

Software Engineering and Service-Oriented Systems

– SOC and Service Orchestration –

Francesco Tiezzi



IMT - Institutions, Markets, Technologies

Institute for Advanced Studies Lucca

Lucca, Italy - September, 2013

Setting the scene

- This course focusses on **Service-Oriented Systems (SOSs)**
- We will introduce the notion of:
 - ▶ **Service-Oriented Computing** as a paradigm for developing SOSs
 - ▶ **Service** as a basic block for building SOSs

Setting the scene

- This course focusses on **Service-Oriented Systems (SOSs)**
- We will introduce the notion of:
 - ▶ **Service-Oriented Computing** as a paradigm for developing SOSs
 - ▶ **Service** as a basic block for building SOSs

Service-Oriented Computing (SOC)

- A compute paradigm for distributed and e-business computing
- Aims at enabling developers to build networks of integrated and collaborative applications, regardless of
 - ▶ the platform where the applications run (e.g., the operating system)
 - ▶ the programming language used to develop themthrough the use of *loosely coupled, platform-independent, reusable* components (called **services**)
- A modern attempt to cope with old problems related to information interchange, software integration, and B2B
 - ▶ Finds its origin in object-oriented and component-based software development
- **Service-Oriented Architecture (SOA)**: an architectural style to realize SOC

Web Services Composition

- XML-based technologies like WSDL, UDDI and SOAP
 - ▶ permit describing, locating and invoking web services
 - ▶ are usually sufficient for simple B2B application integration needs
 - Creation of complex B2B applications and automated integration of business processes across enterprises require managing such features as
 - ▶ asynchronous interactions
 - ▶ concurrency
 - ▶ workflow coordination
 - ▶ business transaction activities and exceptions
- ... which the above mentioned standards do not deal with

- This raises the need for designing and employing

Web Services composition languages

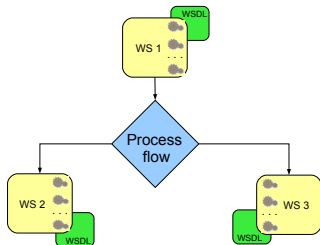
an additional layer on top of the Web Services protocol stack

Orchestration vs. Choreography

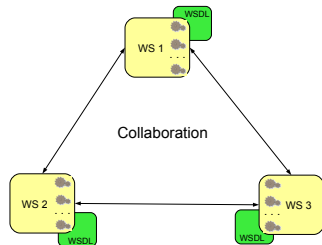
- Service composition permits building complex services out of simpler ones and is still an open challenge
- There are two main views of web services composition
 - ▶ Orchestration (= Executable Business Process)
 - ★ Description of web services interactions, including the business logic and execution order of the interactions
 - ★ Interactions may span applications and/or organizations, and result in a long-lived, transactional process
 - ★ The process is always controlled from the perspective of *one* of the business parties
 - ★ Main enabling technology: WS-BPEL (OASIS standard)
 - ▶ Choreography (= Multi-party Collaboration)
 - ★ Description of the externally observable message exchanges among *multiple* web services
 - ★ No party truly 'owns' the conversation
 - ★ More collaborative in nature: each party involved in the process describes the role it plays in each interaction
 - ★ Main enabling technology: WS-CDL (W3C Recommendation)

Orchestration vs. Choreography

- Service composition permits building complex services out of simpler ones and is still an open challenge
- There are two main views of web services composition



Orchestration

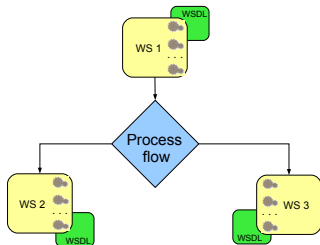


Choreography

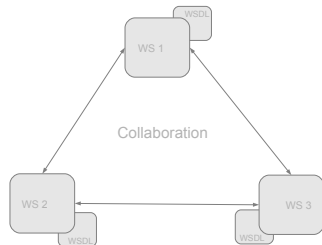
We focus on web service orchestration

Orchestration vs. Choreography

- Service composition permits building complex services out of simpler ones and is still an open challenge
- There are two main views of web services composition



