University of Pisa
MSc in Computer Engineering

Process-driven Information Systems

BUSINESS PROCESS MANAGEMENT SYSTEMS

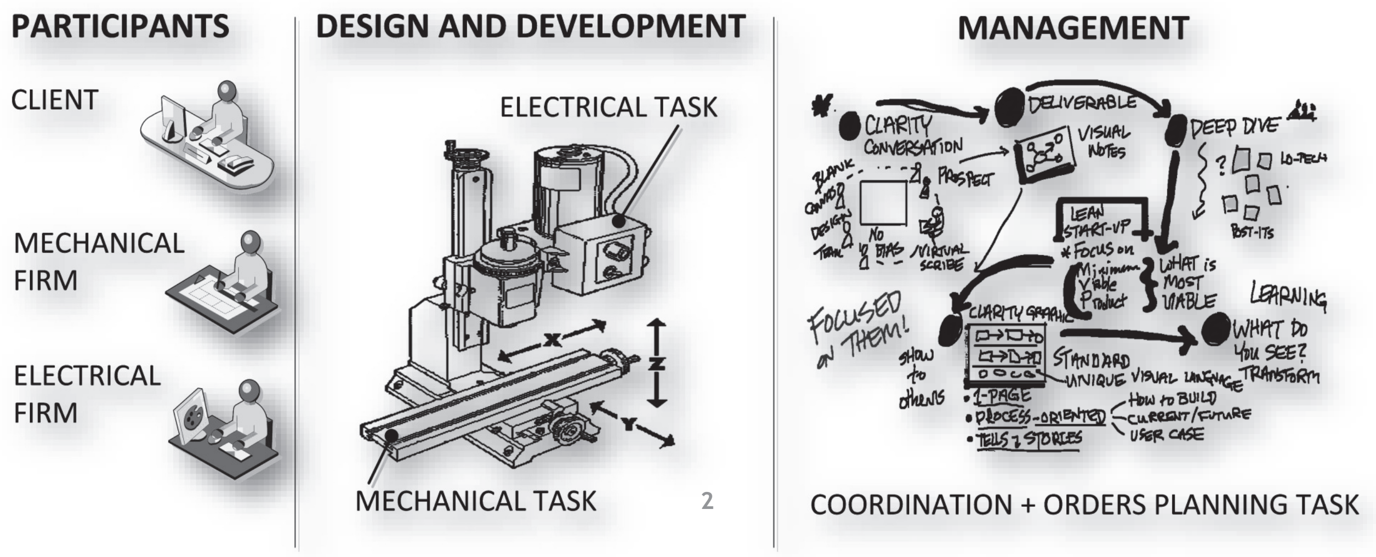
http://www.iet.unipi.it/m.cimino/wdis/

Mario G.C.A. Cimino
Department of Information Engineering

A scenario with workflow and business rules

A business collaboration on the order planning of a machinery

Pilot scenario. The participants involved in the business are (on the left in figure): the client, the mechanical and the electrical firms. Both design and development activities (in the middle), are made of two main tasks: a mechanical task and an electrical task, carried out by the two respective firms. Finally, the management activity (on the right) consists in the coordination of the participants and in the orders planning tasks. With regard to the orders planning, each company schedules tasks on the basis of its own private business rules.

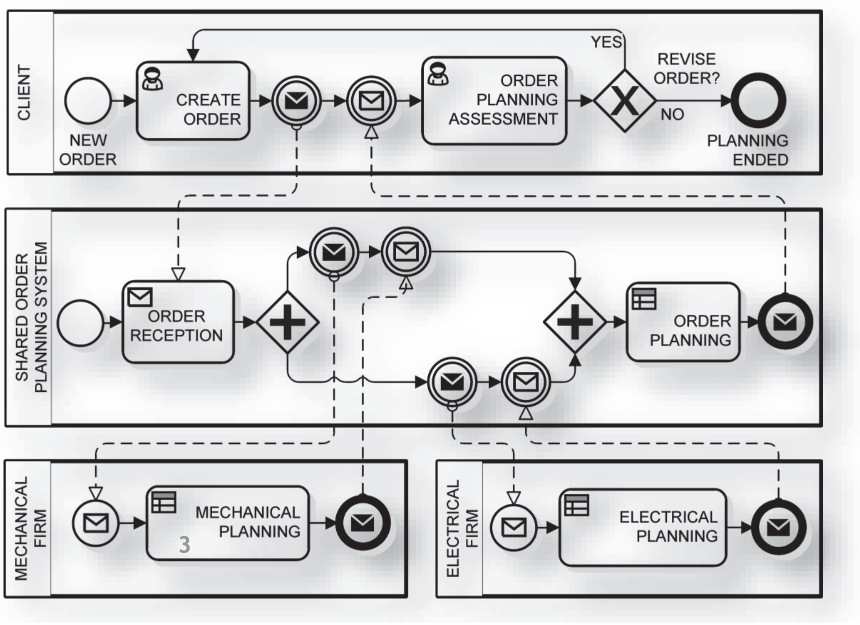


A scenario with workflow and business rules

A) BPMN process diagram of the collaborative planning of an order

A new order is created in a user task of the Client. A message with the order is sent from the client to the Shared Order Planning System. The Planning System splits the order into two parts, i.e. a mechanical and an electrical part, and sends them to the mechanical and electrical firms, respectively. Then, each firm performs its planning, represented as a business rule task. In a business rule task, one or more business rules are applied in order to produce a result or to make a decision, by means of a Business Rule Management System (BRMS) which is called by the process engine.

The BRMS then evaluates the rules that apply to the current situation. Each pool of a firm is supposed to be executed in a firm's private server, whereas the Planning System and the Client pools are supposed to be executed in a shared server. This way, the business rules of each firm are completely hidden to the Community.



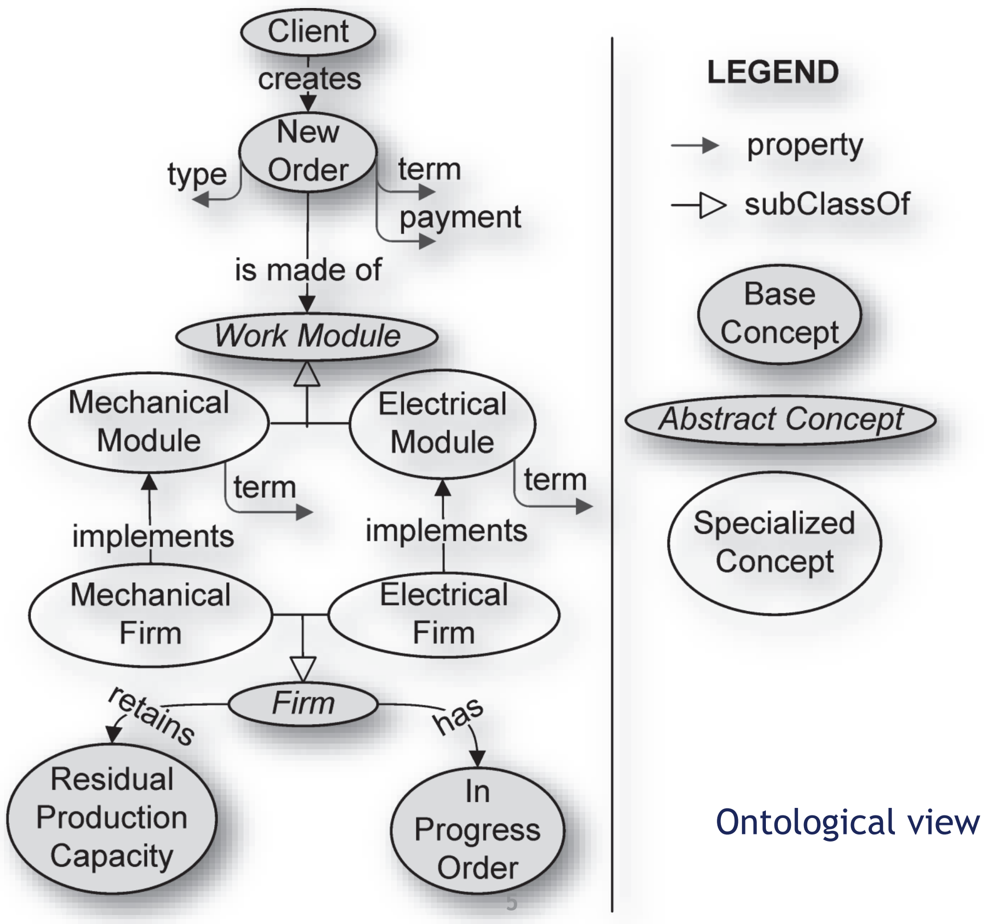
A scenario with workflow and business rules

The decision of each firm is then sent to the Planning System, which carries out a logical combination via another business rule task, i.e., Order Planning, providing the Client with the overall planning of the order. Subsequently, the Client receives the planning and performs an assessment of it. The planning can either be revised, by creating a new order, or accepted, which causes the end of the workflow.

B) Business rules

An order type can be either *standard* or *innovative*, i.e., an order very similar or completely different with respect to the past orders, respectively. An order can be performed either in the *short* or in the *long* period, depending on the following of factors: the order type, the number of "in progress" orders, the payment time, and the residual production capacity. The coordination task consists in conducting an iterative communication between the client and the firms, whose result is the order's planning or its rejection.

An ontological view of the collaborative planning of an order is represented in the next slide, where base concepts, enclosed in gray ovals, are connected by properties, represented by black directed edges. More formally, a *Client creates a New Order*, which is characterized by a *type* (which can assume the value "standard" or "innovative"), a *term* (which can assume the value "short" or "long") and a *payment* (which can assume the value "fast" or "slow").



Ontological view

The new order *is made of Work Modules*. Work module is a generalized and abstract concept, i.e., it cannot be instantiated. In figure, the name of abstract concepts is represented with italic style. *Mechanical Module* and *Electrical Module* are work modules specialized from Work Module. In figure, specialized concepts are shown with white ovals and are connected by white directed edges to the generalized concept. Each module is characterized by a *term* (which can assume the value "short" or "long"), and *is implemented by* a *Mechanical* or *Electrical Firm*, respectively. Each firm inherits two properties from the generalized concept *Firm*. A firm *has* an *in progress order*s and *retains a Residual Production Capacity.* Both properties can assume the value "true" or "false".

For the sake of brevity, in the scenario the ontology is globally shared between participants and the business rules are different for each participant. However, the ontology can be also modularized, to avoid sharing private concepts.

C) Natural-language business rules

- ❑ a mechanical firm places a new order in the short term if its type is standard and there are no in-progress orders; otherwise the order is placed in the long term;
- ❑ an electrical firm places a new order in the short time if there is a residual production capacity and the payment is fast or if the payment is slow and its type is standard;
- ❑ the planning system places a new order in the short term only if both modules have been placed in the short term.

A scenario with workflow and business rules

D) Formal IF-THEN rules

```
TASK: MECHANICAL PLANNING

RULE 1:
If newOrder.type Is standard
And inProgressOrder Is true
Then mechanicalModule.term Is long

RULE 2:
If newOrder.type Is innovative
Then mechanicalModule.term Is long

RULE 3:
If newOrder.type Is standard
And inProgressOrder Is false
Then mechanicalModule.term Is short
```

```
TASK: ELECRICAL PLANNING

RULE 1:
If residualProductionCapacity Is false
Then electricalModule.term Is long

RULE 2:
If residualProductionCapacity Is true
And newOrder.payment Is slow
And newOrder.type Is innovative
Then electricalModule.term Is long

RULE 3:
If residualProductionCapacity Is true
And newOrder.payment Is fast
Then electricalModule.term Is short

RULE 4:
If residualProductionCapacity Is true
And newOrder.payment Is slow
And newOrder.type Is standard
Then electricalModule.term Is short
```

```
TASK: ORDER PLANNING

RULE 1:
If mechanicalModule.term Is long
Then newOrder.term Is long

RULE 2:
If electricalModule.term Is long
Then newOrder.term Is long

RULE 3:
If mechanicalModule.term Is short
And electricalModule.term Is short
Then newOrder.term Is short
```

A scenario with workflow and business rules

E) Collaborative Analytics

- ❑ Business rules are usually designed according to goals which are measurable via related Key Performance Indicators (KPIs), for each company and for the community itself.
- ❑ For this reason, the usability of the data flow connected to the workflow is a fundamental requirement.
- ❑ In a collaborative network the computation of KPIs must preserve the marketing value of data source to be aggregated, avoiding industrial espionage between competitors.
- ❑ The focus here is not on specific KPIs: the technique is suitable for any business measurements that need to be aggregated handling company’s data.
- ❑ The problem in general can be brought back to comparing providers’ performance. In practice, a collective comparison is related to the “*to share or not to share*” dilemma, an important reason for the failure of data sharing in collaborative networks.

