Lesson 11 – Programming language

Service Oriented Architectures Security

Module 1 - Basic technologies

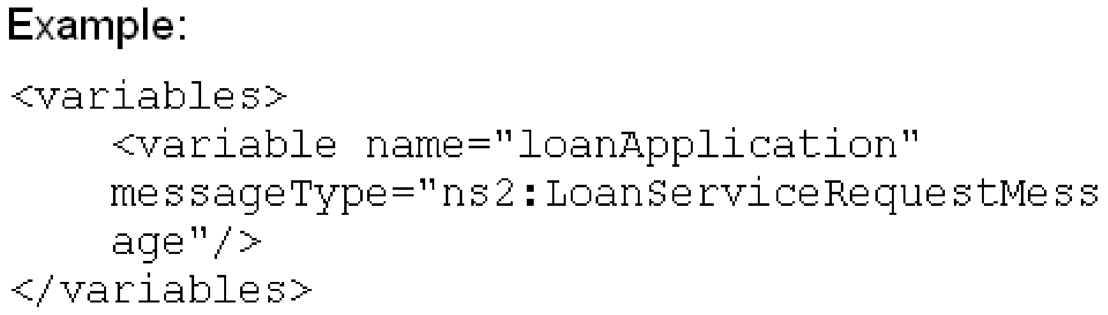
Unit 5 – BPEL

Ernesto Damiani

Università di Milano

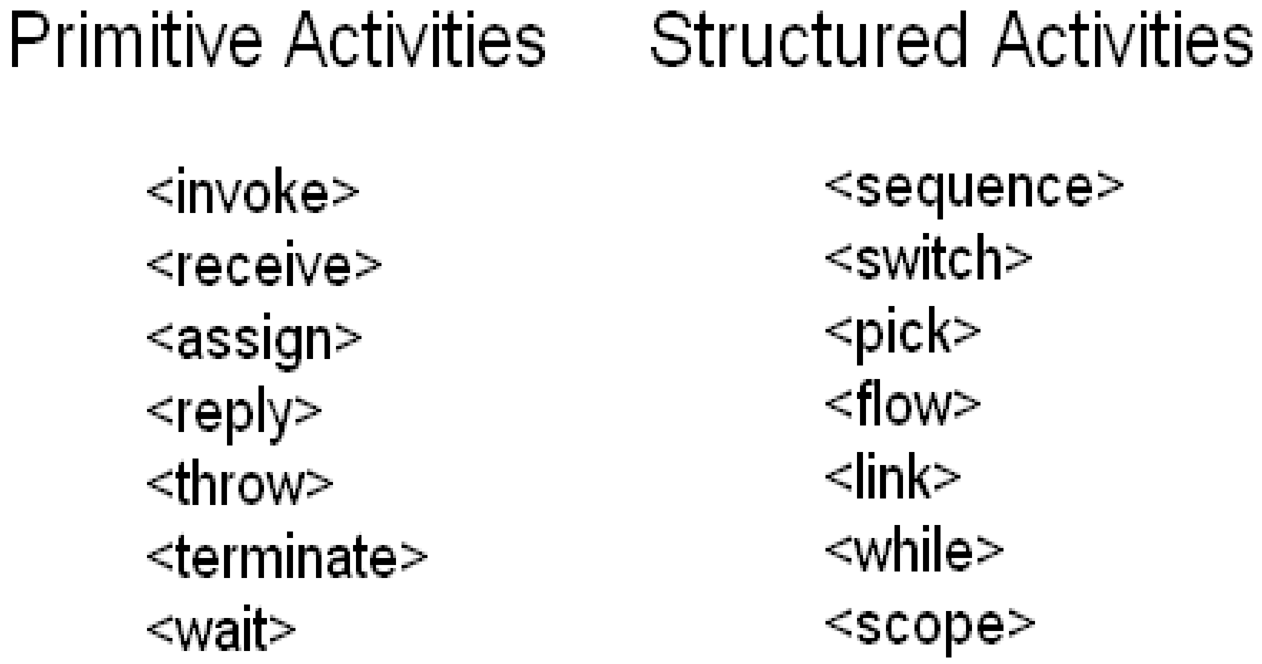
Variables

- Used to store, reformat and transform messages
- Required to send and receive messages
- Each variable has a Type


Example:

```
<variables>
    <variable name="loanApplication"
    messageType="ns2:LoanServiceRequestMess
    age"/>
</variables>
```

Activities (1)



Primitive Activities	Structured Activities
<invoke>	<sequence>
<receive>	<switch>
<assign>	<pick>
<reply>	<flow>
<throw>	<link>
<terminate>	<while>
<wait>	<scope>

Activities (2)

- <invoke>
  - Invoke a service synchronously
    - Ex.: Invoke Credit Service
- <receive>
  - Waits for the incoming message, either to start the process or for a callback
    - Ex.: Wait for a message from United Loan
- <reply>
  - Return response for synchronous process, relate to initial <receive>
- <assign>
  - Copy data between variables, expressions and endpoint references
  - Used with XPath expressions and XSLT engine
    - Ex.: Copy Load Application from input payload to United Loan input

