Composizione di Servizi Web

Giusy Di Lorenzo, Valeria Vittorini

Università degli Studi di Napoli Federico II

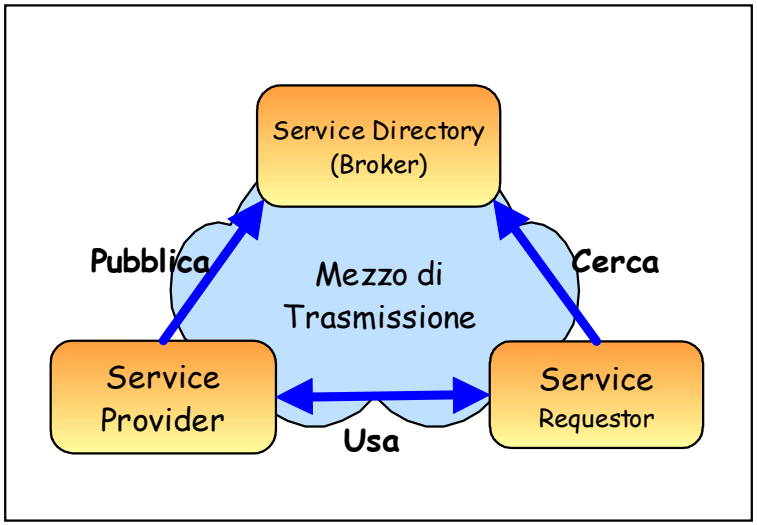
Corso di Laurea in Ingegneria Informatica

Corso di Applicazioni Telematiche 2008-09

Web Services

- Componenti software indipendenti dalla piattaforma e dall'implementazione
- Tecnologie e protocolli standard
- Invocati da agenti software attraverso API

- Attori coinvolti
  - Service Provider
  - Service Requestor
  - Service Directory



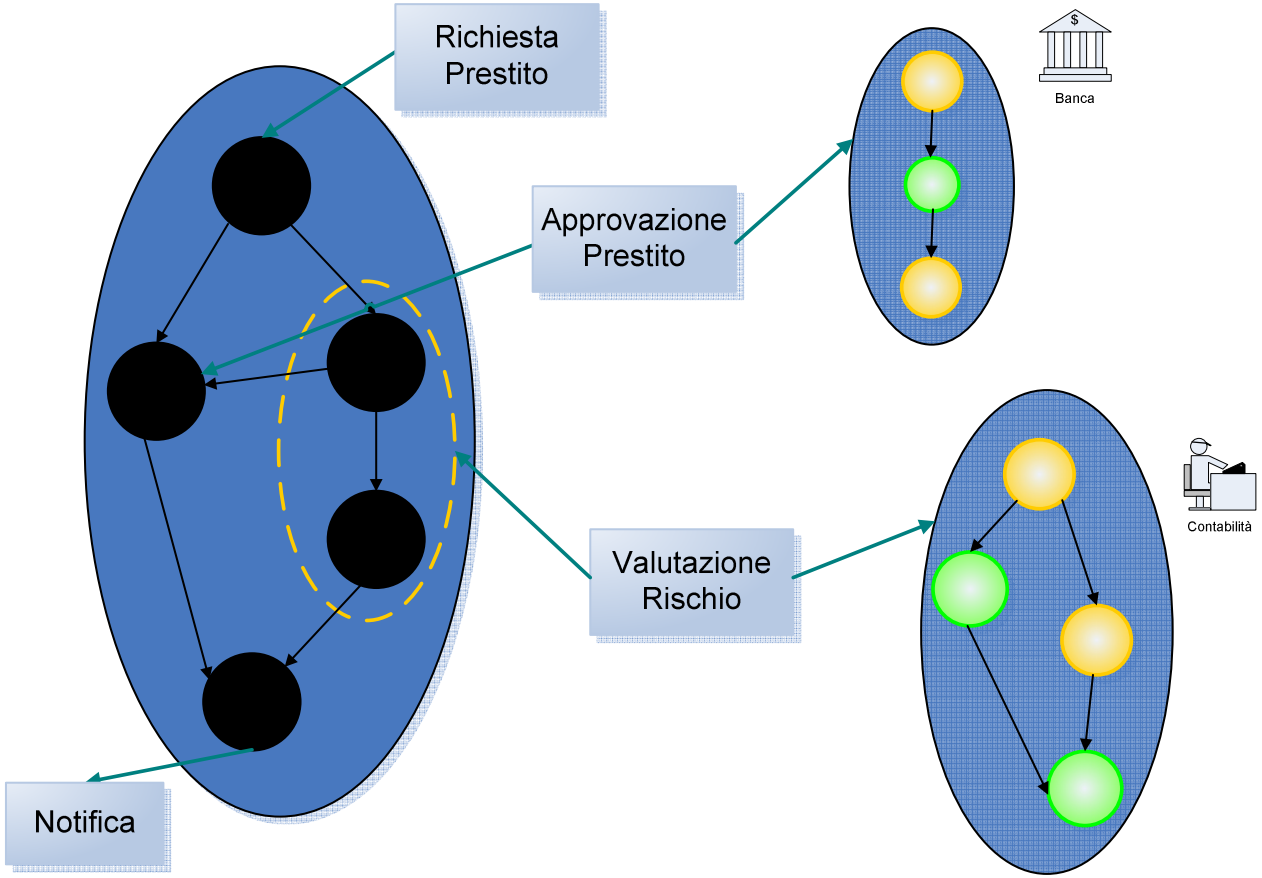
Standard WSDL

- Web Services Description Language (WSDL)
- Standard w3c per la descrizione dei web services
  - Il WSDL di un servizio web fornisce:
  - L'elenco dei servizi offerti;
  - I dati necessari al servizio;
  - I dati in risposta;
  - Le modalità di invocazione del servizio

