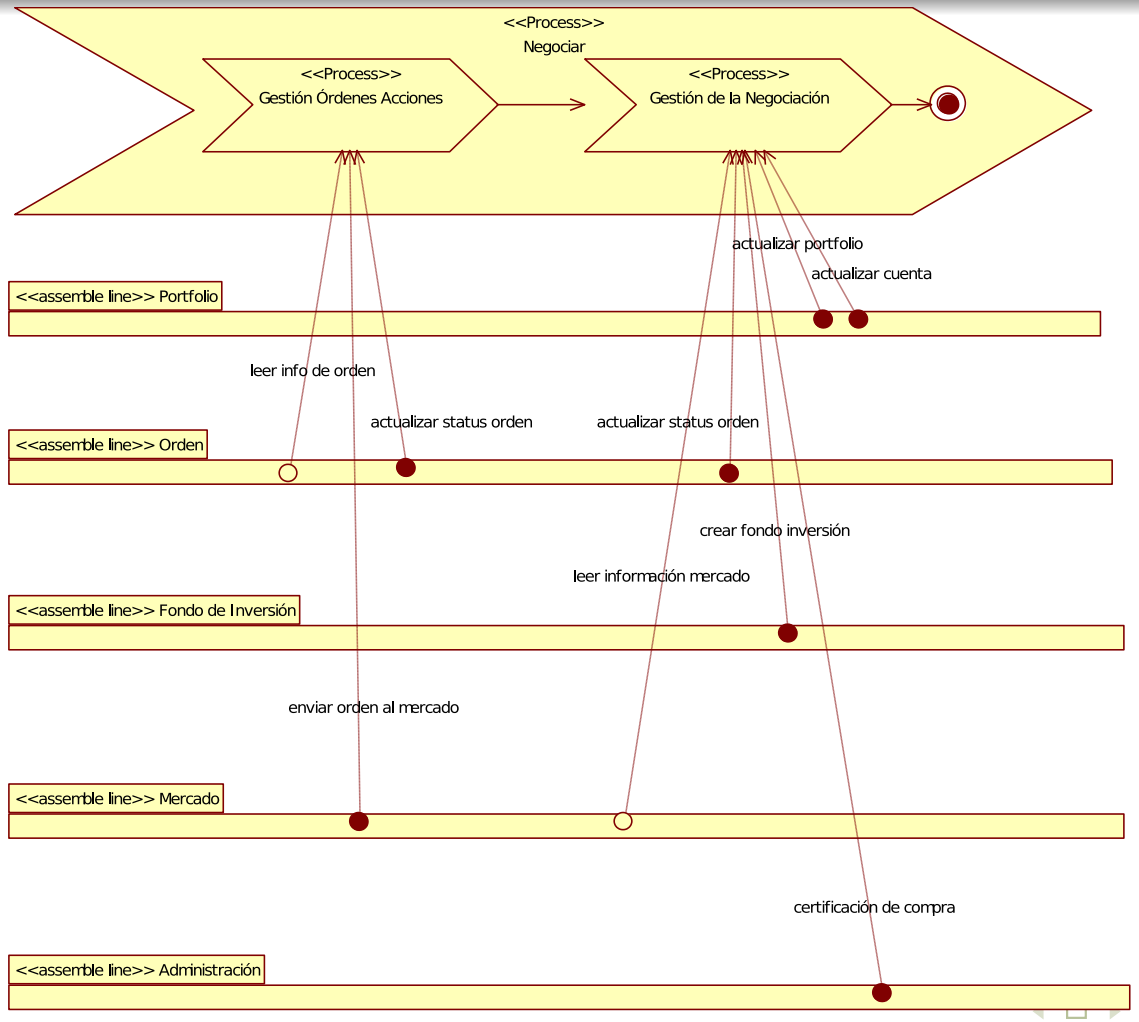
- State Transition Diagrams (STD),
- Sequence Diagrams,
- Collaboration Diagrams,
- Process Diagrams
- Assembly Line Diagrams