Scope

- Scopes can be used to divide the business process into organized parts
- A <scope> is an execution context for the contained activities, and a process is, itself, a <scope>
- A <scope> defines local variables and can catch and handle either specific faults or all faults that occur with it
  - Ex: GetCreditRating Scope – Invoke Credit Service and catch exceptions

Control flow (1)

- BPEL provides the usual branching and looping control flow constructs
- A <sequence> executes activities in serial order
- A <switch> executes at most one alternative based on expressions specified on child <case> elements with an optional <otherwise>
  - Ex: choose between United and Star Loan offers based on lower APR
- A <while> loops through activities while a variable's value is true

Control flow (2)

- BPEL provides a parallel control construct through the <flow> activity
  - Ex: Invoke United and Star Loan services in parallel
- More complex synchronization is achieved through "join" expressions composed of link statuses and boolean operations (&& and ||)

Partner Links

- Links to all parties that process interacts
- Links can be to Web Services
  - Ex: CreditService, UnitedLoanService, StarLoanService
- Links can be to other BPEL processes as well
- PartnerLinkTypes
  - Declares how parties interact and what each party offers

Fault handling

- Handle faults to enable completion of process using <faultHandlers>
- Use <catch> activity to handle specific faults
  - Ex: catch bad credit exception and terminate the process
- Use <catchAll> to handle all other faults

Event handling

- Message events
  - Useful to address wait for several messages
- Alarm events
  - Make process wait for a callback for a certain period of time
- <pick> activity
  - Process should wait the occurrence of one event in a set of events
    - Ex: Loan Flow could be changed to use <pick> activity that waits only 30 minutes for a Loan request

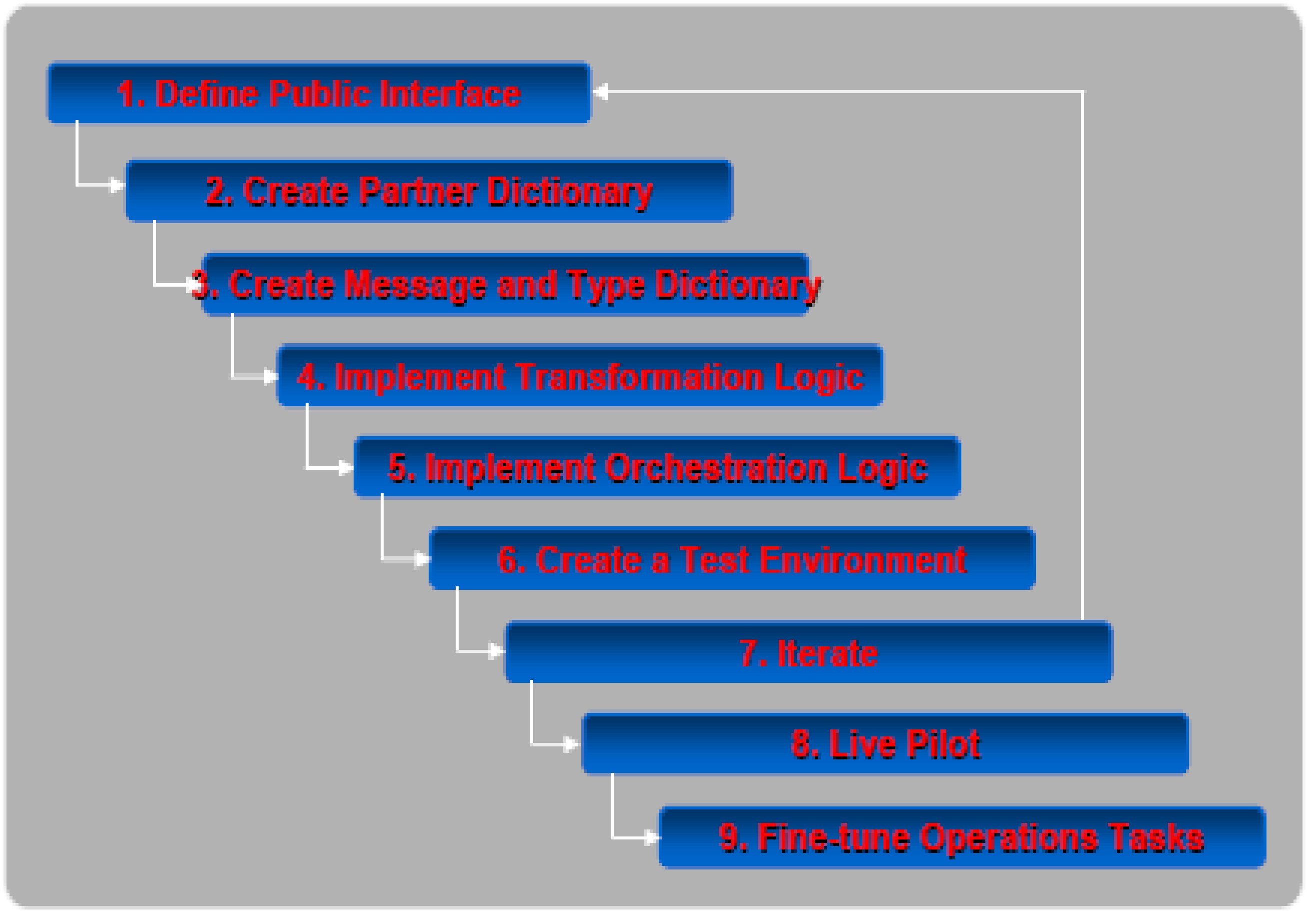
Correlation (1)