Composizione di WS

- Obiettivo: Ottenere un nuovo servizio combinando le funzionalità offerte da servizi esistenti
- Vantaggi:
  - Integrazione di applicazioni
  - flessibilità
  - Sviluppo rapido e a basso costo

Esempio


Richiesta Prestito
Approvazione Prestito
Valutazione Rischio
Notifica
Banca
Contabilità

Composizione

- Richiede:
  - Specifica del servizio che si vuole ottenere
    - Requisiti funzionali
    - Requisiti non funzionali
  - Individuazione e selezione dei servizi componenti
  - Definizione della logica del processo di composizione (flusso)
  - Deployment ed esecuzione
  - Validazione del servizio composto

Flusso delle operazioni

Può essere definito in diversi modi

- Automi a stati finiti
- Reti di Petri
- State Charts

Hanno in comune il fatto di essere basati su una rappresentazione a stati

Flusso delle operazioni

- Può essere visto come un caso particolare di processo di business e realizzato mediante un workflow
- Processo di business: “A set of one or more linked procedures or activities which collectively realise a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships”
- Workflow: “The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.”

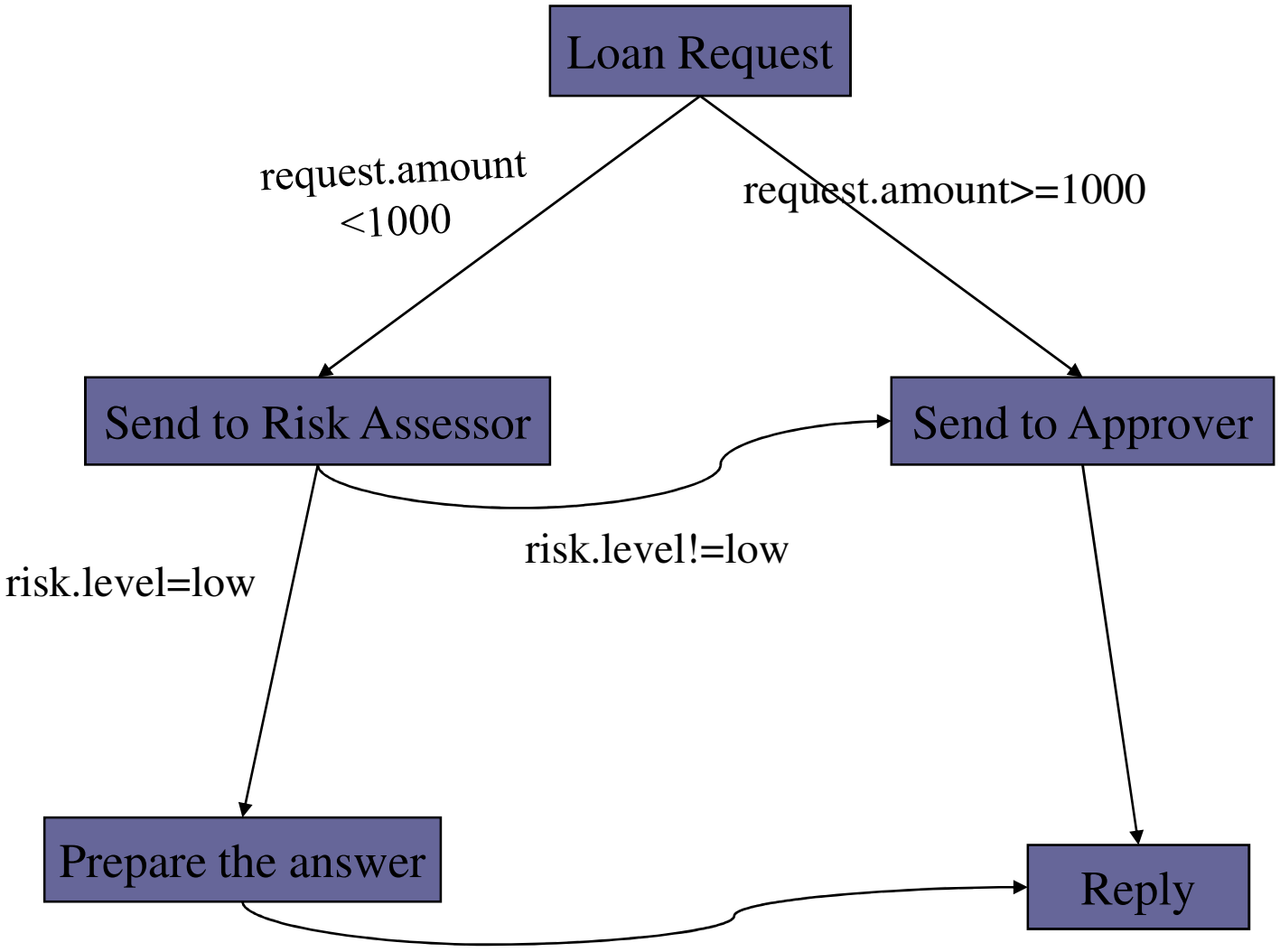
Workflow

- Elementi base:
  - Attività
    - rappresenta una funzione di business ben definita
    - Le attività possono essere composte in attività complesse
  - Data flow
    - rappresenta il flusso di dati scambiati tra le attività
  - Control flow
    - specifica la sequenza di esecuzione delle attività