# Business Behavior View (BBV)–IV

## Conditions:

- It makes possible describing the dynamic behavior of each object which is described in the BPV
- Consistency between BPV and BBV views has to be make sure
- In EPBE a new type of diagram is introduced: Assembly Line Diagram

# Assembly Line Diagram



# Ontology of a service-oriented architecture WS 2.0

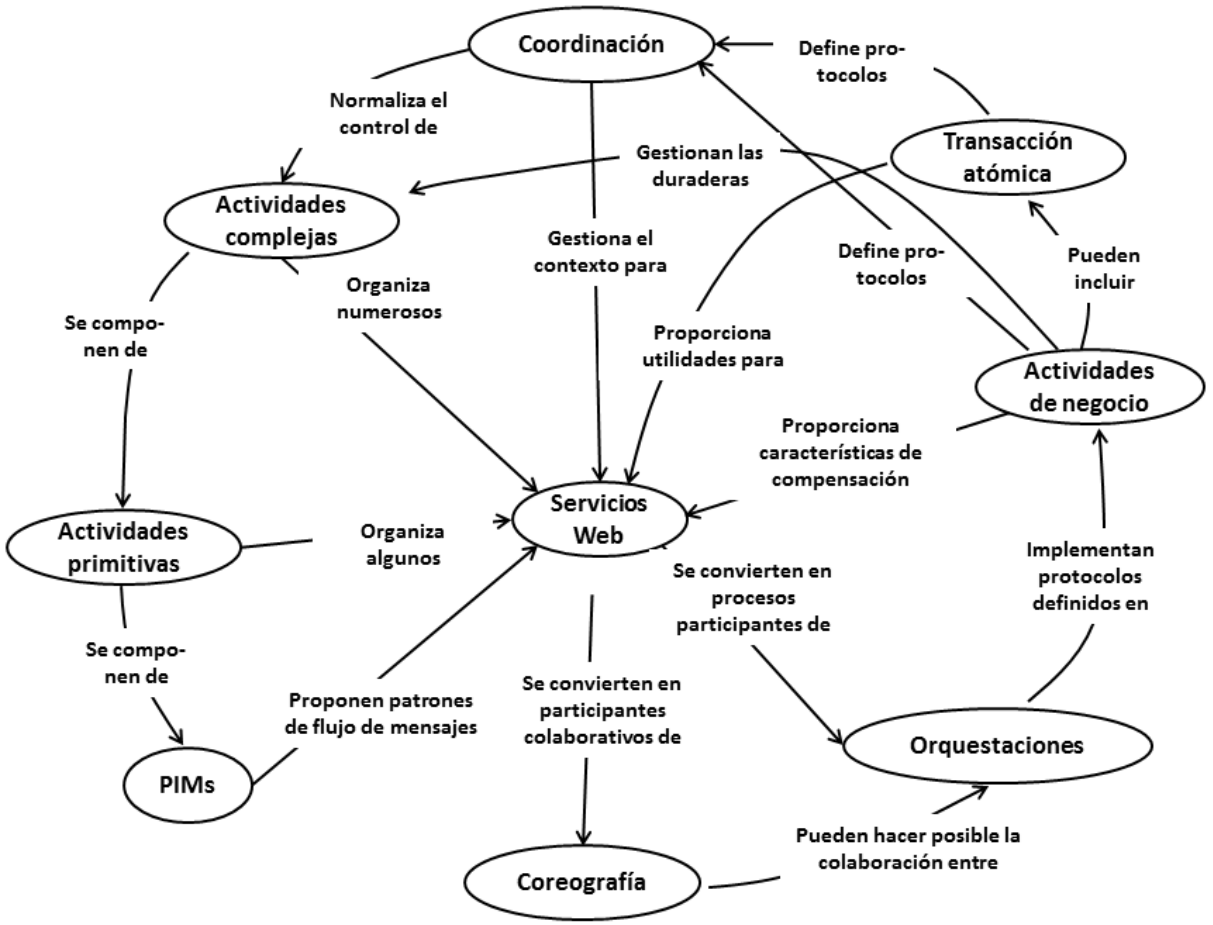



Figure: Concepts and relations that a WS 2.0 architecture includes

# Introduction to WS composition

Figure: WS collaboration

## WS composition

"Process of aggregation of multiple services into a single service in order to carry out more complex functions by an application or a business model"