- BPEL correlates messages based on properties referenced in a <correlationSet>
- Multiple properties can be combined into a composite correlation key
- Properties are typed by XML Schema simple types and bound ("aliased") via Xpath expressions to locations in message parts

Correlation (2)

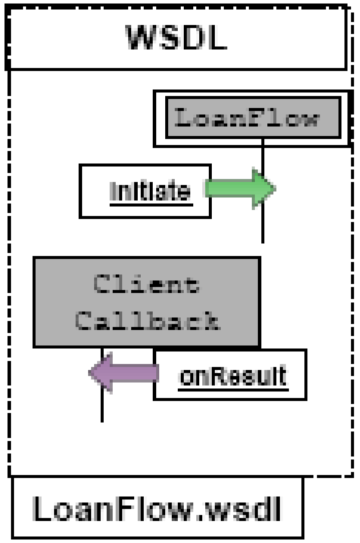
- Non-determinism
- A <pick> activity waits: for a message specified by an <onMessage> child element, where correlation allows a specific process instance to be addressed for an amount of time or until a time, specified with an <onAlarm> child element

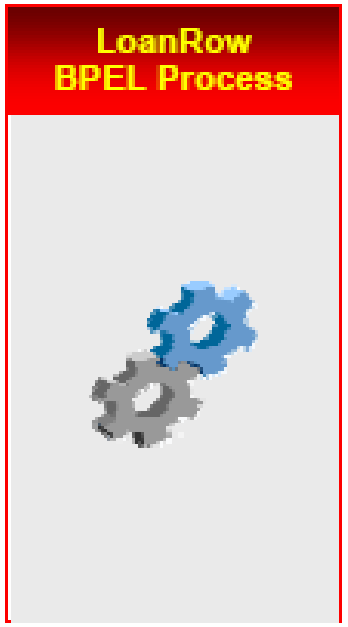
Steps to build business process


1. Define Public Interface
2. Create Partner Dictionary
3. Create Message and Type Dictionary
4. Implement Transformation Logic
5. Implement Orchestration Logic
6. Create a Test Environment
7. Iterate
8. Live Pilot
9. Fine-tune Operations Tasks

Step 1: define public interface

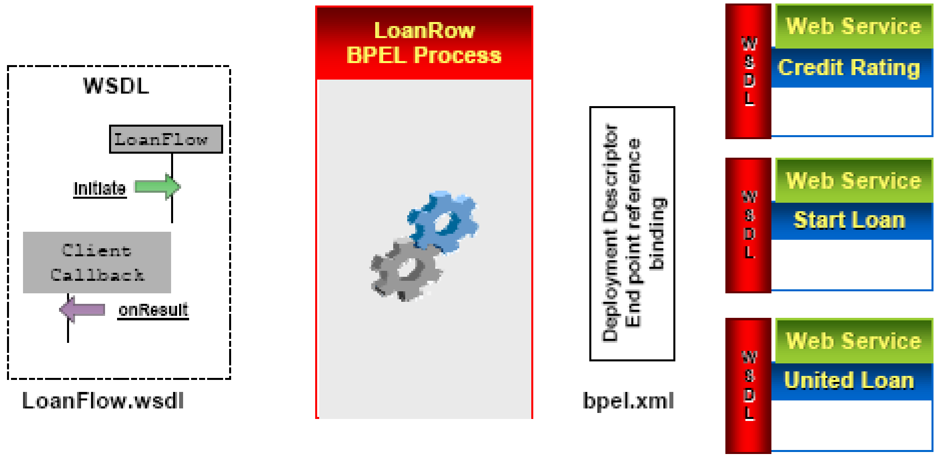
- Deliverables:
  - WSDL description of the interface of the implemented BPEL process