Workflow e WS

- Il flusso logico di invocazioni richieste per realizzare un servizio composto ed il relativo scambio di messaggi è trattato come un caso particolare di workflow
- Serve un linguaggio per definire il workflow ed un motore che esegua workflow espressi mediante quel linguaggio

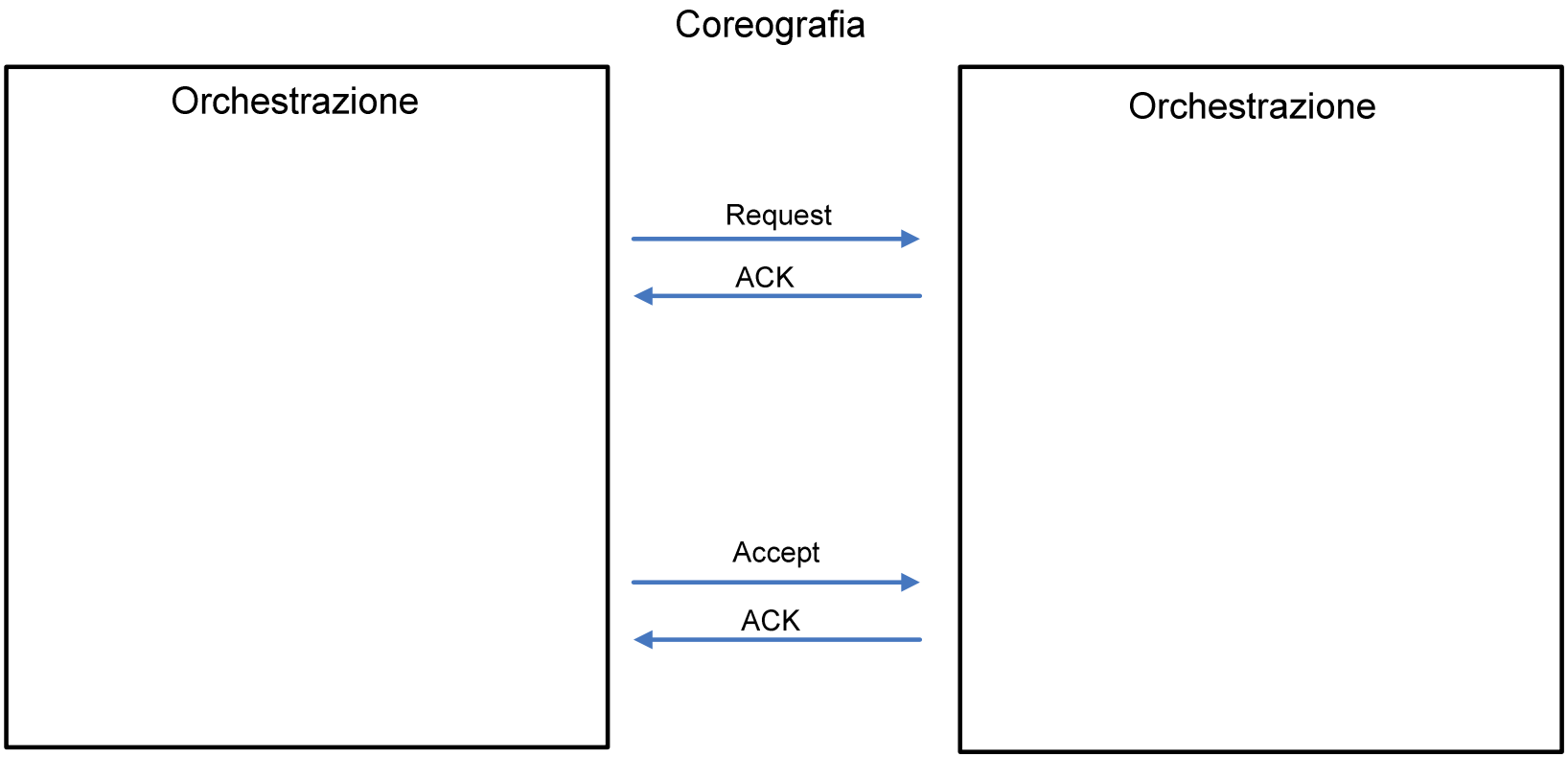
Il workflow dell'esempio



Orchestrazione

- Questo approccio alla composizione di servizi è di tipo *orchestrativo:*
  - *Coordinamento fortemente centralizzato*
  - *Controllo assunto da una delle parti*
- *Una immagine: l'orchestra ed il direttore di orchestra*

Orchestrazione e Coreografia



Coreografia

- **Web Services choreography** involves **non-executable descriptions** of observable behavior of Web Services through the definition of **observable message exchanges** between a collection of services
- Una immagine: la danza e la sua coreografia

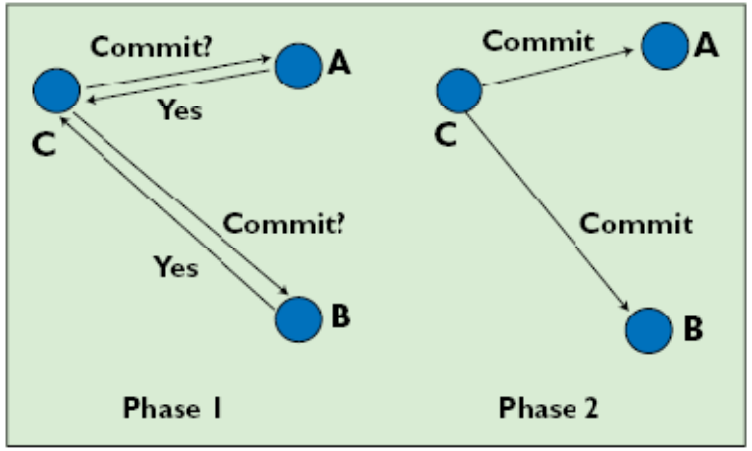
Requisiti del linguaggio di WF

- Invocazione dei servizi
  - Asincrona: affidabilità
  - Concorrente: aumentare le performance
    - Scalabilità
    - Throughput
    - Tempo di risposta
  - Gestione delle eccezioni: integrità delle transazioni
  - Gestione degli errori: compensazione