Orchestration



Choreography

We focus on web service orchestration

Services & Business processes

- A process *orchestrating* web services is called **business process** i.e. an active entity that invokes available services according to a given set of rules to meet some business requirements
- A business process specifies
 - ▶ the potential execution order of operations originating from a collection of Web Services
 - ▶ the shared data passed between these services
 - ▶ the trading partners that are involved in the joint process
 - ▶ their roles with respect to the process
 - ▶ joint exception handling conditions for the collection of Web Servicesand other factors that may influence how Web Services or organizations participate in a process
- Web service orchestration thus permits to program complex inter-enterprise workflow tasks and business transactions

Services & Business processes

- A process *orchestrating* web services is called **business process** i.e. an active entity that invokes available services according to a given set of rules to meet some business requirements
- A business process specifies
 - ▶ the potential execution order of operations originating from a collection of Web Services
 - ▶ the shared data passed between these services
 - ▶ the trading partners that are involved in the joint process
 - ▶ their roles with respect to the process
 - ▶ joint exception handling conditions for the collection of Web Services

and other factors that may influence how Web Services or organizations participate in a process

- Web service orchestration thus permits to program complex inter-enterprise workflow tasks and business transactions

Services & Business processes

- A process *orchestrating* web services is called **business process** i.e. an active entity that invokes available services according to a given set of rules to meet some business requirements
- A business process specifies
 - ▶ the potential execution order of operations originating from a collection of Web Services
 - ▶ the shared data passed between these services
 - ▶ the trading partners that are involved in the joint process
 - ▶ their roles with respect to the process
 - ▶ joint exception handling conditions for the collection of Web Services

and other factors that may influence how Web Services or organizations participate in a process

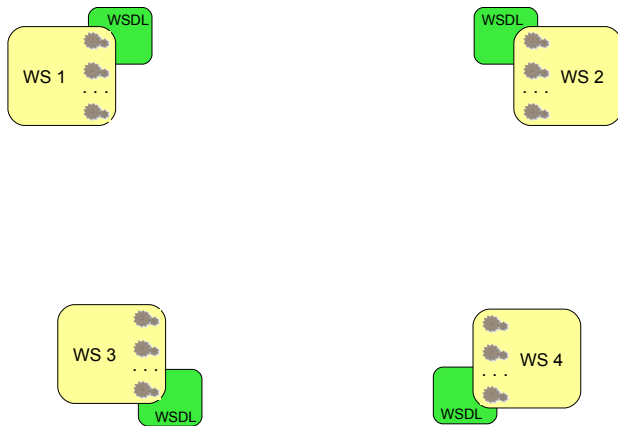
- Web service orchestration thus permits to program complex inter-enterprise workflow tasks and business transactions