# General Ontology of services composition in WS 2.0

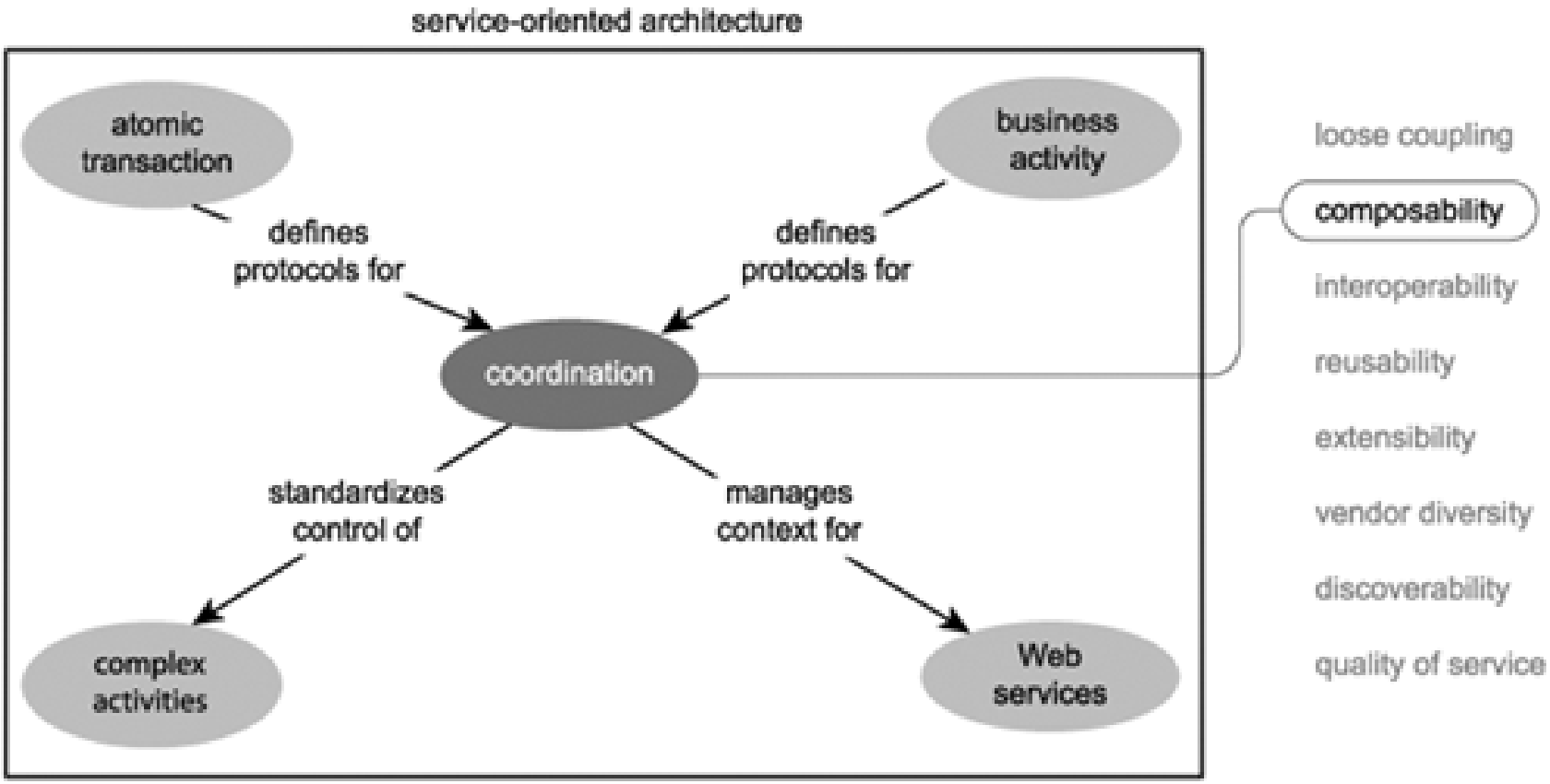



Figure: Relationship with the other parts of the architecture

# WS Orchestration

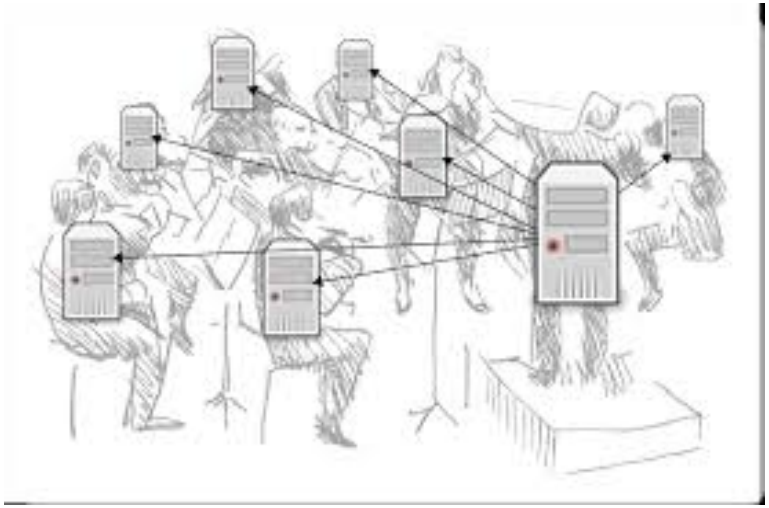


Figure: Orchestration of WS

## Fundamental idea

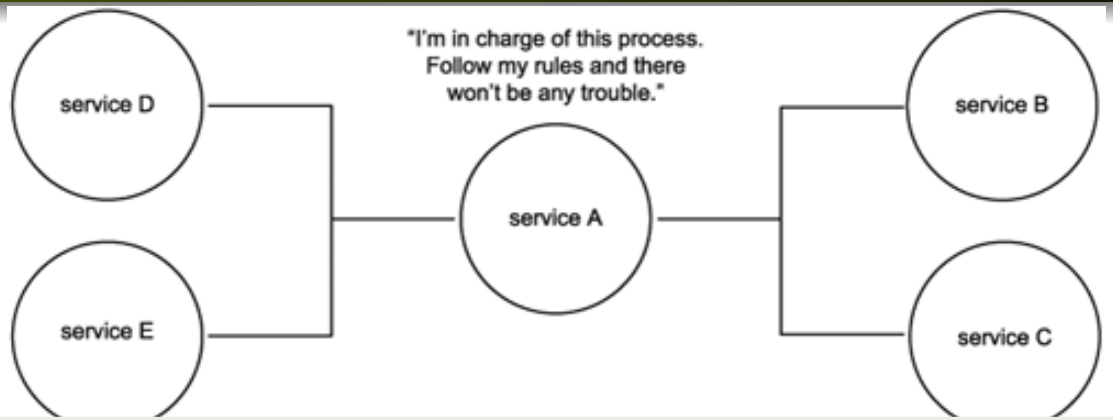
It is a well-defined and centralised part of a business information system *work-flow logic*, which eases the *interoperability* among 2 or more different applications

# WS orchestration

## Objetives

- To make true a possible integration of big business processes without the necessity of rebuilding applications
- Application collaboration is obtained through the introduction of a *work-flow logic* in business models
- To abstract the *work-flow logic* from the problem solution
- Orchestration's deployment becomes most important in service oriented development
  - *work-flow logic* of a business process then only needs requests of previously deployed services