Transazioni di lunga durata

Transazioni Atomiche

- ACIDE
  - Commit a due fasi
  - lock



Transazioni di Business

- A causa di una messaggistica generalmente asincrona non è possibile riservare delle risorse per la durata necessaria

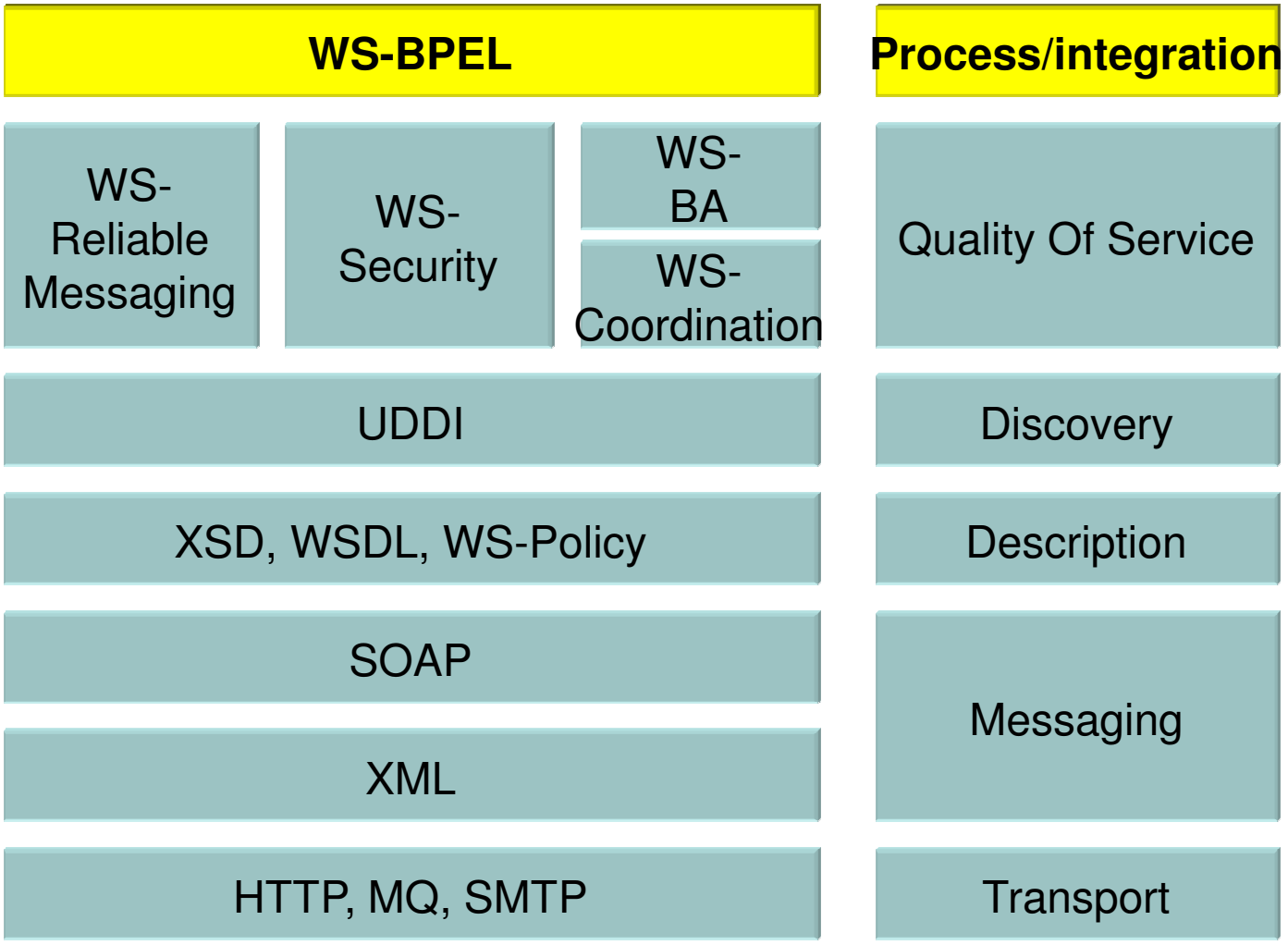
WS –BPEL: Business Process Execution Language For Web Services

- Definisce un modello e una grammatica per descrivere una logica di business tra più partecipanti. E' indipendente dalla piattaforma e basato su XML.
- Permette di creare un nuovo servizio web per composizione di servizi esistenti (orchestrazione)
- I servizi terminali sono rappresentati dai propri documenti WSDL (Web Service Descriprion Language): riuso di WS e incapsulamento di funzionalità

WS- BPEL

- Contains process flow constructs for conditional branching, parallel processes, nested sub-processes, process joins, etc.
- Standard OASIS (Organization for the Advancement of Structured Information Standards e-business standards)
- A joint specification of IBM, BEA, Microsoft, SAP, and Siebel

WS stack


WS-BPEL
Process/integration
WS-Reliable Messaging
WS-Security
WS-BA
WS-Coordination
Quality Of Service
UDDI
Discovery
XSD, WSDL, WS-Policy
Description
SOAP
XML
Messaging
HTTP, MQ, SMTP
Transport

BPEL e WSDL

- I processi BPEL vengono esposti come servizi Web tramite WSDL, con cui vengono descritti i punti iniziali e finali del processo;
- I tipi di dati WSDL vengono utilizzati all'interno di un processo per descrivere le informazioni passate tra le richieste;
- Con WSDL è possibile fare riferimento a servizi esterni.