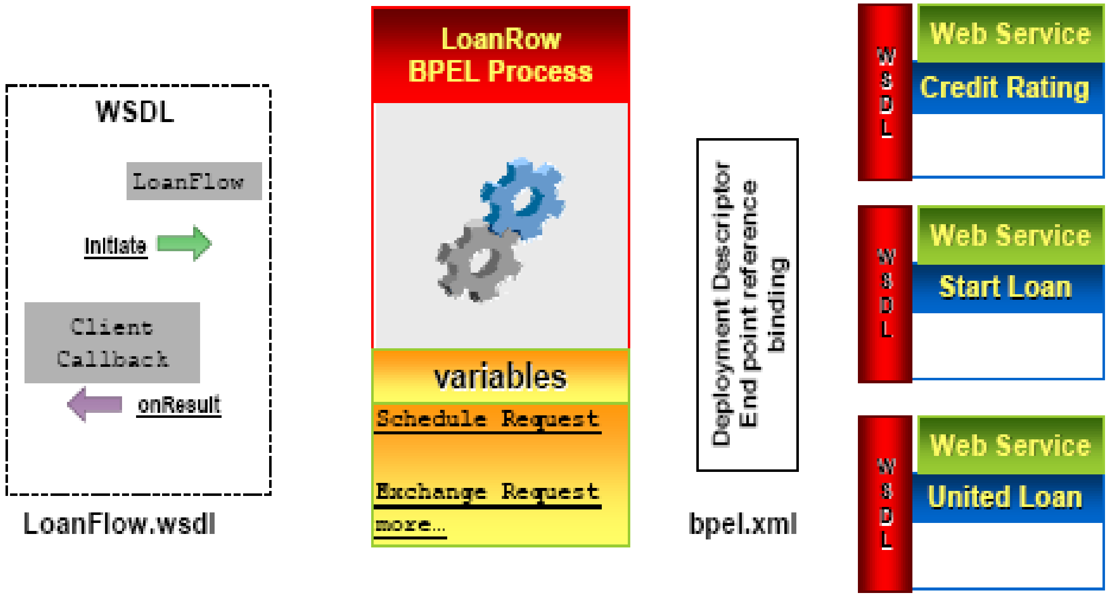
Step 2: create partner dictionary

- Deliverables:
  - List of the WSDL of the services that will be invoked as part of the BPEL Process
  - For each partner, document the order in which operations will be invoked (choreography)
  - Make sure that each use case describes both positive and negative use cases



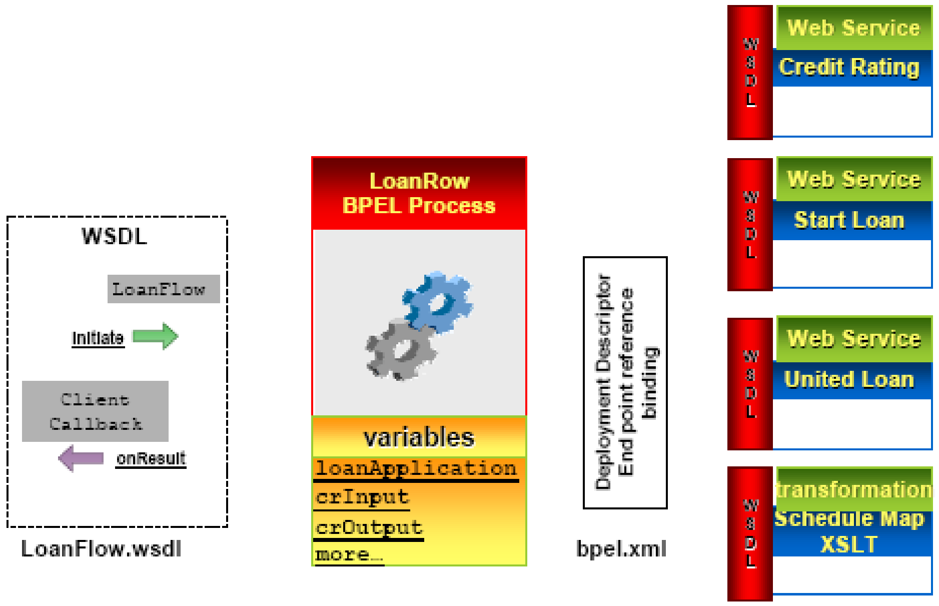
Step 3: create message and type dictionary

- Deliverables:
  - A set of XML Schema files that describe the type of the messages and XML documents used as part of the BPEL process



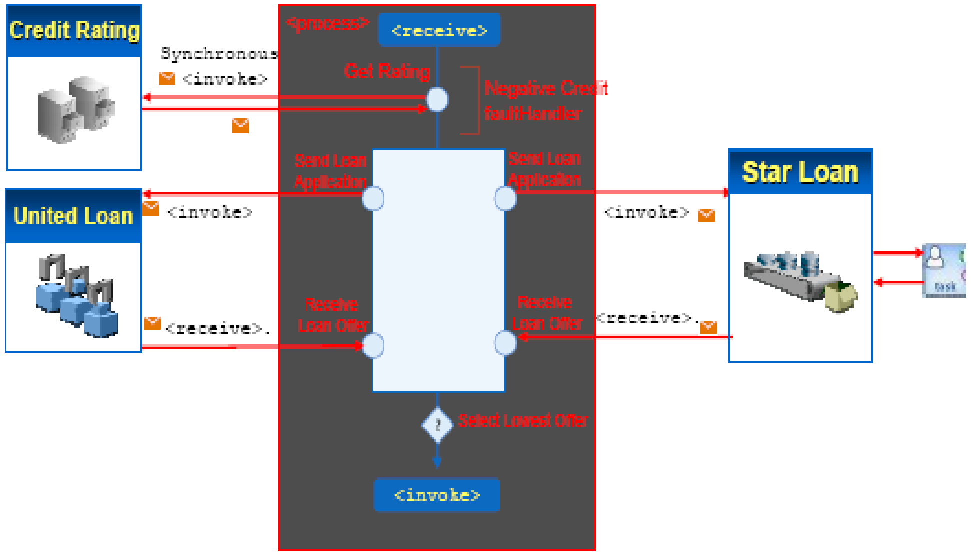
Step 4: transformation logic

- Deliverables:
  - A set of XSLT and XQuery files that encapsulate mapping information across the various types used in the BPEL process