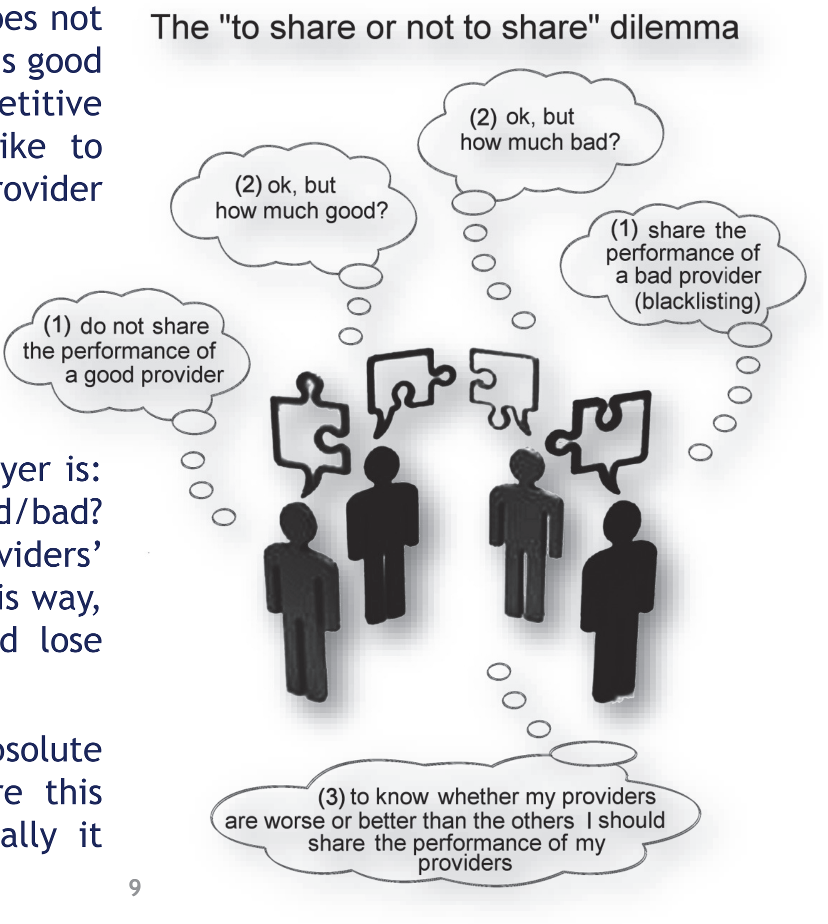
A scenario with workflow and business rules

In the dilemma, a typical buyer does not like to share the performance of his good providers (keeping a competitive advantage over its rivals) and like to share the performance of a bad provider (showing his collaborative spirit).

However, each buyer knows a subset of the providers available on the market.

The fundamental question of a buyer is: how much are my providers good/bad? To solve this question, providers' performance should be shared. This way, buyers with good providers would lose the competitive advantage.

Given that nobody knows the absolute ranking of his providers, to share this knowledge is risky and then usually it does not happen.


The "to share or not to share" dilemma
(2) ok, but how much bad?
(2) ok, but how much good?
(1) share the performance of a bad provider (blacklisting)
(1) do not share the performance of a good provider
(3) to know whether my providers are worse or better than the others I should share the performance of my providers

A scenario with workflow and business rules

Let us consider an extension of the pilot scenario, with a new behavior in the workflow: when the mechanical or the electrical planning does not satisfy the client requirements, the Planning System must be able to select an alternative partner.

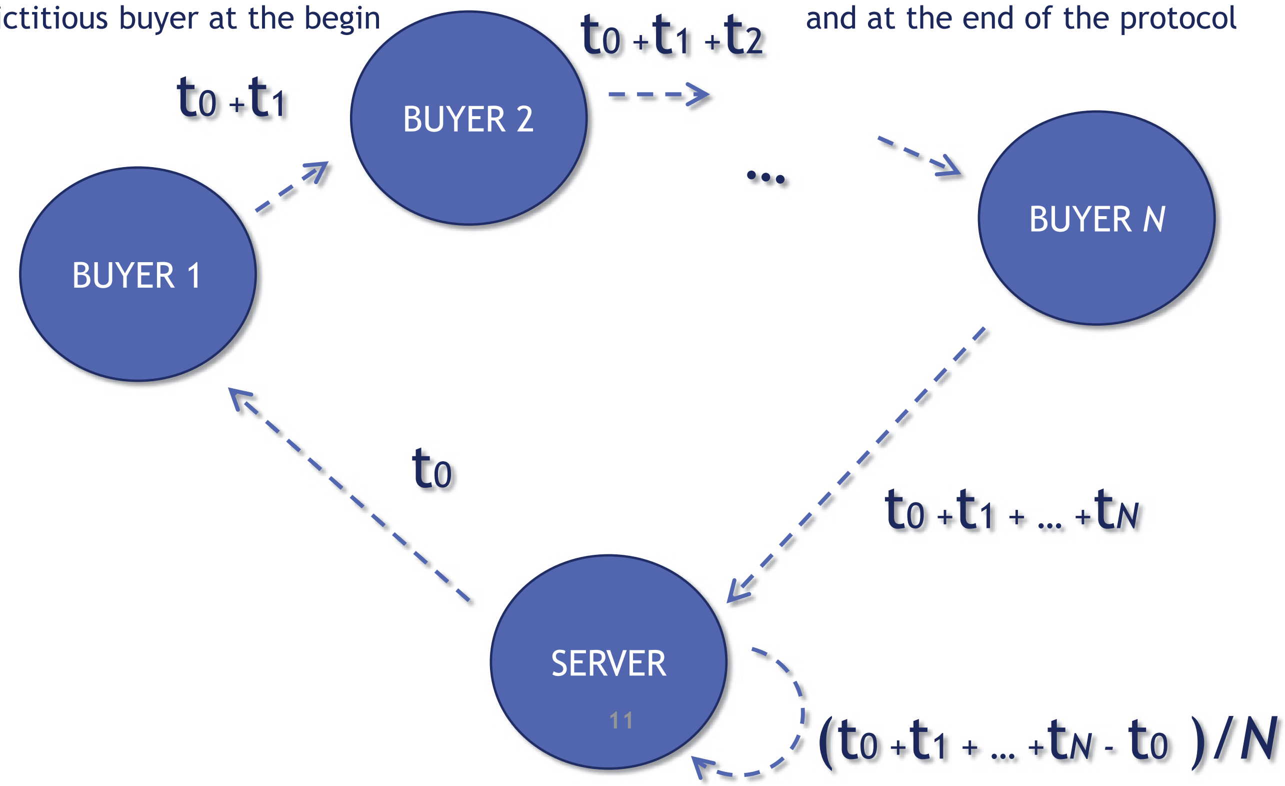
To achieve this extension, an *Order Planning Assessment* activity should be carried out by the Planning System too. Then, another activity, called *Select Alternative Partner*, should compare partners' performance to carry out a selection. Such performance must be made available by a collaborative analytics process.

The next slide shows an example of data flow designed to implement a privacy-preserving collaborative analytics process. The Collaborative Analytics System (called hereafter "System" for the sake of brevity) is the main pool located on a shared server and coordinating pools of registered buyers. Each buyer's pool is located on a private server.

The main goal of the data flow is to create a public collective data by aggregating buyers' private data. For instance, let us consider a community of N buyers B_1, B_2, … B_N, and a community of M vendors V_1, V_2, … V_M, each buyer being supplied by a small subset of the vendors. The average delivery time of the vendors of a buyer is an example of private datum, whereas the average delivery time of the vendors of all buyers is an example of collective datum.

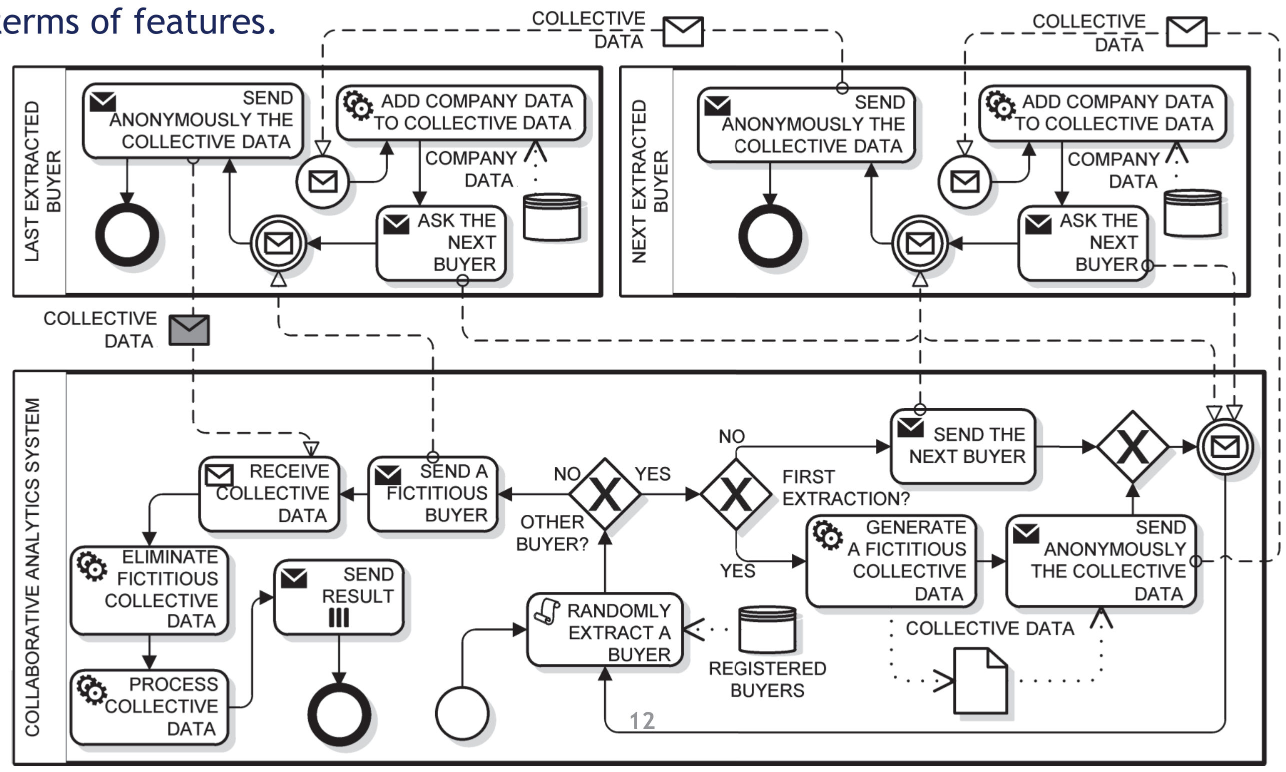
A scenario with workflow and business rules

The problem: how to calculate the average without sending each term t_k to the server?
The solution: each buyer receives a partial summation, adds its own term and sends the next partial summation to the next buyer. The server orchestrates step-by-step a random sequence of buyers. At each step, the next buyer is asked to the server, which does not manage partial summations. The messaging is trusted but anonymous and the server can act as a fictitious buyer at the begin and at the end of the protocol



A scenario with workflow and business rules

In general the aggregation process protects buyers' datum from being publicized. More specifically, at the beginning the System randomly extracts a buyer and generates a fictitious collective datum. A fictitious datum is an artificial creation that mimics real-world datum, and then cannot be distinguished from actual datum in terms of features.