- A BPEL process is a **reusable definition** that can be deployed in different ways and in different scenarios, while maintaining a **uniform application-level behavior** across all of them
- BPEL includes **transactional capabilities** for business processes, as well as **compensation activities** that “undo” the results of longer-running transactions
  - **Example:** A compensation activity for a purchase order activity would result in the status of the pertinent purchase order being changed to “Cancelled”

Processo BPEL

- Un processo è visto come un insieme di attività
- Elementi fondamentali:
  - Attività semplici e strutturate
  - Variabili
  - Scope (ambito di visibilità)
  - Correlazione
  - Compensazione
  - Fault handling

Principali attività

- <receive> allows the business process to do a blocking wait for a matching message to arrive.
- <reply> allows the business process to send a message in reply to a message that was received through a <receive>. The combination of a <receive> and a <reply> forms a request-response operation for the process.
- <invoke> allows the business process to invoke a one-way or request-response operation on a portType offered by a partner.

Principali attività

- <assign> is used to copy data from one place to another.
- <throw> generates a fault from inside the business process.
- <terminate>: terminate the entire service instance. It is only available in executable processes.
- <wait> allows you to wait for a given time period or until a certain time has passed.
- <empty> allows you to insert a “do nothing” instruction to the process.

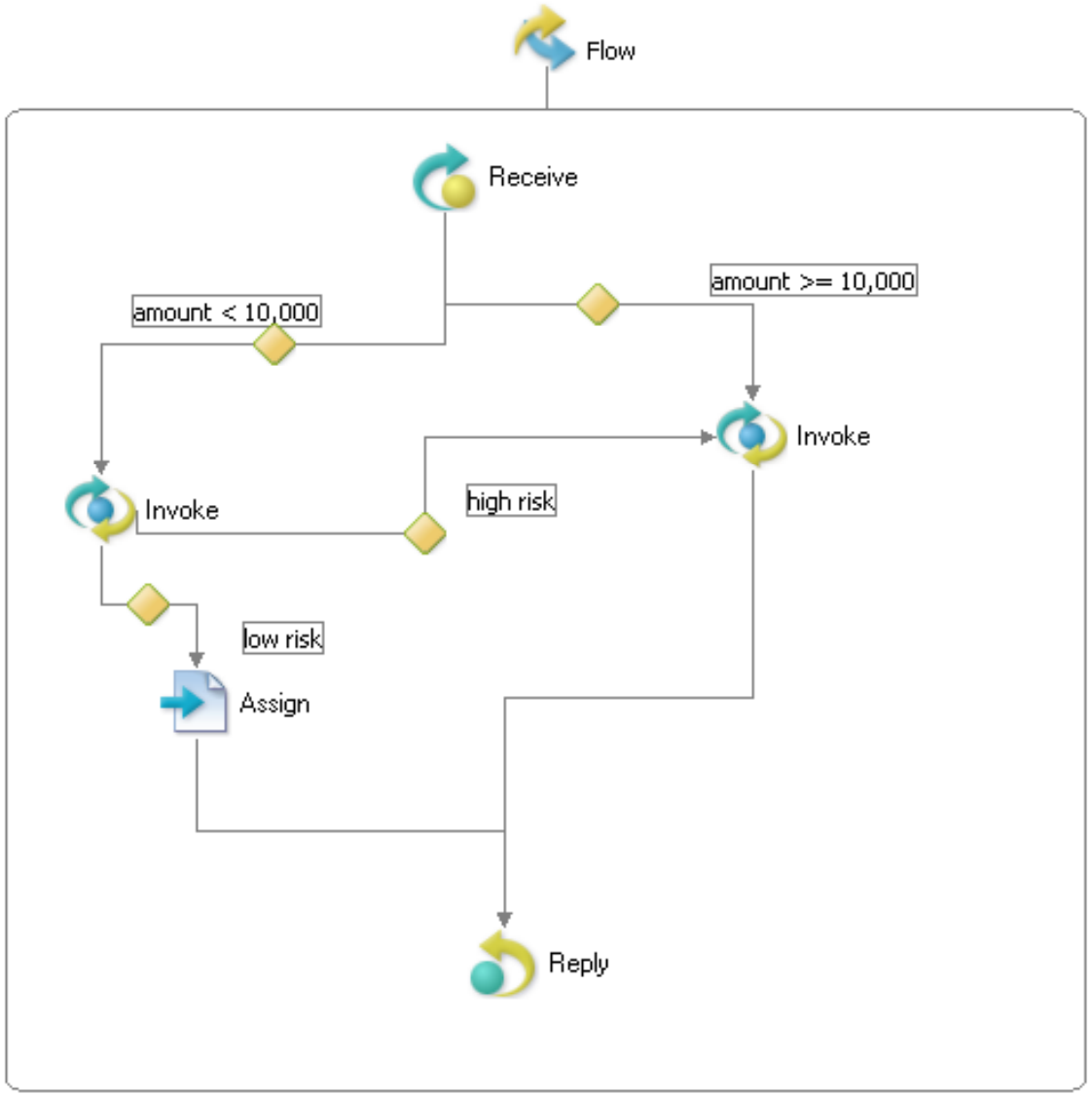
Costrutti di controllo
(definiscono la sequenza delle attività)