# Orchestration as a service




- Company federation and service orientation
- Service design propitiates the *interoperability*
- As it is another service, it gives system's extension without affecting any of the component interoperability
- The logics that is included in an orchestration allows us to obtain the representation of any company as standard models

# Web Services Business Process Execution Language (WS-BPEL)

## Características of WS-BPEL/BPEL4WS/BPEL

- Key extension for making possible to obtain 2nd generation WS
- WS-BPEL makes use of concepts and WS-* terminology, in order to model business processes
- Currently valid standard: WS-BPEL 2.0 (see www.oasis-open.org)

# Web Services Interoperability Interface Information at OASIS

## Web Services Interoperability Interface

# Business protocols and process definition

## Orchestration characteristics

- *Work-flow logics*: events, conditions and business rules
- A protocol is defined, i.e.: how the participants of an orchestration interact with each other in order to perform each one a task of the business
- The *work-flow logics* in an orchestration is contained in a business process definition

# Services of processes and among associated

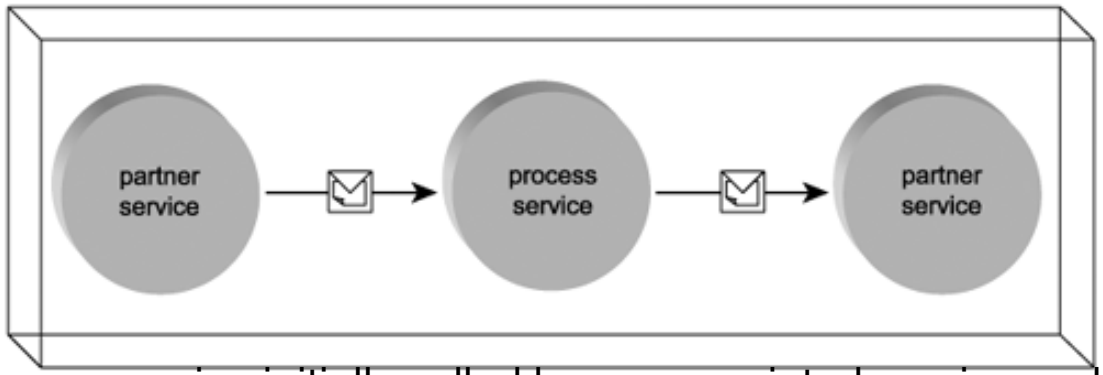



Figure: A process service initially called by an associated service and then invoked by another

## Orchestration semantics

- The participant processes are represented as services
- The *process service* can get being associated (become a "partner") and/or being called remotely

# Work-flow descriptions with WS-BPEL

## Fundamental idea that supports BPEL

- To decompose any *work-flow logics* into a series of pre-established basic actions
- To identify *work-flow* basic actions that can be assembled by using a logics of structured activities: *secuence*, *switch*, *repetition*, etc.
- It is important to get an unambigous and previously defined execution order of activities

# Work-flow descriptions with WS-BPEL - II

## Orchestration and synchronization

- A work flow cannot end until all the activities that it contains are finished
- A global synchronization/orchestration logics is therefore obtained from individual *work-flows*
- The modularity principle is then applied to *work-flows*, which can be obtained by the definition of *links*: formal dependencies among the activities carried out on different flows
  - An activity cannot be completed until all the requirements defined on its output links are fulfilled
  - Before starting any activity the input links requirements (contained items) must be satisfied