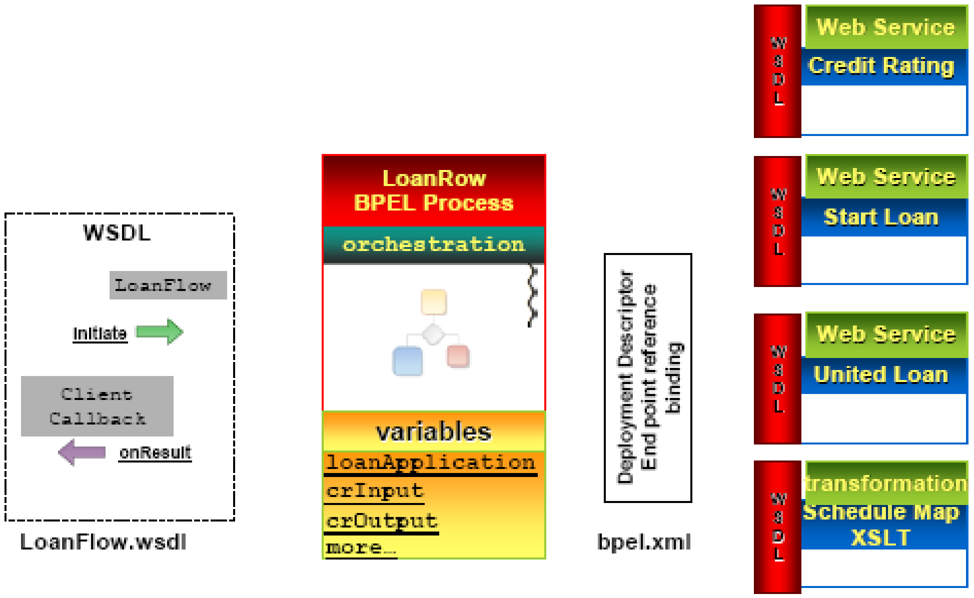
Step 5: orchestration logic

- Deliverables:
  - Implement the workflow that ties the interactions across partners into an end-to-end business process
  - Make sure that all exceptions and timeouts are managed properly



Step 6: iterate

- Deliverables:
  - Add incrementally new partners
  - Keep on improving exception management
  - Create automated test and regression framework

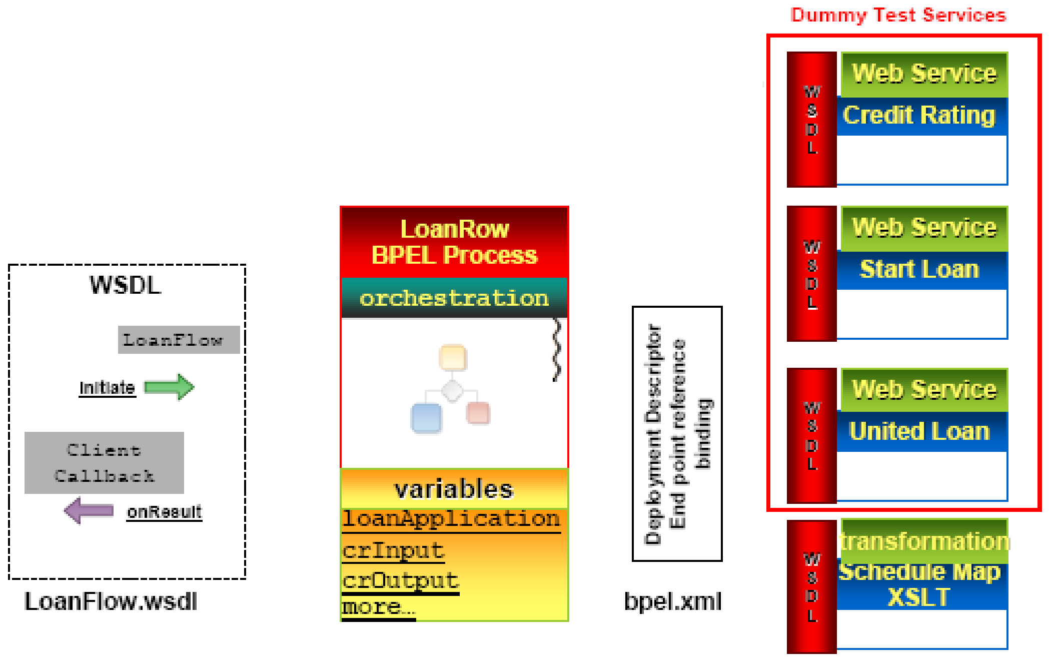


Step 7: create test environment (1)

- Deliverables:
  - Implement dummy test services for each end point (could be BPEL or your favorite Web services publishing technology)
  - Create test scenario for each positive and negative use cases