- <sequence>: an ordered sequence of steps
- <switch>: “case-statement” approach
- <while>: loop
- <pick>: execute one of several alternative paths
- <flow>: parallel steps

Partners

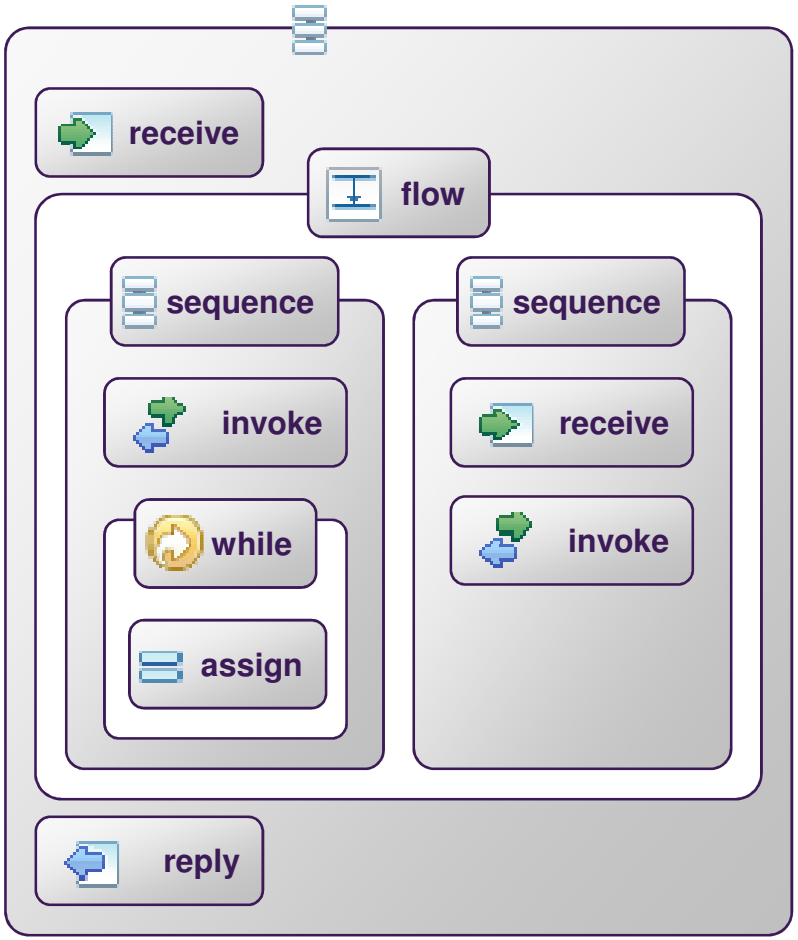
- For <invoke>
  - Invoke an operation at an external web service
  - The external web service is *Process partner*
- For <receive> and <reply>
  - The client sends message to invoke a local web service
  - The client is *client partner*
- <partnerLinks> : The different parties involved in the business process

Servizio approvazione prestito



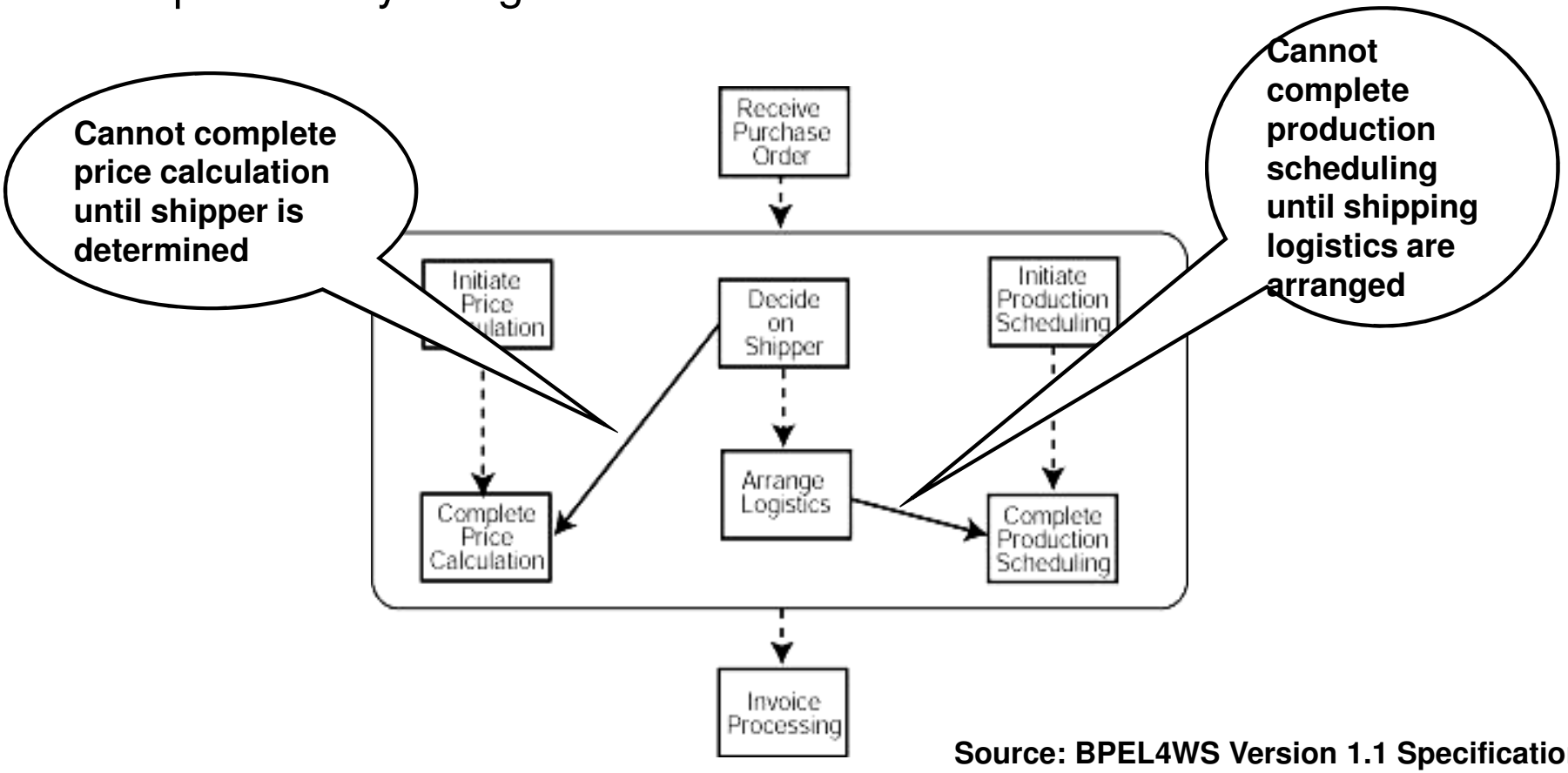
Attività Semplici e Strutturate

```
<sequence>
   <receive .../>
   <flow>
      <sequence>
         <invoke .../>
         <while ... >
            <assign>...</assign>
         </while>
      </sequence>
      <sequence>
         <receive .../>
         <invoke ... >
      </sequence>
   </flow>
   <reply>
</sequence>
```