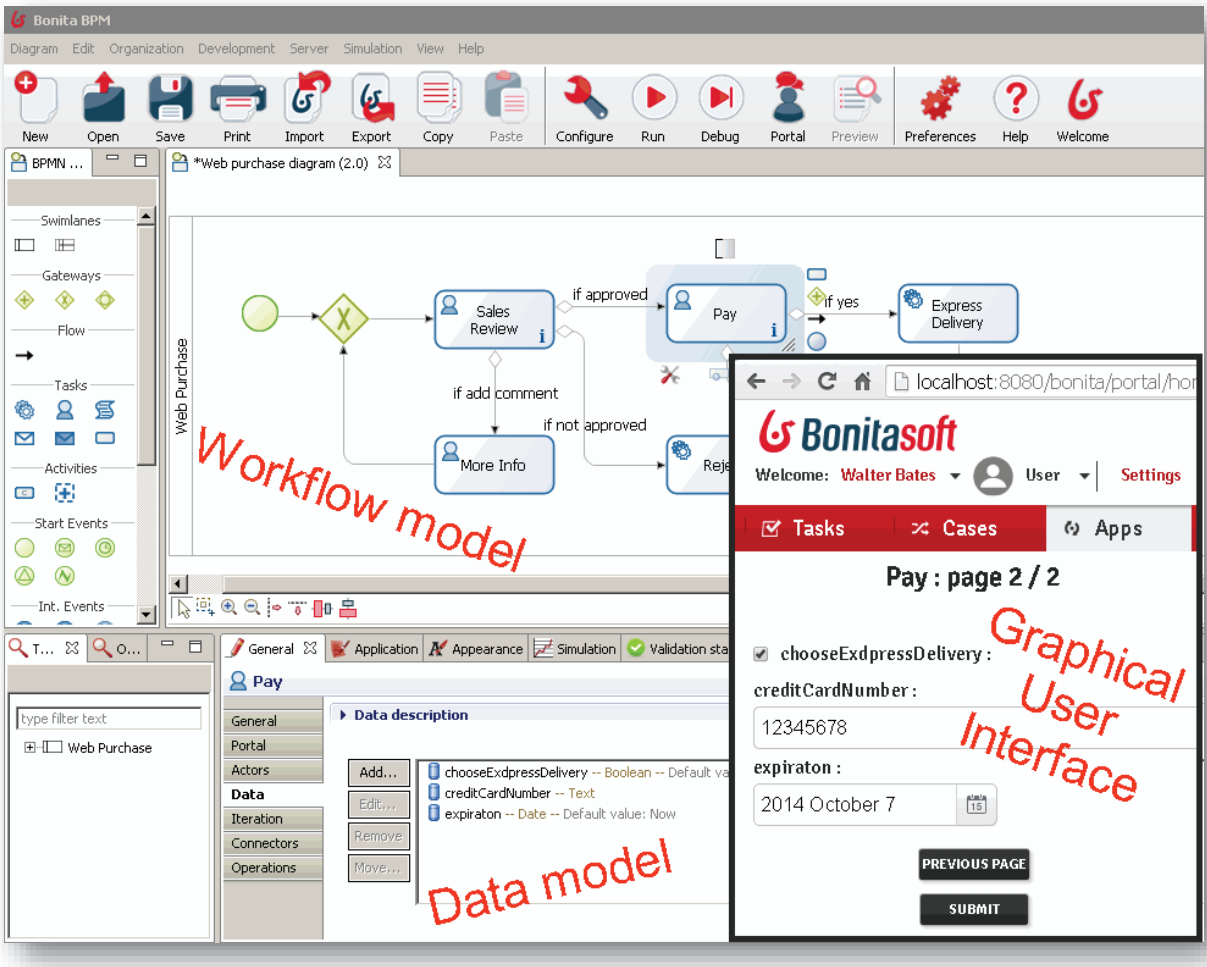


A scenario with workflow and business rules

- ❑ Collective datum is then anonymously sent to the extracted buyer, who adds his private datum to it and ask the System for the next buyer.
- ❑ The system will answer with a randomly extract next buyer. Then, the buyer sends anonymously collective datum. This way, collective datum is incrementally built and transferred from a buyer to another one, under orchestration of the System.
- ❑ Each buyer is not aware of his position in the sequence. This is because the first extracted buyer receives a fictitious collective datum, and because the sender is always anonymous.
- ❑ The last extracted buyer will be provided with a fictitious buyer by the system. Such fictitious buyer actually corresponds to the System itself. After receiving the collective datum, the System subtracts the initial fictitious datum, thus obtaining the actual collective datum, which is then processed (so as to extract some common features) and sent to all buyers.
- ❑ By comparing the collective datum with his private datum, each buyer will be able to assess his position with respect to the collective performance. The results of this process can be used by to select a partner whose performance is higher than the collective performance.

Business Process Management System

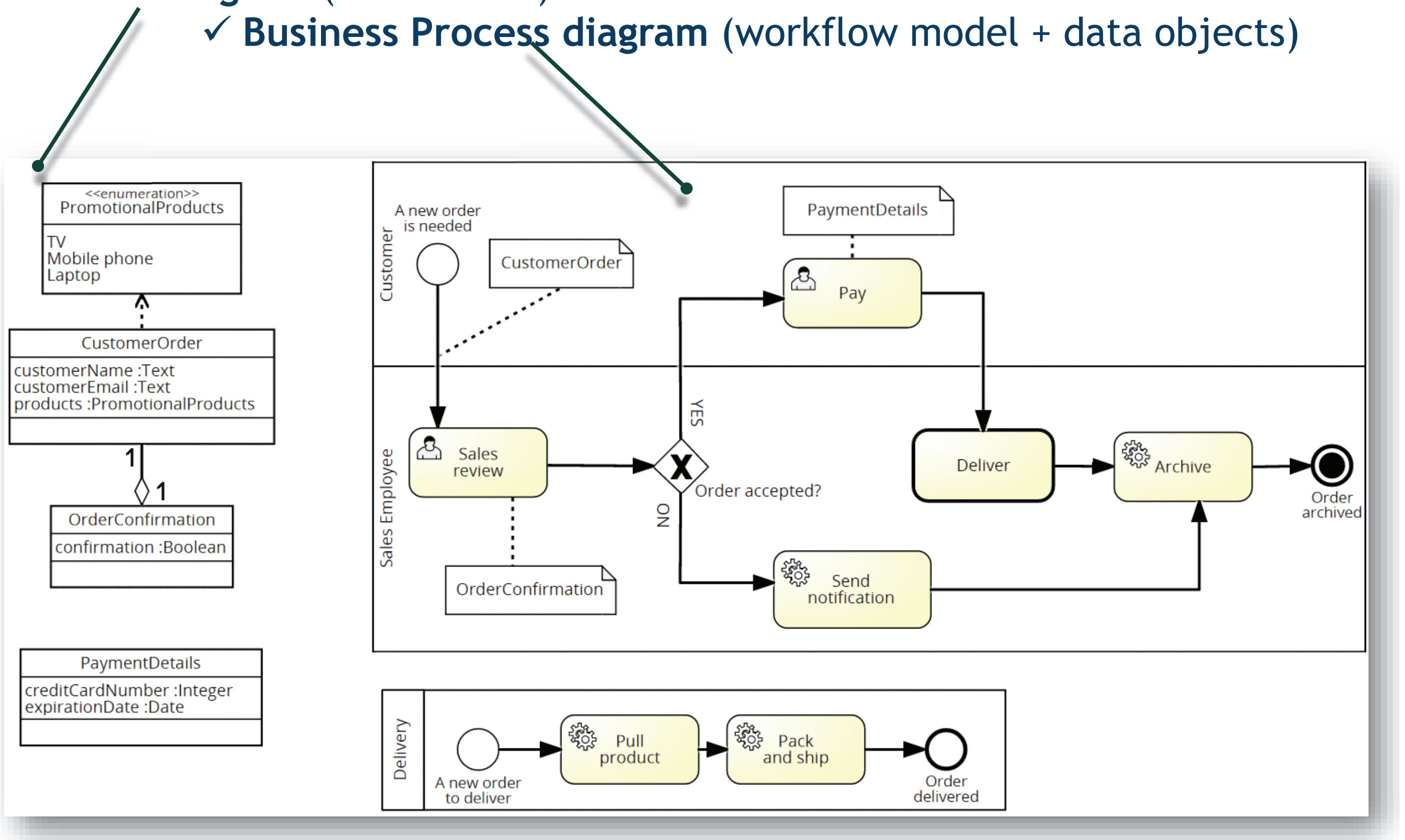
- ✓ Bonita BPM 7 is a powerful application platform for building personalized, process-based business applications that adapt to your business changes in real time.
- ✓ Bonita BPM has two parts: the development environment, Bonita BPM Studio, and the runtime environment, Bonita BPM Platform.
- ✓ Bonita BPM adopts the **model-driven approach**, a software design methodology for the development of software systems, launched by the Object Management Group (OMG) in 2001.
- ✓ With model-driven engineering, specifications are expressed as models. Models can be expressed with standards, such as the executable Unified Modeling Language (UML), and the BPMN.
- ✓ Models are then processed to automatically generate software. Code generation means that an automated tool derives from the models parts or all of the source code for the software system.


Bonita BPM
Diagram Edit Organization Development Server Simulation View Help
New Open Save Print Import Export Copy Paste Configure Run Debug Portal Preview Preferences Help Welcome
*Web purchase diagram (2.0)
Web Purchase
Sales Review
if approved
Pay
if yes
Express Delivery
if add comment
if not approved
More Info
Workflow model
General Application Appearance Simulation Validation sta
Pay
General
Portal
Actors
Data
Iteration
Connectors
Operations
Data description
Add... Edit... Remove Move...
chooseExdpressDelivery -- Boolean -- Default va
creditCardNumber -- Text
expiraton -- Date -- Default value: Now
Data model
localhost:8080/bonita/portal/hor
Bonitasoft
Welcome: Walter Bates User Settings
Tasks Cases Apps
Pay : page 2 / 2
chooseExdpressDelivery :
creditCardNumber :
12345678
expiraton :
2014 October 7
PREVIOUS PAGE
SUBMIT
Graphical User Interface

✓ Our first model, edited with Signavio

✓ **Class diagram** (data model)

✓ **Business Process diagram** (workflow model + data objects)



BP Modeling: Web purchase model with Signavio

1. Download the Bonita BPMS from
   http://www.iet.unipi.it/m.cimino/wdis/
   “Process Management suite: Bonita BPM 7.x [local]”
   http://www.iet.unipi.it/m.cimino/wdis/res/BonitaBPMCommunity-7.5.4.zip