Services & Business processes

- Business processes can be exposed as web services

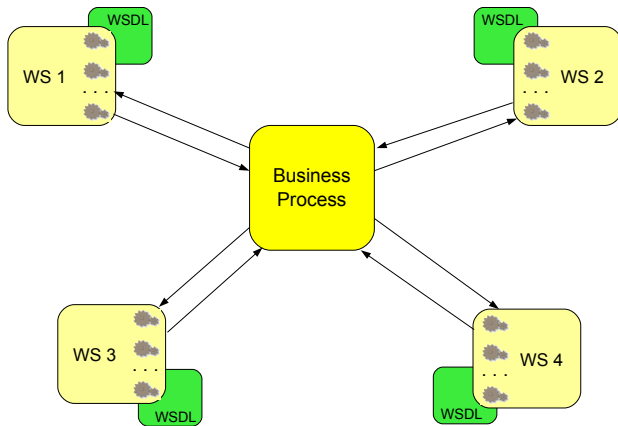
Services & Business processes

- Business processes can be exposed as web services



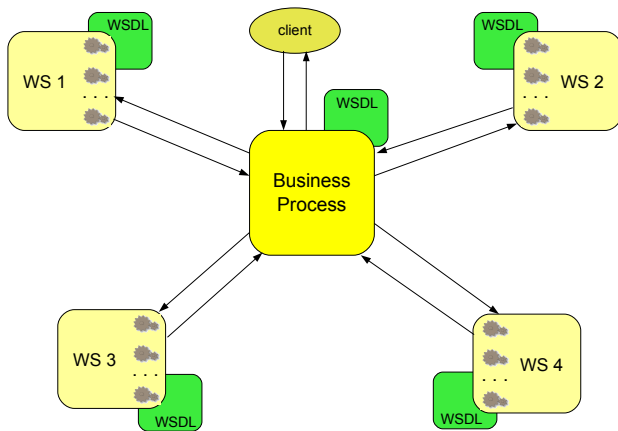
Services & Business processes

- Business processes can be exposed as web services



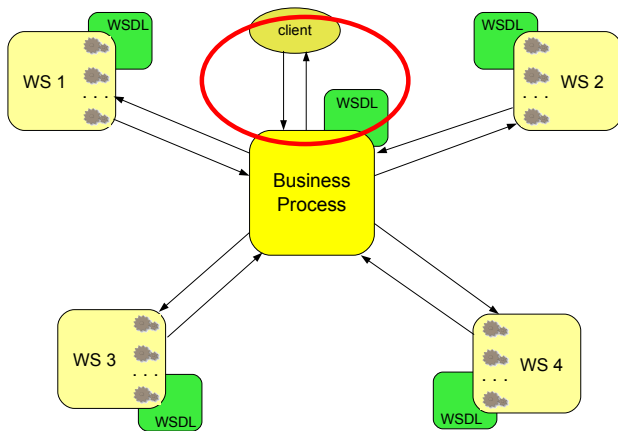
Services & Business processes

- Business processes can be exposed as web services



Services & Business processes

- Business processes can be exposed as web services

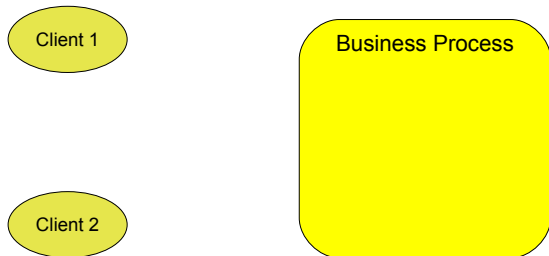


Instantiation & Message Correlation

- To serve clients' requests **service instances** are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)

Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)

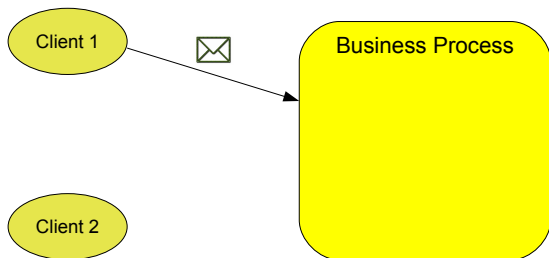


Message correlation

The message content permits identifying the proper target instance

Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)

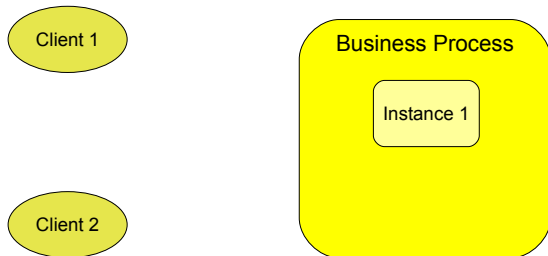


Message correlation

The message content permits identifying the proper target instance

Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)

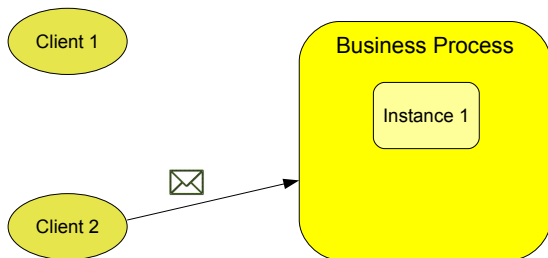


Message correlation

The message content permits identifying the proper target instance

Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)

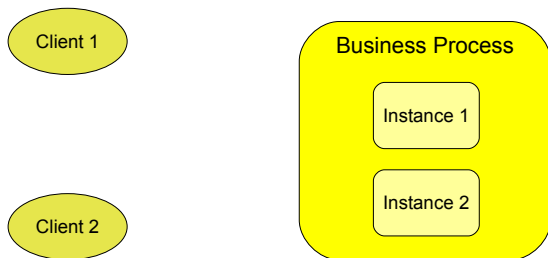


Message correlation

The message content permits identifying the proper target instance

Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)