# Orchestration and SOA

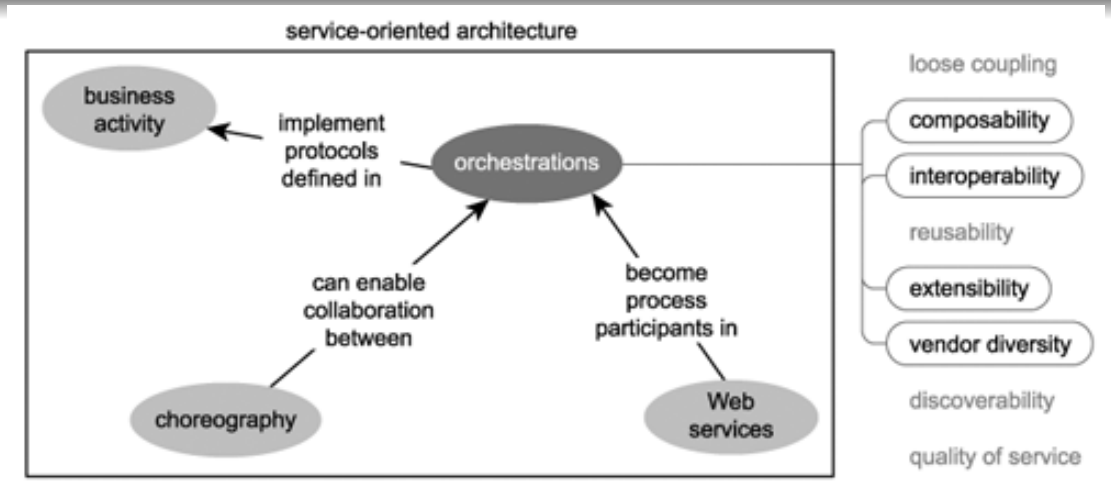



Figure: Orchestration connected with other parts of a SOA

## Business companies automatization principles

- The process logics of a company becomes centralised as an orchestration
- Orchestrations propitiate application environments oriented to extensible and adaptative services

# Benefits of using orchestration in a SOA

## Summary

- Centralised modification of the *work-flow logics*
- To ease the fusion of business processes
- Wide scale service definition (SOAs) leverages the future evolution towards federated companies
- Current technology helps to attain the objective: today's middleware allows us to integrate orchestration engines in environments of service oriented applications

# WS choreography

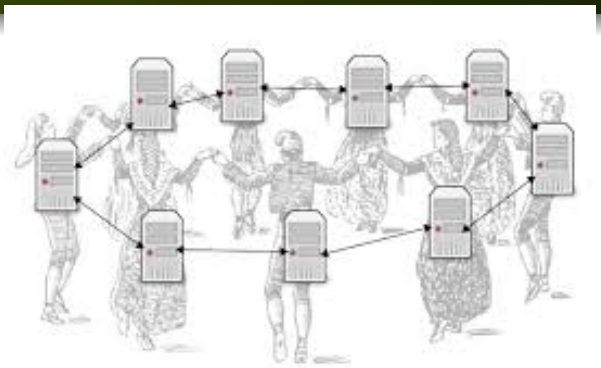


Figure: Choreographies of WS

## Fundamental idea

A well-defined part of the *business logics* that eases the *interoperability* of services when collaboration goes beyond of company/organisation limits.
Can be seen as universal collaboration and interoperability patterns for carrying out business tasks that are shared by different organisations.