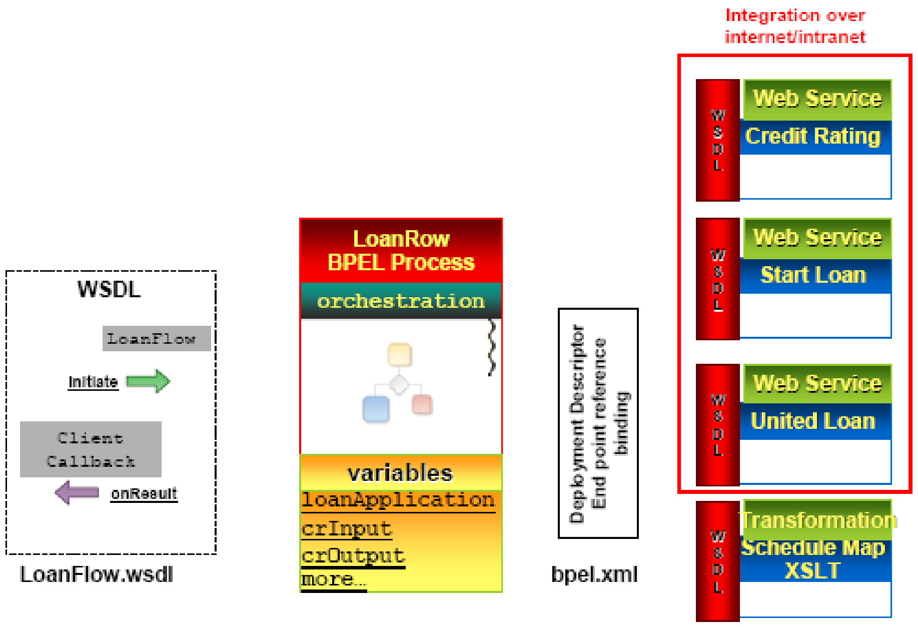
Step 7: create test environment (2)

- Crash test, longevity test (integrity/reliability)
- Performance test, stress test



Step 8: live pilot

- Deliverables:
  - Wire BPEL process to real end points
  - Run regression tests



Step 9: fine-tune operation tasks

- Deliverables:
  - Exception Management
  - Integration with Web Service Management Framework
  - Security
  - Archiving

Cross platform

Application Server
- Oracle Application Server
- WebLogic Server
- WebSphere
- JBoss

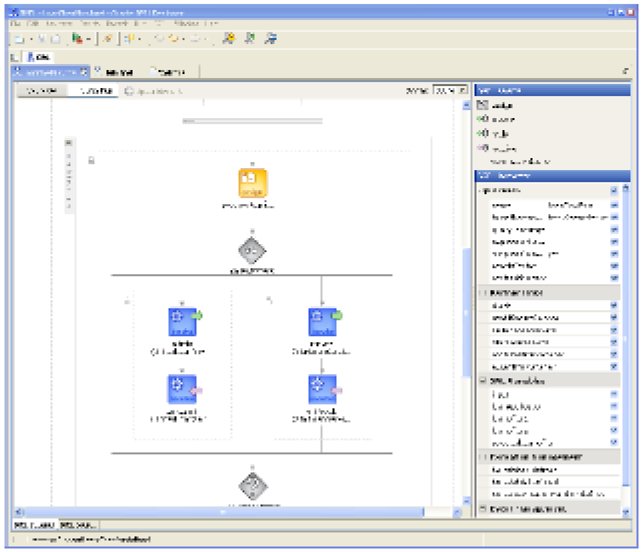
Database
- Oracle Database
- SQL Server
- Oracle Lite
- Sybase
- Pointbase

IDE
- JDeveloper
- Eclipse

Operating Systems
- Linux
- Window XP/2003
- Solaris
- HP UX
- zOS

BPEL Designer

- Native BPEL Support
- Drag-and-drop process modeler
- UDDI and WSIL service browser
- Visual XPATH editor
- One-click build and deploy

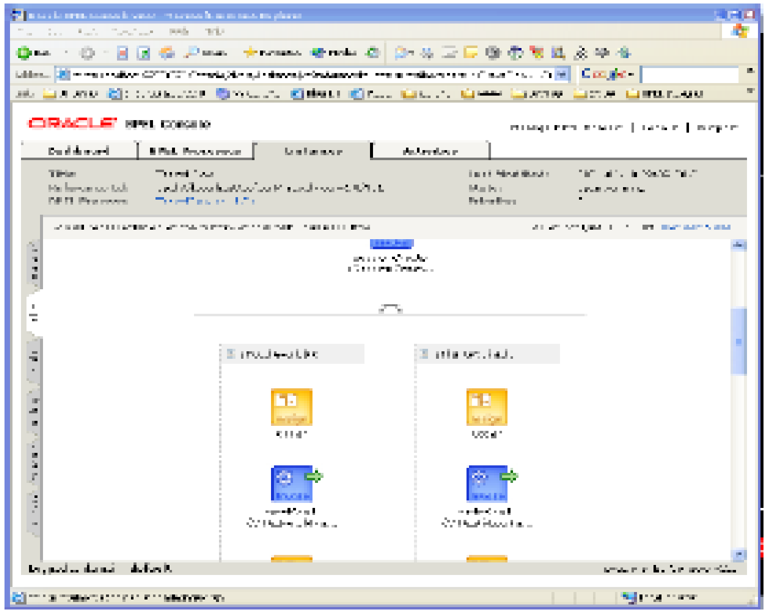


BPEL Console (1)