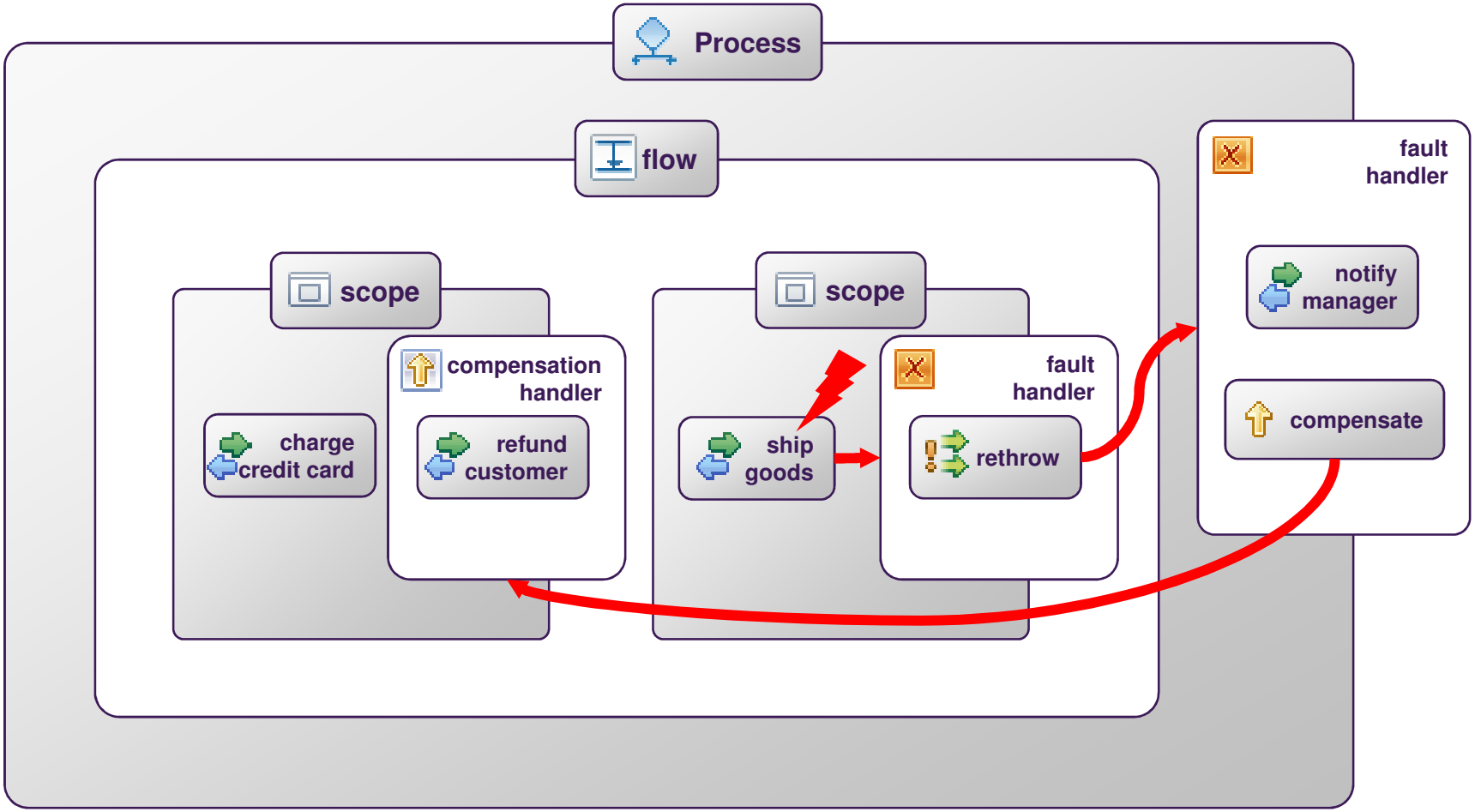
BPEL is capable of modeling complex business processes, and the dependencies between various tasks

- The following is a BPEL process for handling a **purchase order:**
- The synchronization dependencies between concurrent tasks are expressed by using “links” to connect them



Source: BPEL4WS Version 1.1 Specification

Fault e Compensazione



Composizione

- Richiede:
  - Specifica del servizio che si vuole ottenere
    - Requisiti funzionali
    - Requisiti non funzionali
  - Individuazione e selezione dei servizi componenti
  - Definizione della logica del processo di composizione (flusso)
  - Deployment ed esecuzione
  - Validazione del servizio composto

Due fasi

- Composizione “logica”
- Composizione “fisica”
- La prima fase genera un grafo che descrive il flusso garantendo che i requisiti funzionali siano verificati dai servizi selezionati
- La seconda fase vede la trasposizione di questo grafo in un wf espresso mediante BPEL (o altro linguaggio)

Composizione automatica. Problematiche.