# WS choreography – II

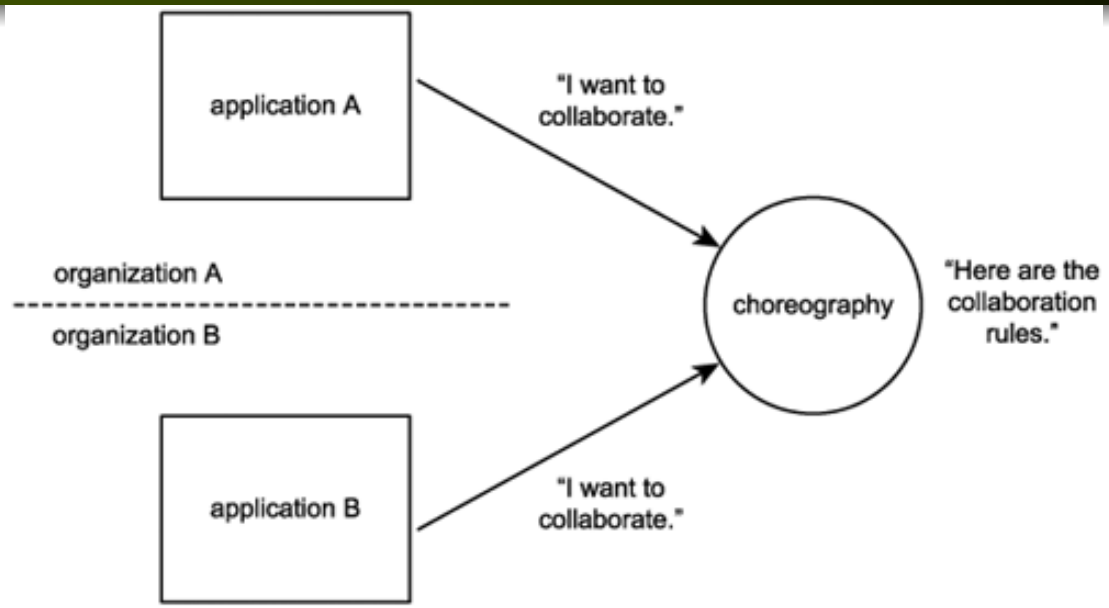



Figure: A choreography makes possible the colaboration among participants.

## Motivation

Multiple services of different organisations need a "choreography" when they have to work together in order to attain a common objective in the deployment of a business task

# Choreography as a service

## Characteristics of a choreography

- Serve for exchanging *public messages*, i.e., to set the collaboration among WS that belong to different organisations
- Differently from the “orchestration" mechanism, no organisation can control the *collaboration logics*
- Can be also used for creating a collaboration among applications that belong to the same organisation
- *Actions* within a choreography are structured as message exchange sequences among WS

# Choreography as a service – II

## Channels

- A message exchange is defined as a relationship between one pair of roles, each one of these assumed by a partner
- A choreography action is defined by a set of role-pairs of a WS
- A channel defines the representation of messages exchange between 2 partners of a role-pair
- Information about a channel can be passed on a message in order to obtain a way to implement *service discovery*
- *Work units* of a choreography include interactions among WS, which are constrained by conditions set to finish up each communication successfully

# WS-CDL choreography description

## Web Services Choreography Description Language

- Programming language based on XML that serves to describe the collaboration among partners by definition of each relation's participant common and observable behaviors
- The language specification is due to W3C ( W3C Web Services Choreography Working Group, closed in July 2009), ver: https://www.w3.org/TR/ws-cdl-10/
- At moment, we can observe overlapings between WS-CDL and WS-BPEL for orchestrations
- WS-CDL specifications propitiate dynamic discovery of services and the widescale collaboration among many participants

# Web Services Choreography Description Language Specification Site

## WS-CDL Interface

W3C Candidate Recommendation

W3C

### Web Services Choreography Description Language Version 1.0

W3C Candidate Recommendation 9 November 2005

**This version:**
http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/
**Latest version:**
http://www.w3.org/TR/ws-cdl-10/
**Previous version:**
http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/
**Editors:**
Nickolas Kavantzas, Oracle
David Burdett, Commerce One
Gregory Ritzinger, Novell
Tony Fletcher, Choreology
Yves Lafon, W3C
Charlton Barreto, Adobe Systems Incorporated



#### Abstract

The Web Services Choreography Description Language (WS-CDL) is an XML-based language that describes peer-to-peer collaborations of participants by defining, from a global viewpoint, their common and complementary observable behavior; where ordered message exchanges result in accomplishing a common business goal.

The Web Services specifications offer a communication bridge between the heterogeneous computational environments used to develop and host applications. The future of E-Business applications requires the ability to perform long-lived, peer-to-peer collaborations between the participating services, within or across the trusted domains of an organization.

The Web Services Choreography specification is targeted for composing interoperable, peer-to-peer collaborations between any type of participant regardless of the supporting platform or programming model used by the implementation of the hosting environment.