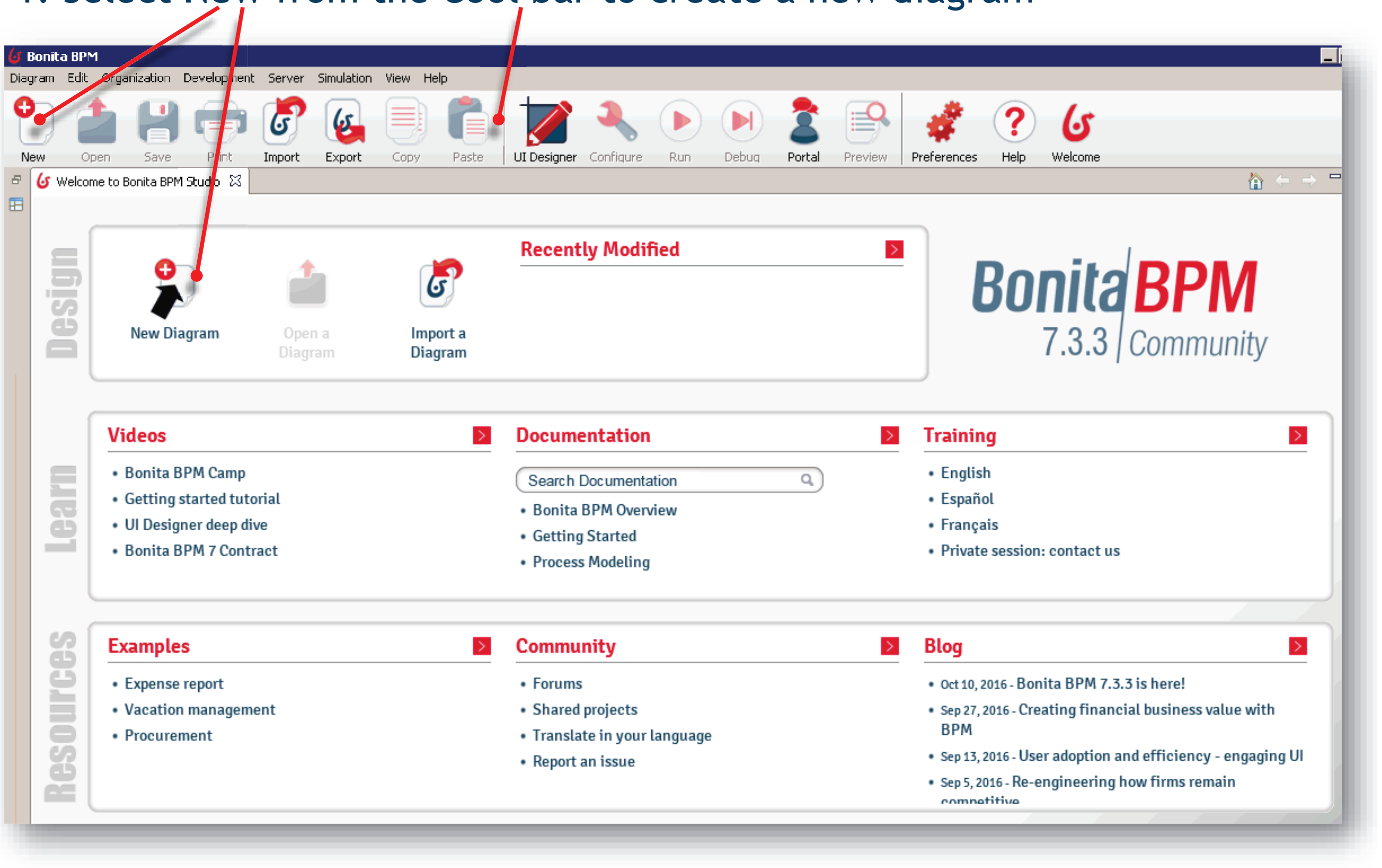
2. Extract it to c:\wdis

3. If needed, change the JDK: create a batch file (go.bat)

set JAVA_HOME=C:\wdis\jdk8
set PATH=C:\wdis\jdk8\bin;%PATH%
java -version (1.8)
BonitaBPMCommunity64.exe

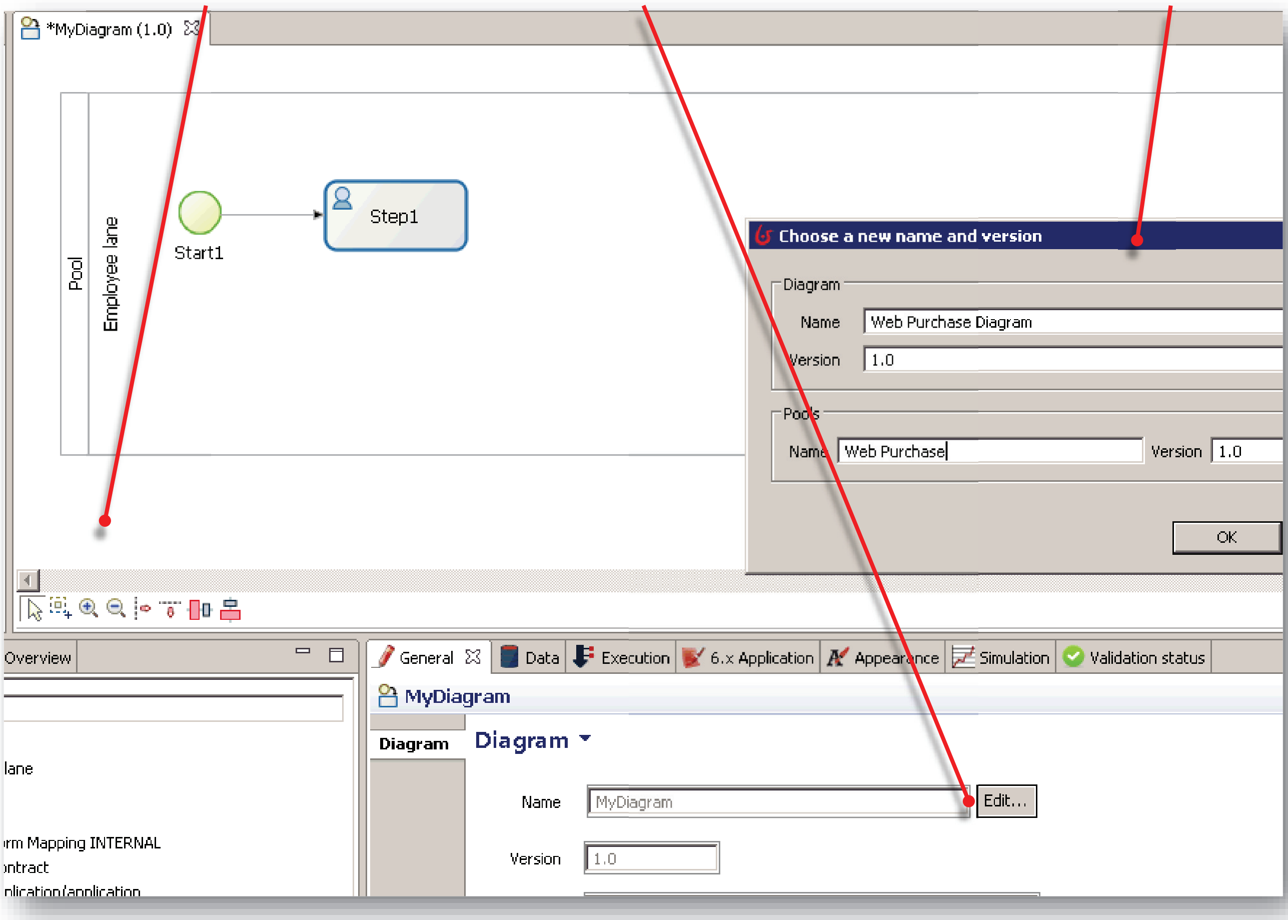
BP Management: Web purchase example with Bonita BPM

1. Select **New** from the Cool bar to create a new diagram


Bonita BPM
Diagram Edit Organization Development Server Simulation View Help
New Open Save Print Import Export Copy Paste UI Designer Configure Run Debug Portal Preview Preferences Help Welcome
Welcome to Bonita BPM Studio
Design
New Diagram
Open a Diagram
Import a Diagram
Recently Modified
Bonita BPM 7.3.3 Community
Learn
Videos
Bonita BPM Camp
Getting started tutorial
UI Designer deep dive
Bonita BPM 7 Contract
Documentation
Search Documentation
Bonita BPM Overview
Getting Started
Process Modeling
Training
English
Español
Français
Private session: contact us
Resources
Examples
Expense report
Vacation management
Procurement
Community
Forums
Shared projects
Translate in your language
Report an issue
Blog
Oct 10, 2016 - Bonita BPM 7.3.3 is here!
Sep 27, 2016 - Creating financial business value with BPM
Sep 13, 2016 - User adoption and efficiency - engaging UI
Sep 5, 2016 - Re-engineering how firms remain competitive