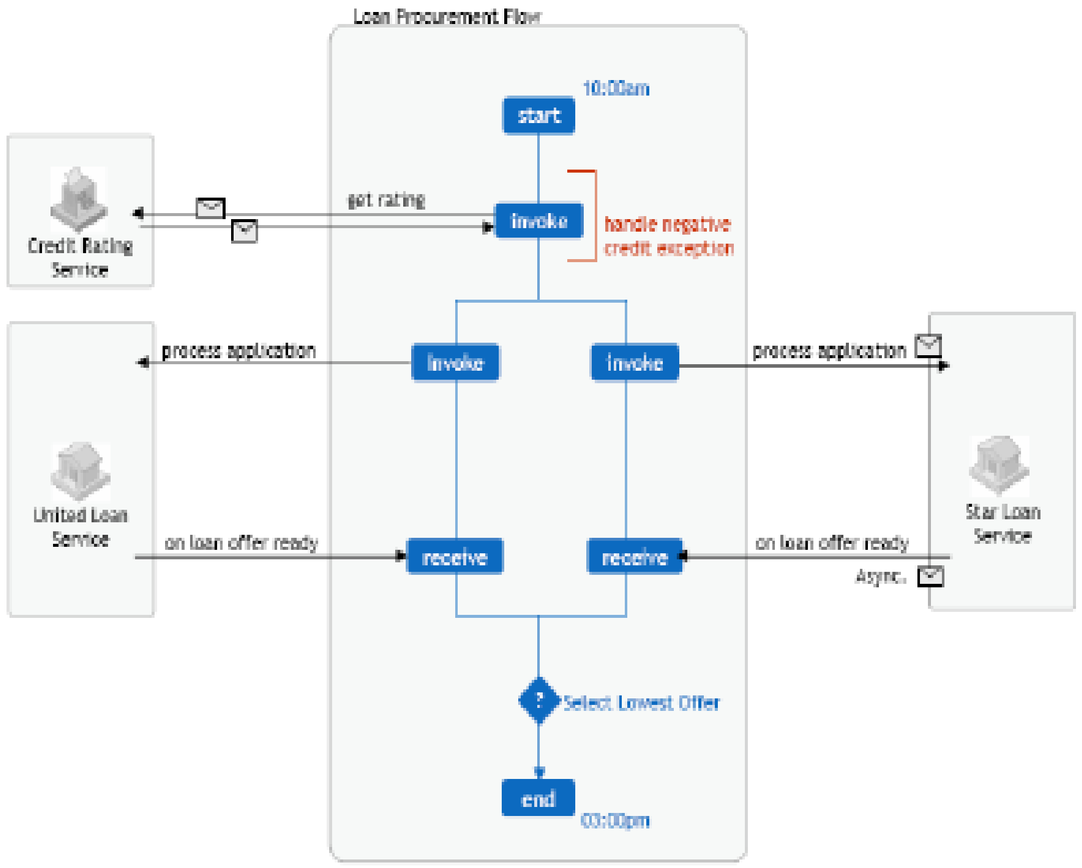
- Key features
  - Visual Monitoring
  - Auditing
  - BPEL Debugging
  - In-flight Instance

BPEL Console (2)