#### Status of this Document

# Web Services Choreography Description Language Specification Contents

## WS-CDL Interface

W3C Candidate Recommendation

Publication as a Candidate Recommendation does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as othe
work in progress.

Table of Contents

# Reusability, compositionality and modularity

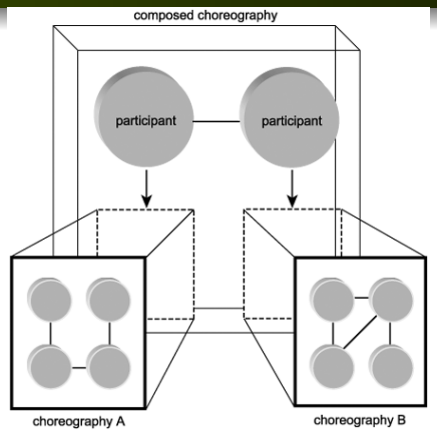
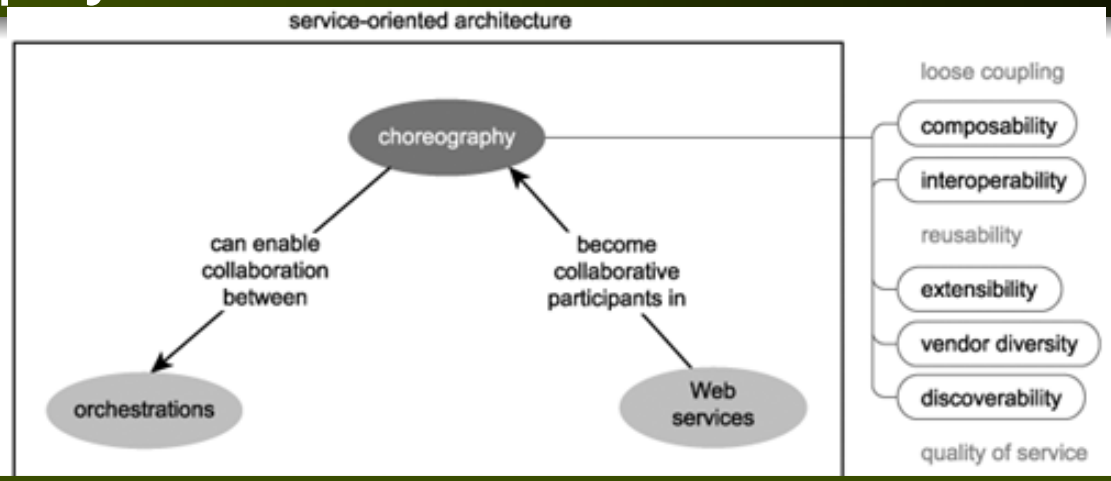



Figure: A choreography made up of 2 smaller choreographies.

- They are designed in a *reusable* way to be applied to different business tasks, which share actions
- Propitiates assembling of choreographies through an importation service inside a framework
- They can be structured into modules, which include taks to be used by a hierarchy of choreographies

# Choreography and SOA




## Fundamental ideas

- Choreographies allow extending or dynamically modifying the participant business processes
- Help to configure complex SOAs that can go beyong the company limits
- Propitiate WS discovery and agil design and deployment of business tasks inside any organisation

# Orchestrations vs. choreograhies

## Coincidences and diferences

- An orchestration can be understood as the specific application of a choreography to a given business
- Either orchestrations as choreographies can be used for modeling complex message exchange patterns
- An orchestration normally represents the work-flow of an organisation:
  - The company owns and controls the collaboration logics
- The control logics of a choreography does not usually belong to only one organisation:
  - the message exchange pattern proposed by one choreography is normaly used for collaboration among WS of different organisations

# Cites

[ ]

# References I

Bibliographic references #26,27,29,33,35,52,69,77,78,93 y 126 of the book:

Capel, M. (2016).

*Desarrollo de Software y Sistemas Basados en Componentes y Servicios*.

Garceta Grupo Editorial, Madrid, one edition.