Bonita BPM: Diagram and Pool Name

2. Click outside the pool, click on Edit, Enter Diagram and Pool Name



Diagram

Name	Web Purchase Diagram
Version	1.0

Pools

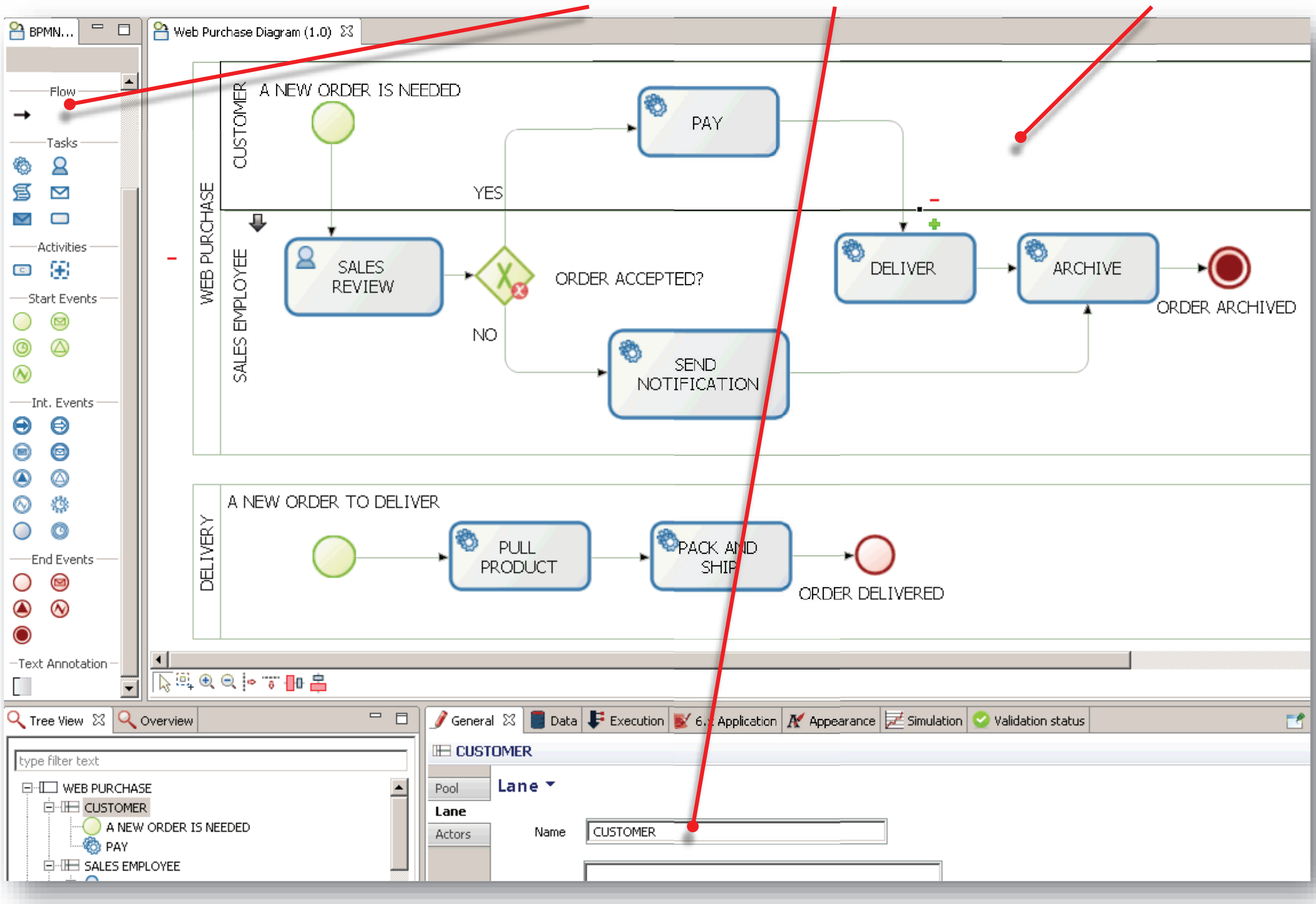
Name	Web Purchase	Version	1.0

Diagram

Name	MyDiagram
Version	1.0

Bonita BPM: Draw the BPMN model and set names

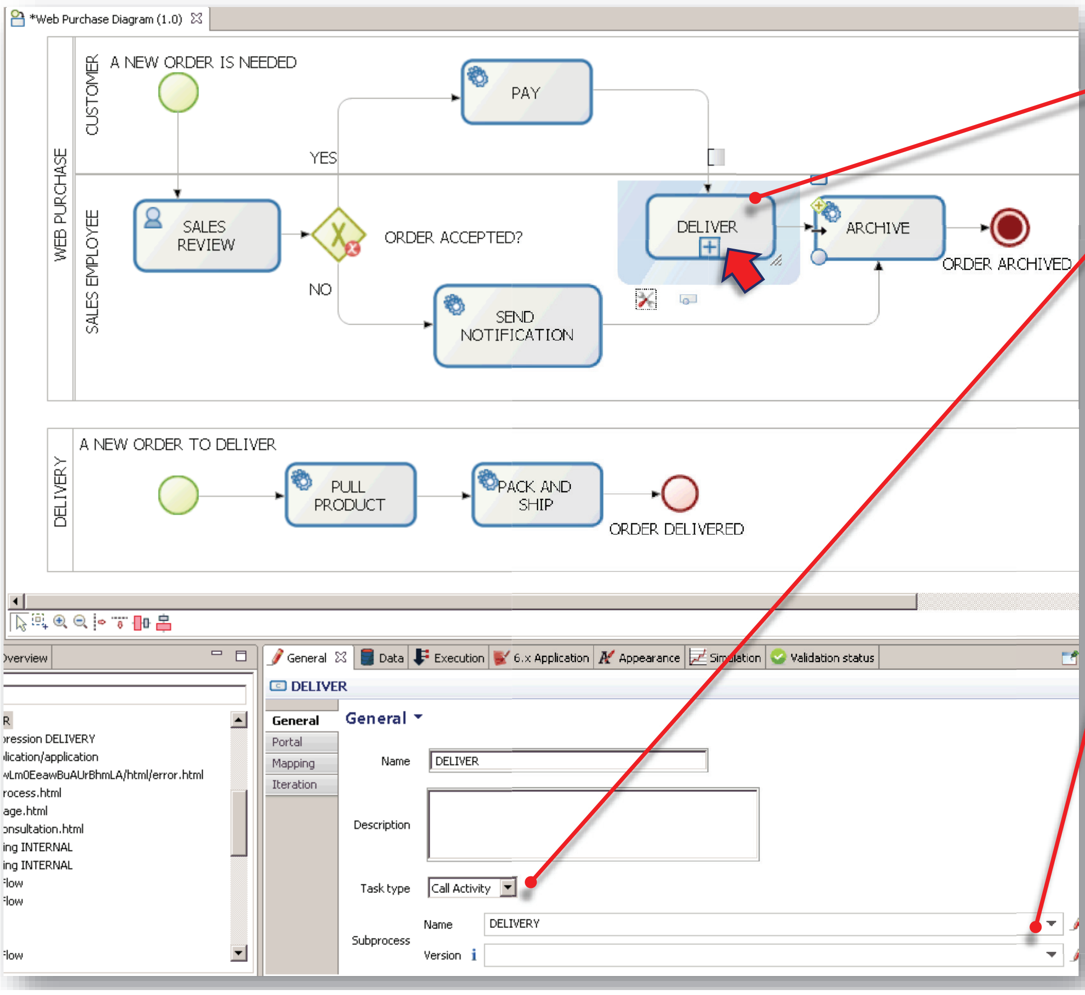
3. Create the diagram using the toolkit, configure the selected element



Lane

Name	CUSTOMER

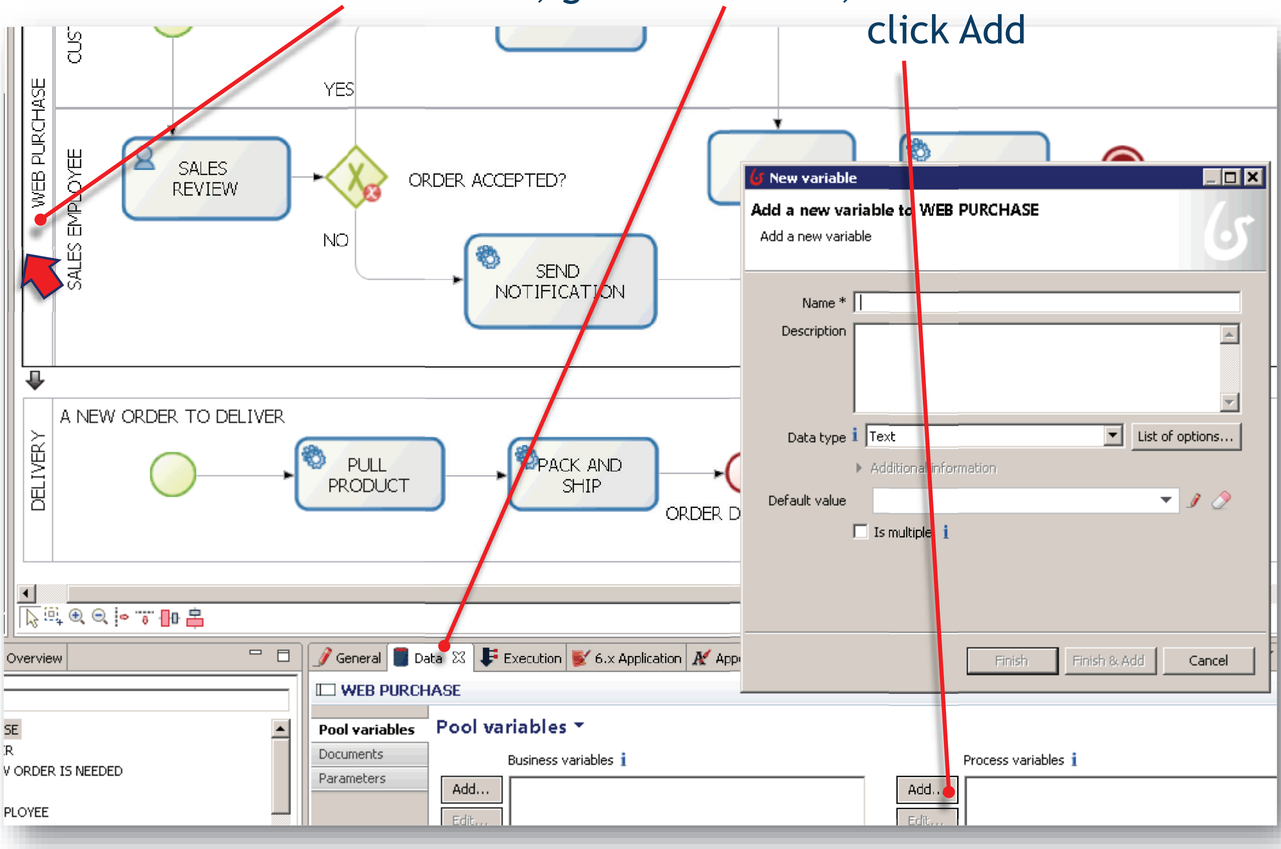
Bonita BPM: Connect a sub-process to a pool



4. Select the Deliver task, choose the task type *Call Activity*, and choose *Delivery* as a target name

Bonita BPM: Add Process variables

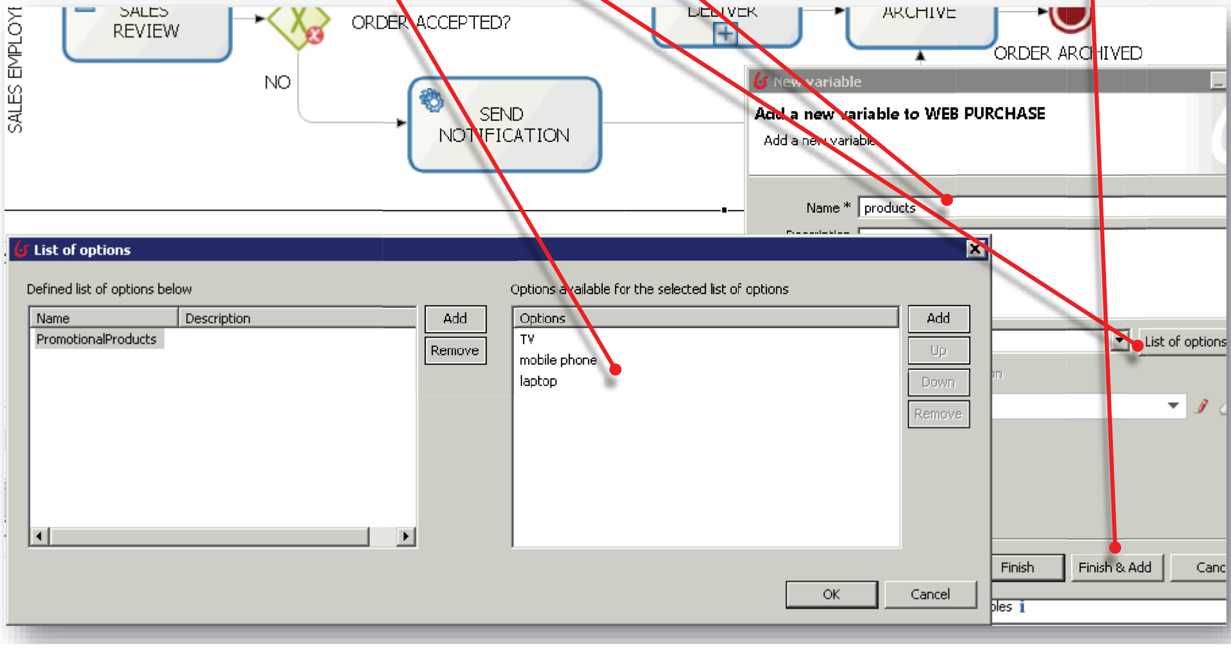
5. Select the Web Purchase Pool, go to Data Pane, on Process variable click Add



Bonita BPM: Add enumeration process variables

Process variables: can be used in a process and until the process instance is completed.

6. Enter *customerName*, leave Data type *Text*, and press *Finish&Add*
   enter *customerEmail*, leave Data type *Text*, and press *Finish&Add*
   enter *creditCardNumber*, Data type *Integer,* press *Finish&Add*
   enter *expirationDate,* Data type *Date,* press *Finish&Add*
   enter *confirmation*, Data type Boolean, press *Finish&Add*
   enter *products,* click on *List of options*, Name: *PromotionalProducts*,
   Options: *TV, Mobile phone e laptop*.

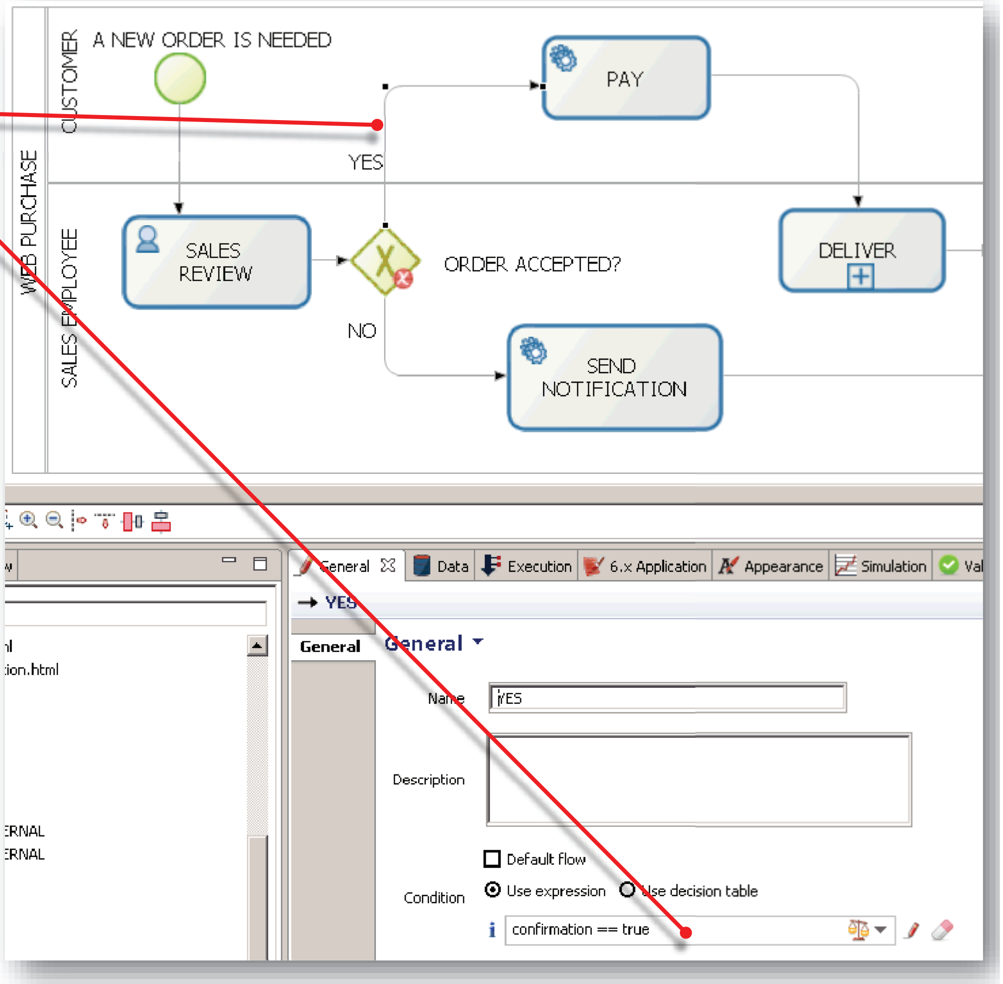


Bonita BPM: set expressions for the exclusive transitions

Transitions:

Select the branch "YES", enter the expression "confirmation == true"

Select the branch "NO", anter the expression "confirmation == false".