- Necessità di strumenti automatici per la composizione
- Come esprimo la specifica del servizio desiderato?
- Come trovo e seleziono i servizi?
- Come assicuro che i requisiti funzionali siano rispettati?
- E i requisiti non funzionali?
- Come assicuro che i servizi siano compatibili in termini di Input/Output?
- Come valido il workflow?

Alcune soluzioni…

- Synthy (IBM)

http://domino.research.ibm.com/comm/research_people.nsf/pages/biplav.Synthy.html

- METEOR-S

http://lsdis.cs.uga.edu/projects/meteor-s/

- Self-Serv

Link utili

- OASIS Web Services Business Process Execution Language (WSBPEL) TC
- http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
- Motori BPEL
  - ActiveBPEL
  - http://www.activebpel.org/docs/samples.html
  - Apache ODE: a WS-BPEL compliant web services orchestration engine
  - http://www.infoq.com/news/2008/07/ApacheODE
  - http://ode.apache.org/getting-ode.html
  - Jopera (anche Jopera per Eclipse)
  - http://www.jopera.org/node/208 (estensione BPEL verso REST)
- Eclipse e BPEL
- http://www.eclipse.org/bpel/
  - BPEL designer
  - http://www.eclipse.org/bpel/downloads.php

A Sample – LoanApproval.bepl

```
<process name="loanApprovalProcess"
        targetNamespace="http://acme.com/loanprocessing"
        suppressJoinFailure="yes"
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        xmlns:lns="http://loans.org/wsdl/loan-approval"
        xmlns:loandef="http://tempuri.org/services/loandefinitions"
        xmlns:asns="http://tempuri.org/services/loanassessor"
        xmlns:apns="http://tempuri.org/services/loanapprover">

  <variables>
    <variable name="request"
              messageType="loandef:creditInformationMessage"/>
    <variable name="riskAssessment"
              messageType="asns:riskAssessmentMessage"/>
    <variable name="approvalInfo"
              messageType="apns:approvalMessage"/>
    <variable name="error"
              messageType="loandef:loanRequestErrorMessage"/>
  </variables>
```

```
<partnerLinks>
   <partnerLink name="customer"
         partnerLinkType="lns:loanApprovalLinkType"
         myRole="approver"/>
   <partnerLink name="approver"
         partnerLinkType="lns:loanApprovalLinkType"
         partnerRole="approver"/>
   <partnerLink name="assessor"
         partnerLinkType="lns:riskAssessmentLinkType"
         partnerRole="assessor"/>
 </partnerLinks>
 <faultHandlers>
   <catch faultName="lns:loanProcessFault"
        faultVariable="error">
    <reply partnerLink="customer"
         portType="apns:loanApprovalPT"
         operation="approve"
         variable="error"
         faultName="invalidRequest"/>
   </catch>
 </faultHandlers>
```

```
<flow>
    <links>
      <link name="receive-to-assess"/>
      <link name="receive-to-approval"/>
      <link name="approval-to-reply"/>
      <link name="assess-to-setMessage"/>
      <link name="setMessage-to-reply"/>
      <link name="assess-to-approval"/>
    </links>
<receive name="receive1" partnerLink="customer"
            portType="apns:loanApprovalPT"
            operation="approve" variable="request"
            createInstance="yes">
      <source linkName="receive-to-assess"
             transitionCondition="bpws:getVariableData('request',
'amount')&lt;10000"/>
      <source linkName="receive-to-approval"
             transitionCondition="bpws:getVariableData('request',
'amount')&gt;=10000"/>
    </receive>
```

```
<invoke name="invokeAssessor" partnerLink="assessor"
        portType="asns:riskAssessmentPT"
        operation="check"
        inputVariable="request"
        outputVariable="riskAssessment">
    <target linkName="receive-to-assess"/>
    <source linkName="assess-to-setMessage"
            transitionCondition="bpws:getVariableData('riskAssessment',
'risk')='low'"/>
    <source linkName="assess-to-approval"
            transitionCondition="bpws:getVariableData('riskAssessment',
'risk')!='low'"/>
  </invoke>

  <assign name="assign">
    <target linkName="assess-to-setMessage"/>
    <source linkName="setMessage-to-reply"/>
    <copy>
      <from expression="'yes'"/>
      <to variable="approvalInfo" part="accept"/>
    </copy>
  </assign>
```

```
<invoke name="invokeapprover"
            partnerLink="approver" portType="apns:loanApprovalPT"
            operation="approve"
            inputVariable="request"
            outputVariable="approvalInfo">
        <target linkName="receive-to-approval"/>
        <target linkName="assess-to-approval"/>
        <source linkName="approval-to-reply" />
      </invoke>

      <reply name="reply" partnerLink="customer"
portType="apns:loanApprovalPT"
            operation="approve" variable="approvalInfo">
        <target linkName="setMessage-to-reply"/>
        <target linkName="approval-to-reply"/>
      </reply>
    </flow>
</process>
```