- Administration
- Performance Tuning
- Partitioning/Domains

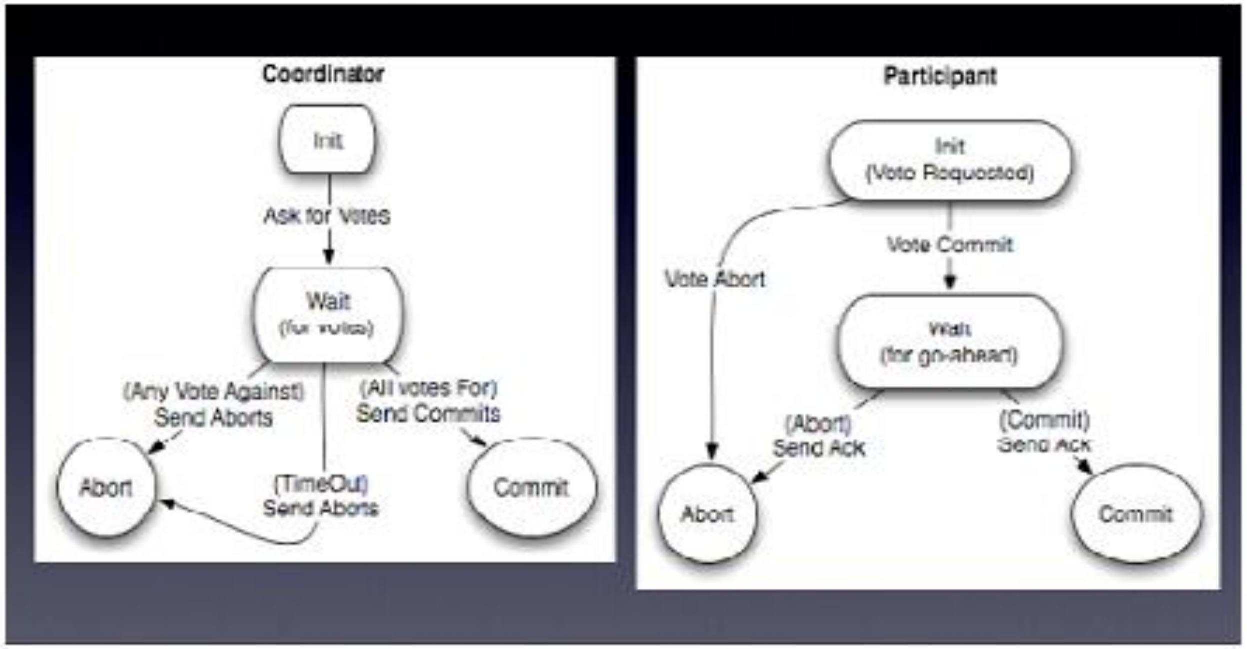


Example: loan service



- The problem
- e.g. in programming: x = x+1 and x = x+y in sequence/in parallel
- Databases, Distributed networking
- ACID
  - Atomic
  - Consistent
  - Isolated
  - Durable
  - Traditional transactions

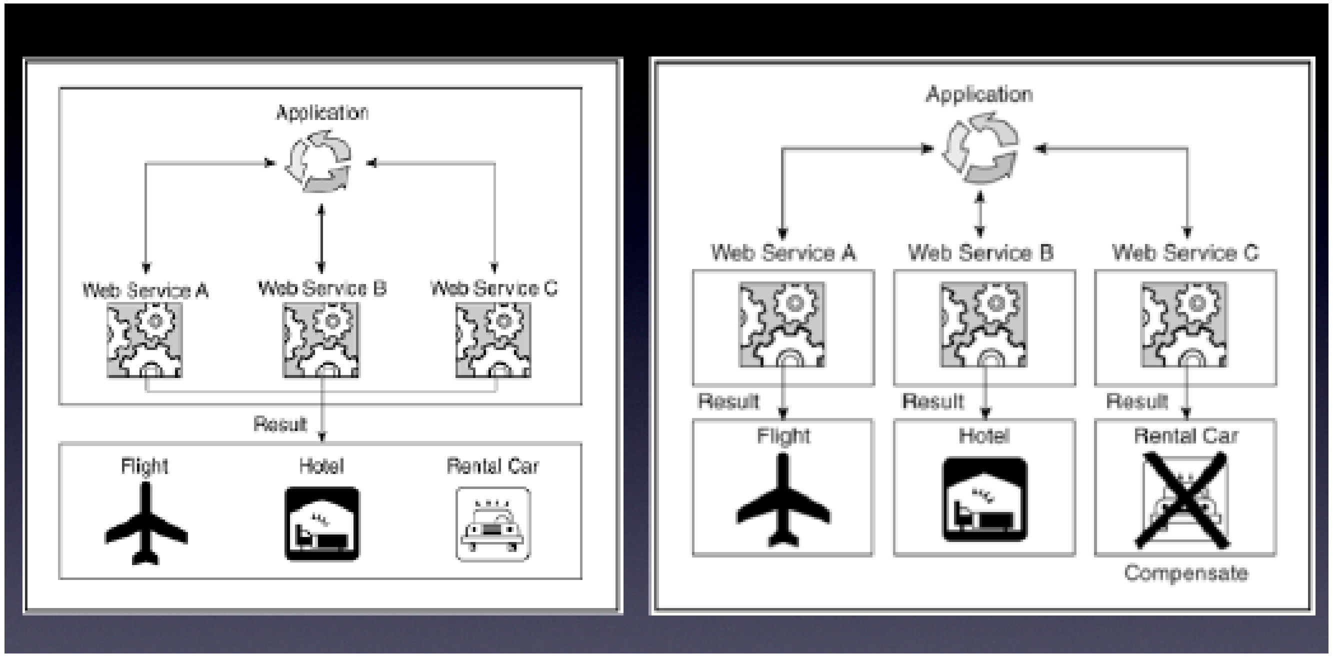
Two phase commit



Extended transactions

- Need for Extended Transactions in Web Services
- Rationale for Non-ACID requirements
  - Long duration, alternate failure handling, selected outcome inclusion, non-blocking across enterprises
- Web Services Protocols and Framework Standards
  - WS-Coordination
  - WS-Atomic Transaction
  - WS-Business Activity

Classic and basic transactions


Application
Web Service A
Web Service B
Web Service C
Result
Flight
Hotel
Rental Car
Application
Web Service A
Web Service B
Web Service C
Result
Result
Result
Flight
Hotel
Rental Car
Compensate