Bonita BPM: configure a message connector for a task

Connector:

Select the *Send notification* task,

Select *Execution* tab

Select *Connectors out*

➢ *Add*

In the popup windows Select *Categories Messaging*

➢ Email (SMTP)

Name: *Send notification*

➢ Next

➢ enter gmail account

➢ Next

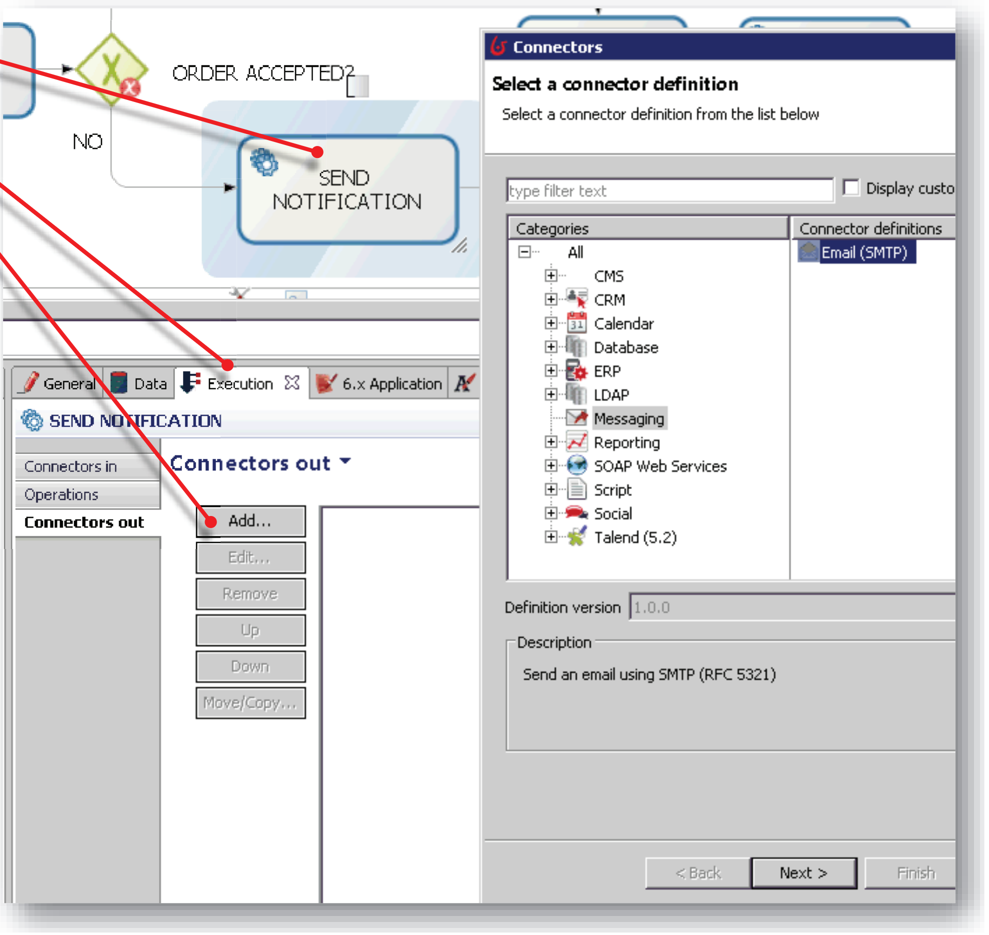
➢ enter *from, to*

➢ *Next*

➢ enter

Subject: *Order refused*, Message: *We are unable to fill your order at this time*

➢ Click on test, ignore the popup warning, check your email, > Finish.



Bonita BPM: create actors for lanes with human tasks

Connector:

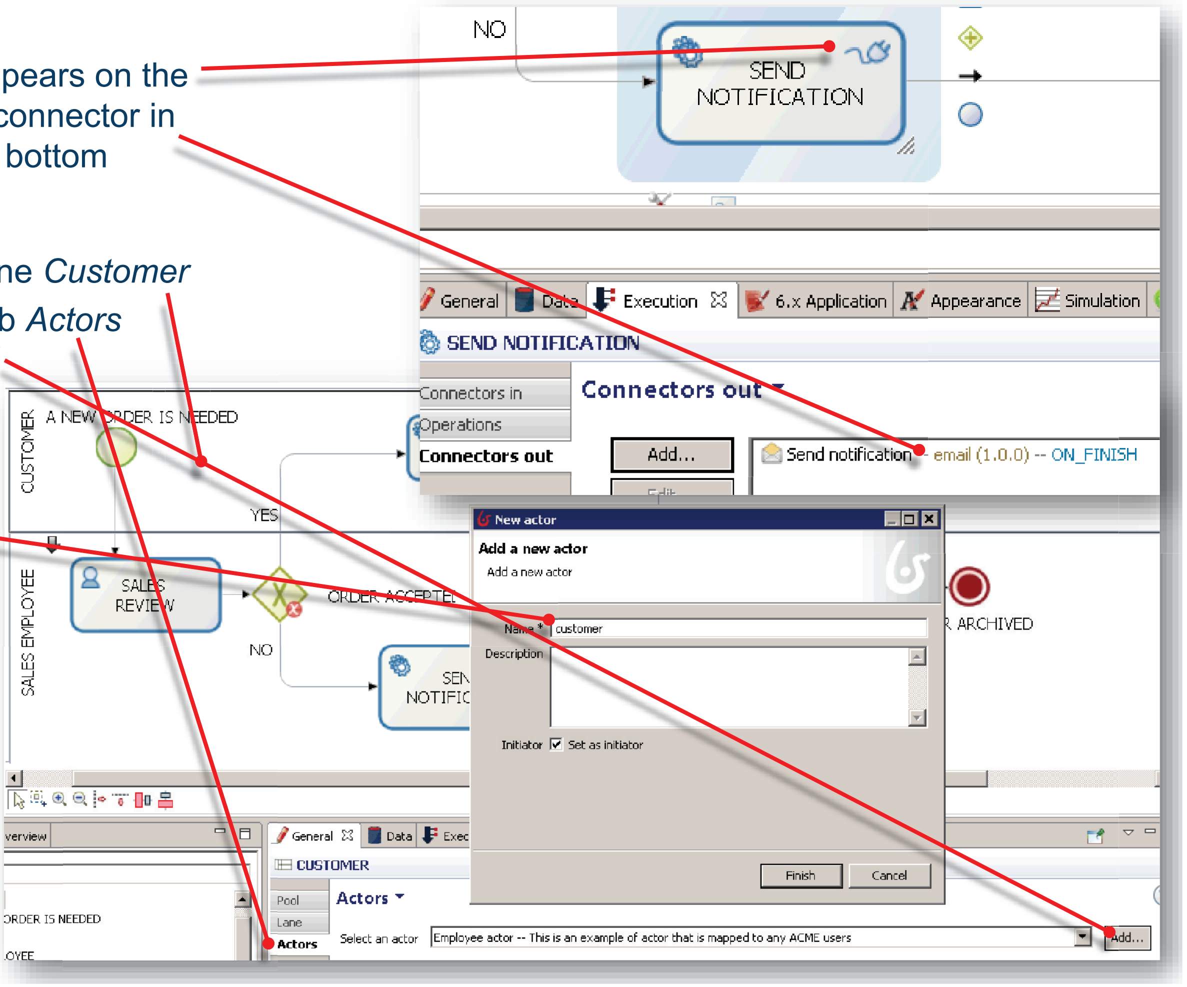
A plug icon appears on the task, and the connector in the tab on the bottom

Actors:

- Select the lane *Customer*
- Select the tab *Actors*
- Click on *Add*
- Name: *customer*
- *Set as initiator*

Check *set as initiator*

- Select the Lane *Sales Employee*
- *Add*
- Name: *employee*
- Finish



Bonita BPM: Map actors to people, forms for human tasks

Mapping Actors –people

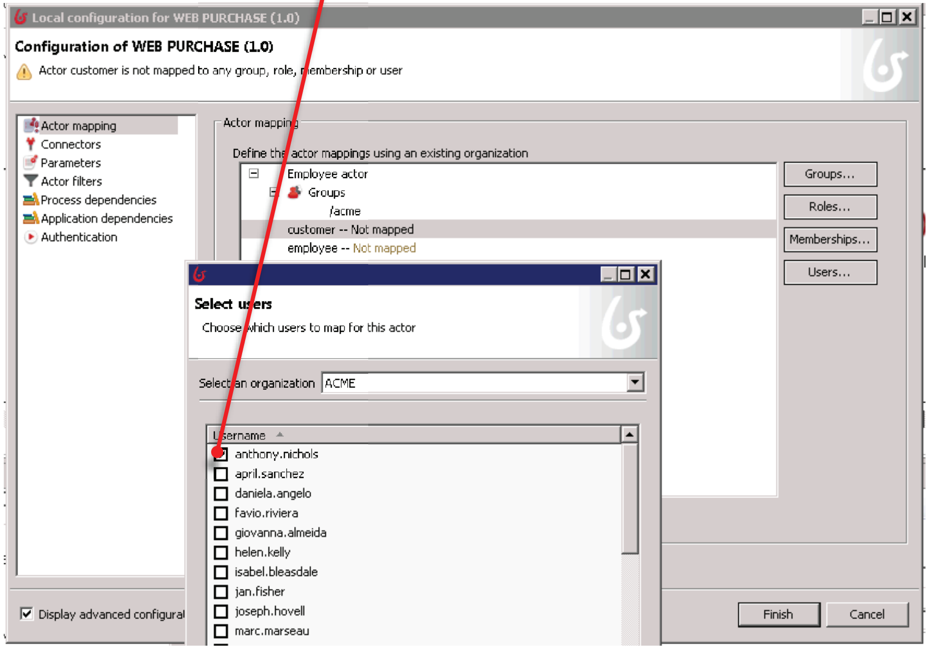
- Click on *Configure* on the cool bar.
- Select *customer* in Actor mapping > *Users* > *anthony.nichols (pwd* bpm)

Similarliy

- Click on *Configure* on the cool bar.
- Select *employee* in Actor mapping > *Users* > *april.sanchez (pwd* bpm)

Forms and Data Objects

- Select the *Web Purchase Pool* > Tab *Execution* > *Tab Instantiation Form*
  - 6.x
  - 6.x Application
  - Add
  - …


Local configuration for WEB PURCHASE (1.0)

Configuration of WEB PURCHASE (1.0)

Actor customer is not mapped to any group, role, membership or user

Define the actor mappings using an existing organization

- Employee actor
  - Groups
    - /acme
- customer -- Not mapped
- employee -- Not mapped