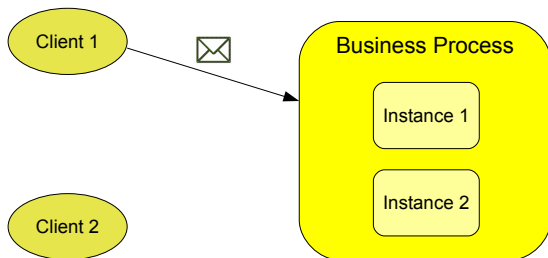


Message correlation

The message content permits identifying the proper target instance

Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)

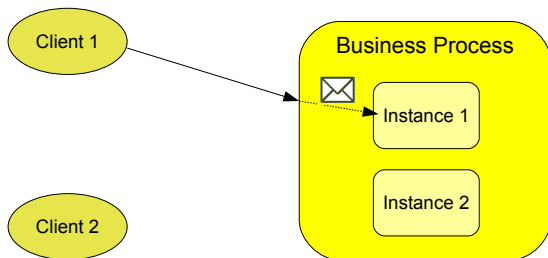


Message correlation

The message content permits identifying the proper target instance

Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)

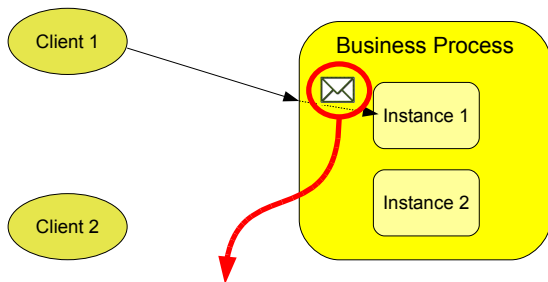


Message correlation

The message content permits identifying the proper target instance

Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)

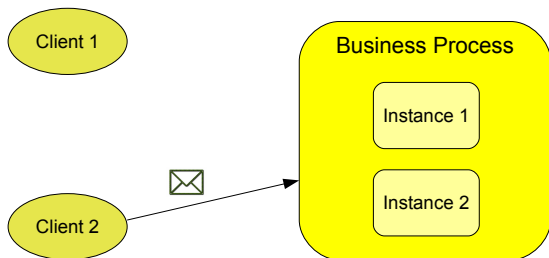


Message correlation

The message content permits identifying the proper target instance

Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)

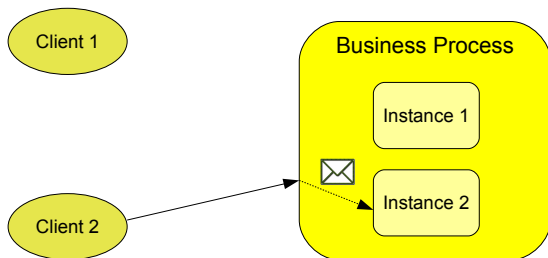


Message correlation

The message content permits identifying the proper target instance

Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)

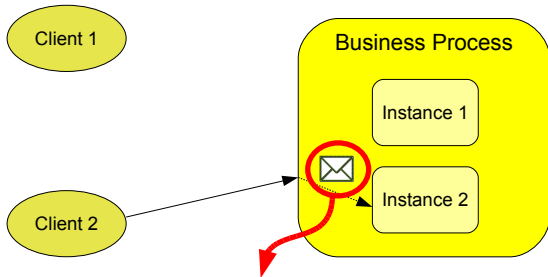


Message correlation

The message content permits identifying the proper target instance

Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)



Message correlation

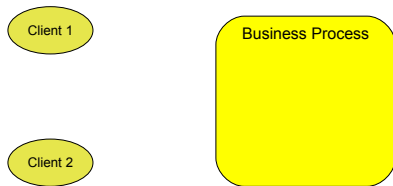
The message content permits identifying the proper target instance

Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)
- Message correlation permits
 - ▶ integrating *asynchronous* services, that take from a few minutes to some days to complete
 - ▶ tying messages together in order to build long-lived interactions
 - ▶ implementing statefull *multiparty* conversations

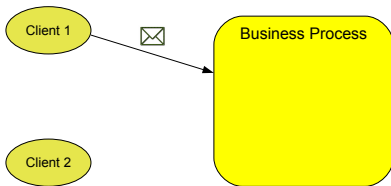
Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)
- Message correlation permits
 - ▶ integrating *asynchronous* services, that take from a few minutes to some days to complete
 - ▶ tying messages together in order to build long-lived interactions
 - ▶ implementing statefull *multiparty* conversations



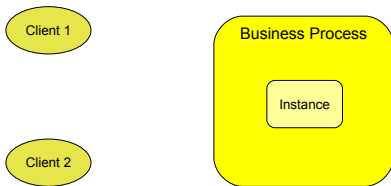
Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)
- Message correlation permits
 - ▶ integrating *asynchronous* services, that take from a few minutes to some days to complete
 - ▶ tying messages together in order to build long-lived interactions
 - ▶ implementing statefull *multiparty* conversations



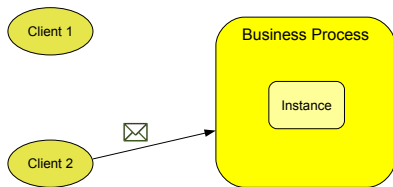
Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)
- Message correlation permits
 - ▶ integrating *asynchronous* services, that take from a few minutes to some days to complete
 - ▶ tying messages together in order to build long-lived interactions
 - ▶ implementing statefull *multiparty* conversations



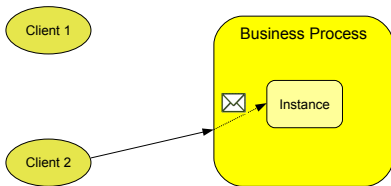
Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)
- Message correlation permits
 - ▶ integrating *asynchronous* services, that take from a few minutes to some days to complete
 - ▶ tying messages together in order to build long-lived interactions
 - ▶ implementing statefull *multiparty* conversations



Instantiation & Message Correlation

- To serve clients' requests service instances are created
- When a message arrives, it must be delivered:
 - ▶ either to a new instance (new conversation)
 - ▶ or to an existing instance (old conversation)
- Message correlation permits
 - ▶ integrating *asynchronous* services, that take from a few minutes to some days to complete
 - ▶ tying messages together in order to build long-lived interactions
 - ▶ implementing statefull *multiparty* conversations




WS-BPEL

Web Services Composition Languages

- Different organizations have been involved and are presently working on the design of languages for specifying business processes
- Two WS-BPEL's forerunners are
 - ▶ Microsoft's XLANG
a block-structured language with basic control flow structures
 - ★ e.g. sequence, switch (conditional), while (looping), all (parallel) and pick (choice based on timing or external events)
 - ▶ IBM's WSFL (Web Services Flow Language)
a language for specifying arbitrary directed acyclic graphs
- Afterwards, the two proposals have been combined into a new language, WS-BPEL, that has been submitted to OASIS for standardization also by BEA systems, SAP and Siebel Systems

WS-BPEL

- Web Services Business Process Execution Language Version 2.0
- Is an **OASIS**  standard (11 April 2007)
- Is the most widespread language for orchestration of Web Services
- Has an XML-based syntax and relies on the following XML-based specifications
 - ▶ WSDL for interfaces
 - ▶ XML Schema for types
 - ▶ XPath for expressions

WS-BPEL: engines

- Three of the most known freely available WS-BPEL engines



Oracle BPEL Process Manager

<http://www.oracle.com/technology/bpel>



ActiveBPEL







<http://www.activevos.com>



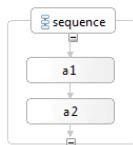
Apache ODE

<http://ode.apache.org>

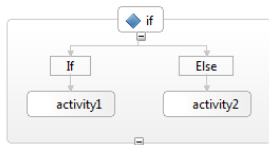
WS-BPEL: basic activities

-  empty to do nothing
-  invoke to invoke an operation offered by a (partner) service
 - ▶ partner services are identified by *partner links* defining the shape of peer-to-peer conversational relationships
-  receive to wait for a request to arrive
-  reply to send a message in reply to a received request
-  assign to update the values of variables with new data
-  wait to wait for a given time period or until a certain point in time has been reached

WS-BPEL: control flow activities



to perform a collection of activities in sequential order

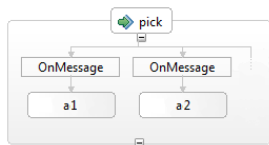


to select exactly one activity for execution from two alternatives

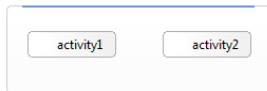


to repeat an activity as long as a given condition is true

WS-BPEL: control flow activities



to wait for one of several possible requests to arrive
or for a time-out to occur