Select users

Choose which users to map for this actor

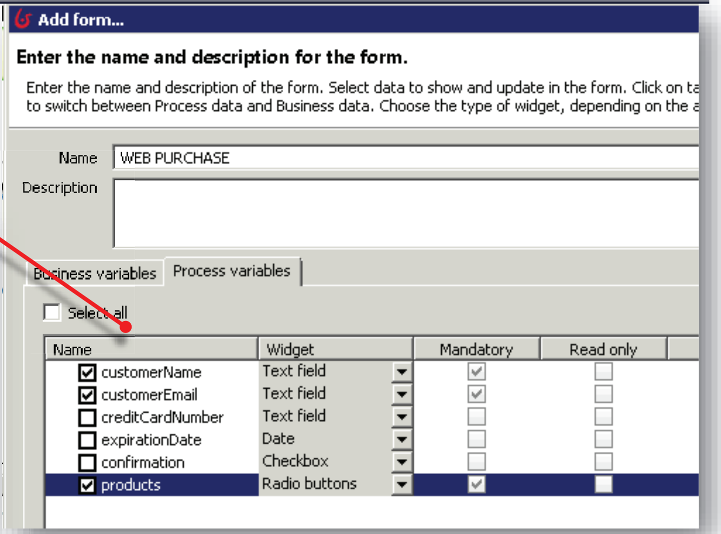
Select an organization: ACME

Username
☑ anthony.nichols
☐ april.sanchez
☐ daniela.angelo
☐ favio.riviera
☐ giovanna.almeida
☐ helen.kelly
☐ isabel.bleasdale
☐ jan.fisher
☐ joseph.hovell
☐ marc.marseau

Bonita BPM: forms for human tasks

Forms and Data Objects

- Select the Process Variables *customerName, customerEmail*, *products.*
- A default form is created


Add form...

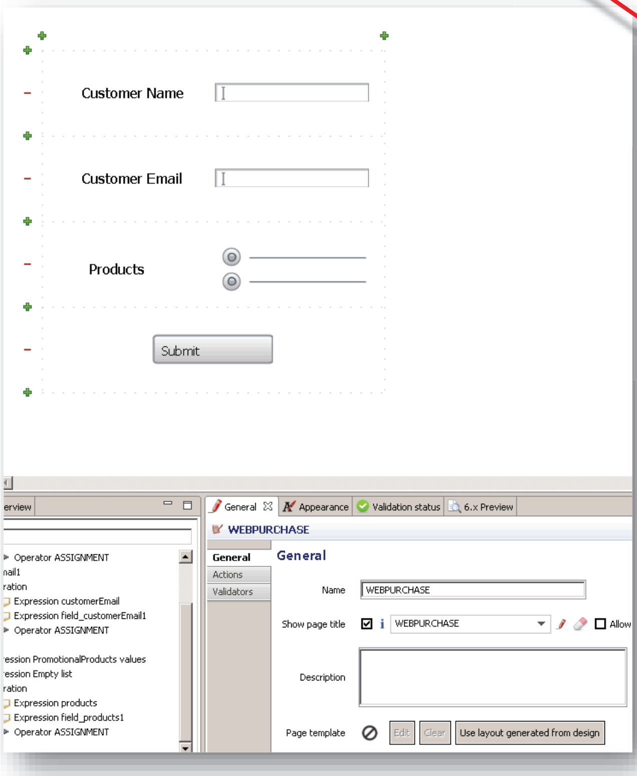
Enter the name and description for the form.

Enter the name and description of the form. Select data to show and update in the form. Click on ta
to switch between Process data and Business data. Choose the type of widget, depending on the a

Name: WEB PURCHASE

Description:

Name	Widget	Mandatory	Read only
☑ customerName	Text field	✓	☐
☑ customerEmail	Text field	✓	☐
☐ creditCardNumber	Text field	☐	☐
☐ expirationDate	Date	☐	☐
☐ confirmation	Checkbox	☐	☐
☑ products	Radio buttons	✓	☐



Operator ASSIGNMENT
nail1
ration
Expression customerEmail
Expression field_customerEmail1
Operator ASSIGNMENT
ession PromotionalProducts values
ession Empty list
ration
Expression products
Expression field_products1
Operator ASSIGNMENT

Name: WEBPURCHASE

Show page title: ☑ WEBPURCHASE

Description:

Page template

Activity *Sales review*

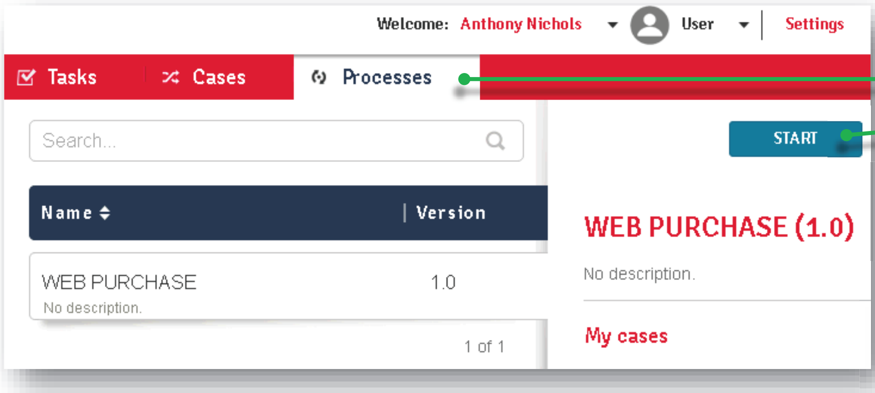
- Tab *Execution* > *Tab Form* > *6.x Application* > Add Select *customerName*, *customerEmail*, and *products* as read only; finally add *confirmation*

Activity *Pay*

- Tab *General* > *task type: Human*
- Tab *Execution* > *Tab Form* > *6.x Application* > Add > Select *creditCardNumber*, and *expiration Date.*

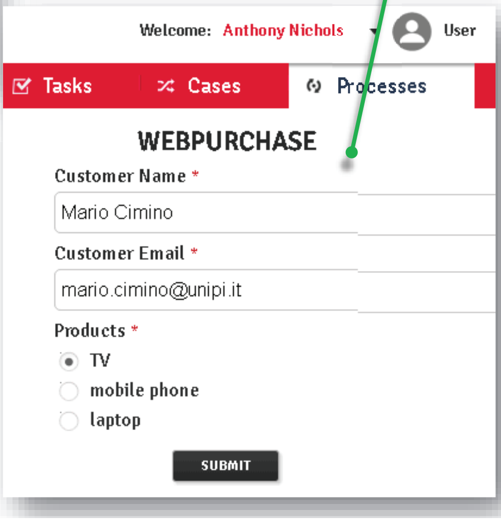
Bonita BPM: use case

- Click the Run button in the Cool bar
- Open two different browsers and point to http://localhost:8080/bonita/login.jsp
- First browser > customer login > username: *anthony.nichols* password: *bpm*
- Second browser > employee login > username: *april.sanchez* password: *bpm*



- as a customer, on the tab *processes* click on *start*

- fill the Web Purchase form and click on *submit*



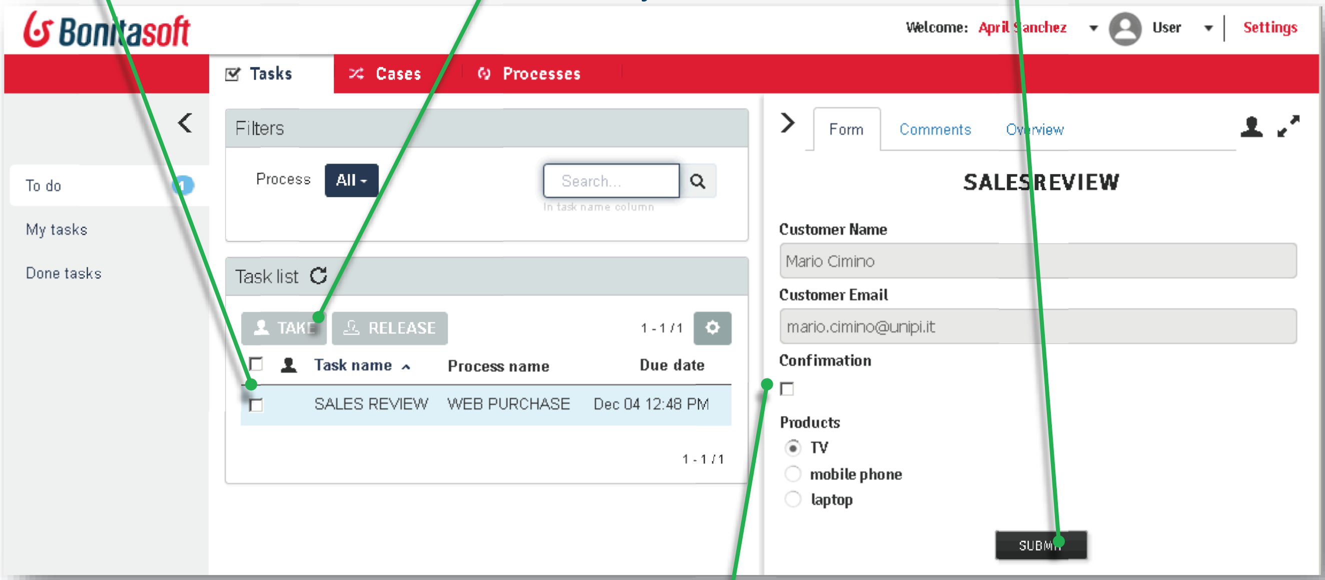
Customer Name *: Mario Cimino
Customer Email *: mario.cimino@unipi.it
Products *: [✓ TV] mobile phone laptop

- In the second browser, as an employee, on the tab tasks there is a task **to do**

Bonita BPM: use case

- Select the task and press *take*

- first case: press on SUBMIT (without confirmation) > you will receive an email



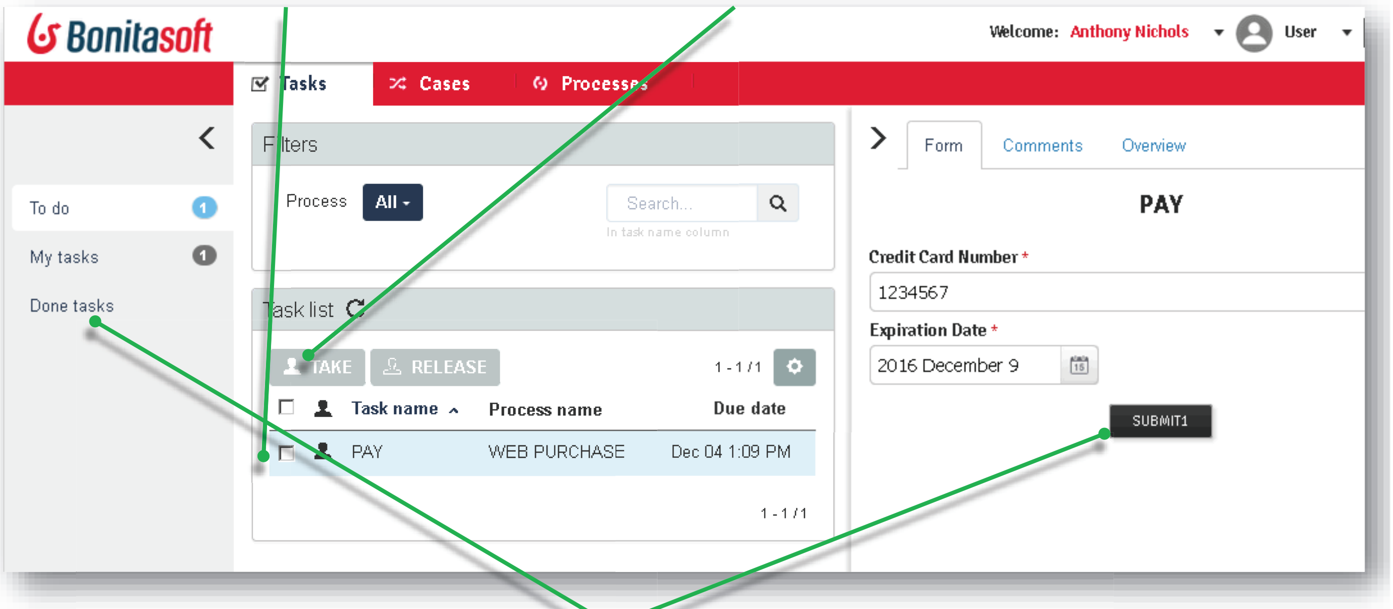
Task name	Process name	Due date
SALES REVIEW	WEB PURCHASE	Dec 04 12:48 PM