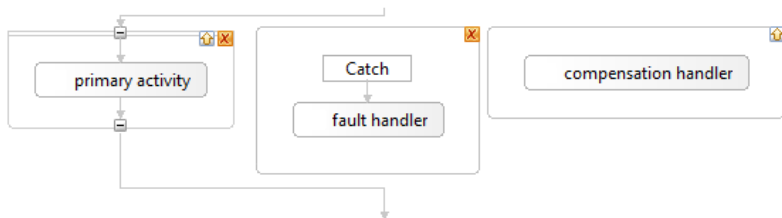
to concurrently perform a set of activities (flow activity)

WS-BPEL: fault and compensation






- *Fault handling*: similar to exception handling of 'classic' programming languages
- *Compensation*: execution of specific activities (attempting) to reverse the effects of previously executed activities

WS-BPEL: fault and compensation

- *Fault handling*: similar to exception handling of 'classic' programming languages
- *Compensation*: execution of specific activities (attempting) to reverse the effects of previously executed activities
- *Scope activity*: groups a primary activity together with fault handling activities and a compensation handling activity



WS-BPEL: fault and compensation

-  `exit` to immediately terminate an instance
-  `throw` to generate a fault from inside an instance
-  `rethrow` to rethrow the fault that was originally caught by the immediately enclosing fault handler
-  `compensate` to start compensation on all inner scopes that have already completed successfully, in the *reverse order* of completion
-  `compensateScope` to start compensation of a specified inner scope that has already completed successfully

WS-BPEL: other aspects

- Termination and event handlers within scope activities
- Synchronization dependencies within flow activities
- repeatUntil and forEach activities

WS-BPEL at work

A shipping scenario



Back-end service



Client
service



Shipping service

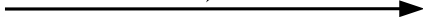
A shipping scenario



Back-end service



Client service



Shipping service

A shipping scenario

- An order is composed of a number of items (e.g. 7)
- Two types of shipments:
 - items are held and shipped together
 - items are shipped piecemeal until the order is fulfilled



Back-end service



Client service



Shipping service

A shipping scenario



Client
service



Back-end service



Shipping service

A shipping scenario



Client
service



Back-end service



Shipping service

A shipping scenario



Back-end service



Client service



Shipping service

A shipping scenario



Client service



Back-end service



Shipping service

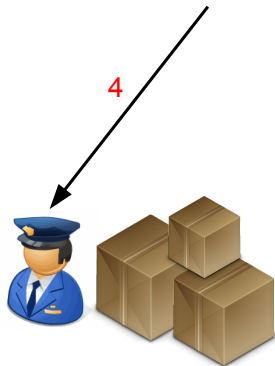
A shipping scenario



Client service



Back-end service



Shipping service

A shipping scenario



Back-end service

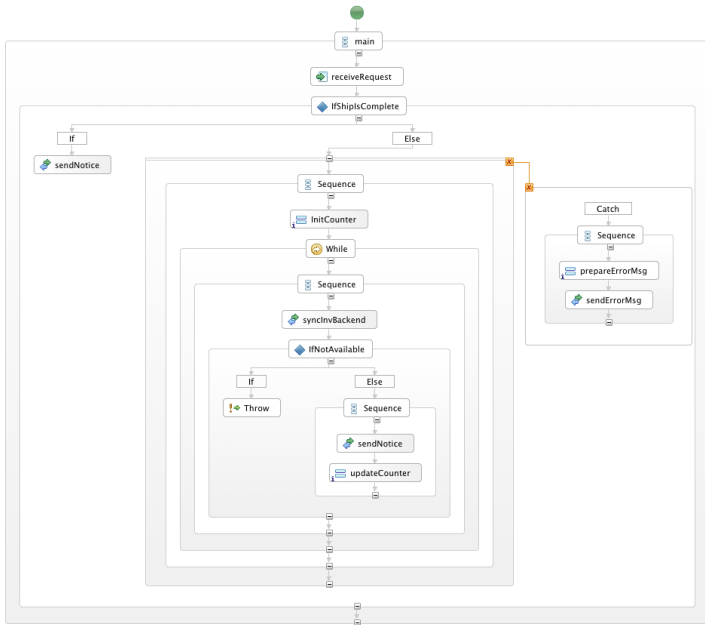


Client service



Shipping service

A shipping scenario in BPEL



References

Some references

- <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf>
- ...