Customer Name: Mario Cimino
Customer Email: mario.cimino@unipi.it
Confirmation: ☐
Products: [✓ TV] mobile phone laptop

- As a customer, start a new process in the first browser
- Fill again the customer form and submit
- As an employee, check the confirmation flag and submit

Bonita BPM: use case

- As a customer, select the task PAY and press *take*



Task name	Process name	Due date
PAY	WEB PURCHASE	Dec 04 1:09 PM

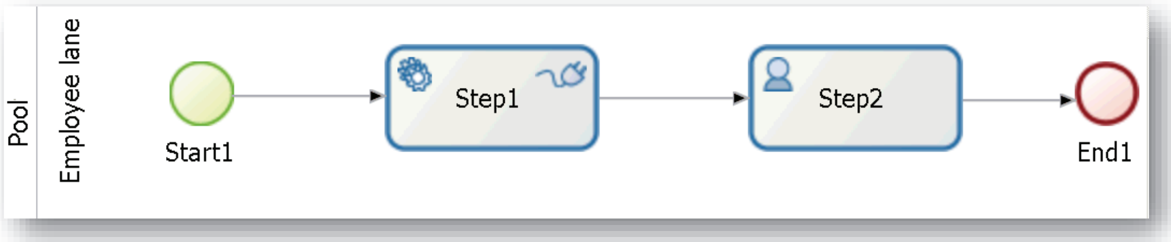
PAY

Credit Card Number *: 1234567

Expiration Date *: 2016 December 9

- Fill the PAY form on the right and submit
- As an employee, you can now see in *done tasks* the task history

Bonita BPM: Database and Web Service connectors



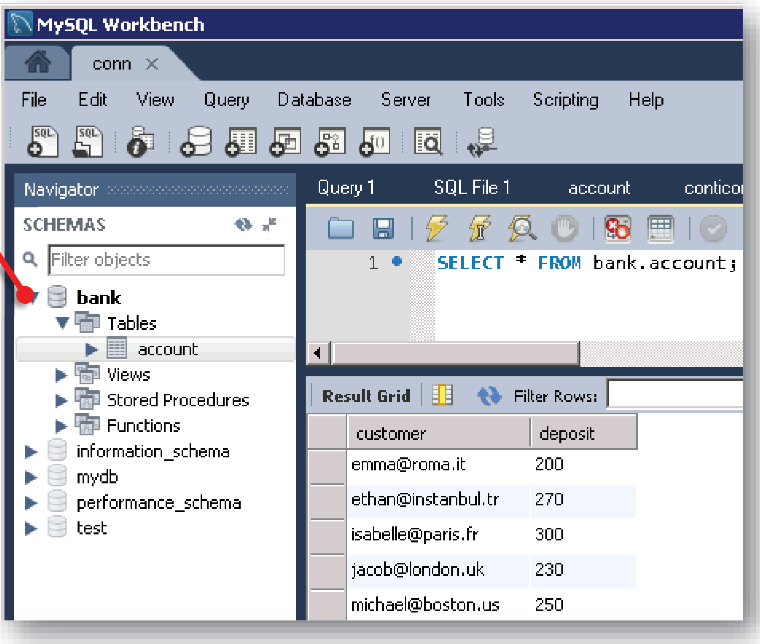
1. **Create the diagram above (for detailed steps see the first tutorial):**
2. New Diagram > complete the flow with the toolkit leaving the default task types.
3. Select *Step1* > *General* Tab > *Task type:* Service.
4. Select *Step2* > *General* Tab > *Task type:* Human.
5. Click on *Save* in the cool bar.
6. **Create the process variables:**
7. Select *Pool* > *Data* Tab > *Process Variables: Add* > Name: *customer* > *Finish* & *Add* > Name: *deposit* > *Finish*
8. ***Create the pool form***
9. Select *Pool* > Tab *Execution* > Instantiation form > 6.x
10. Tab *6.x Application* > *Add* > Select Tab *Process variables* > Select *deposit*, and *mandatory* > *Finish*
11. ***Create the Step2 form***
12. Select *Step2* > Tab *Execution* > form > 6.x
13. Tab *6.x Application* > *Add* > Select Tab *Process variables* > Select *customer*, and *read only* > *Finish*

Bonita BPM: Database connector

14. **Create the MySQL Database:**
15. 1st method: import the file *bank-dump.sql* into a MySQL server.
16. 2nd method: download the file www.iet.unipi.it/m.cimino/wdis/res/dbms.zip and extract it on C:\wdis. Finally, click on C:\wdis\mysqlStart
17. **Access the Database with MySQL client:**
18. Click on C:\wdis\mysqlClient6.1 > Click on the "+" icon close to MySQL connections > enter a name and click OK.
19. Select the *bank* schema > *Tables* > *account* > right click > Select rows.
20. **Create the DB Connector:**
21. On Bonita, select *Step1* > Tab *Execution* > Connectors out (*) > *Add* > Categories: *Database* > *Others* > *Connector definition* > *MySQL 5.5* JDBC 4… > Next
22. Name: *dbconn1* > *Next.* Enter

URL: *jdbc:mysql://localhost:3306/bank*
Username: *root*   Password:
Next

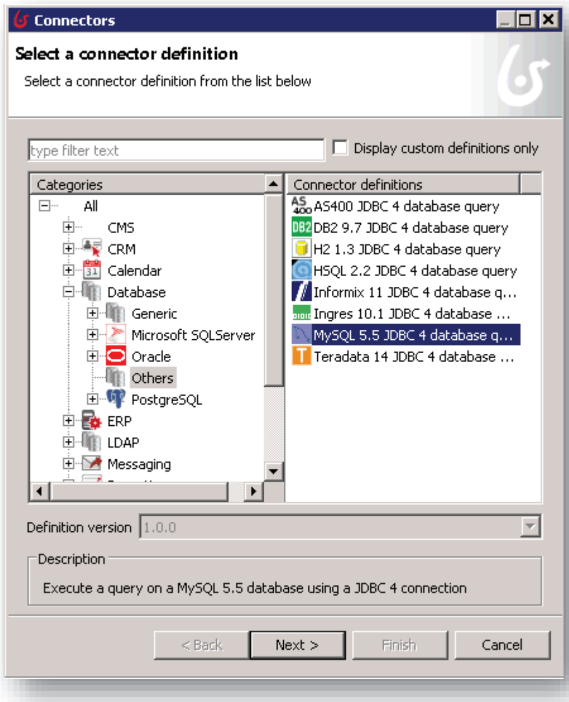
(*) *Connectors out* are carried out at the end of the step, whereas *Connectors in* at the begin of the step.



```
SELECT * FROM bank.account;
```

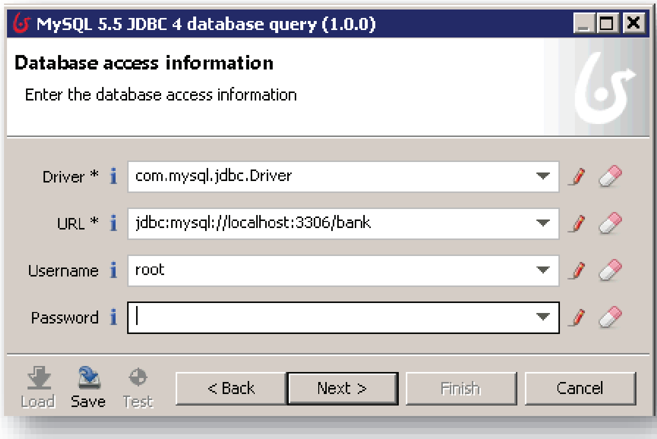
customer	deposit
emma@roma.it	200
ethan@instanbul.tr	270
isabelle@paris.fr	300
jacob@london.uk	230
michael@boston.us	250

Bonita BPM: Database connector

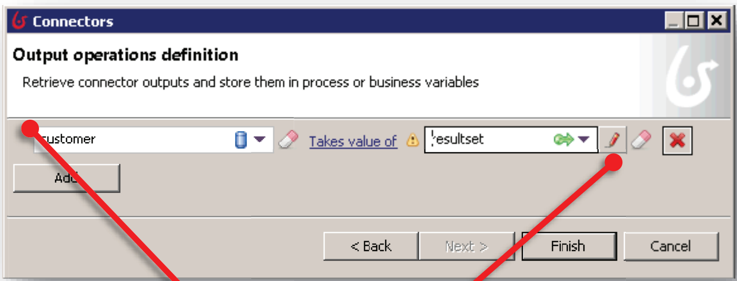

Connectors
Select a connector definition
Select a connector definition from the list below
type filter text
Display custom definitions only
Categories
All
CMS
CRM
Calendar
Database
Generic
Microsoft SQLServer
Oracle
Others
PostgreSQL
ERP
LDAP
Messaging
Connector definitions
AS400 JDBC 4 database query
DB2 9.7 JDBC 4 database query
H2 1.3 JDBC 4 database query
HSQL 2.2 JDBC 4 database query
Informix 11 JDBC 4 database q…
Ingres 10.1 JDBC 4 database …
MySQL 5.5 JDBC 4 database q…
Teradata 14 JDBC 4 database …
Definition version
< Back
Next >
Finish
Cancel

Definition version: 1.0.0

Description: Execute a query on a MySQL 5.5 database using a JDBC 4 connection


MySQL 5.5 JDBC 4 database query (1.0.0)
Database access information
Enter the database access information
Load
Save
Test
< Back
Next >
Finish
Cancel

Driver *	com.mysql.jdbc.Driver
URL *	jdbc:mysql://localhost:3306/bank
Username	root
Password	


Connectors
Output operations definition
Retrieve connector outputs and store them in process or business variables
Add
< Back
Next >
Finish
Cancel

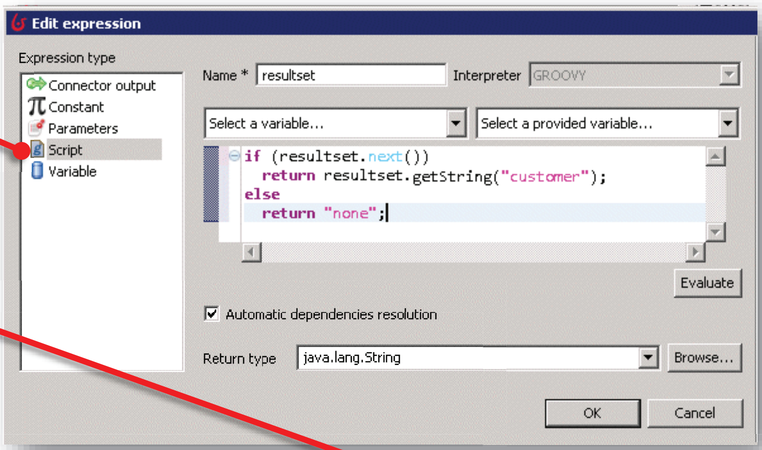
customer — Takes value of — resultset

23. Enter the query
24. *SELECT * FROM account WHERE deposit > ${deposit};*
    (for autocompletion of variables press CTRL + SPACE)
23. Select *Next* > *Scripting Mode* > *Next* > Select target: *customer*
24. Click on the pencil icon to open the Groovy editor.

28. Expression type: *Script*
29. In the text area enter

```
if (resultset.next())
 return resultset.getString("customer");
else
 return "none";
```

28. Click on OK > Finish.
29. *Click on Start button in the coolbar*
30. The Bonita launches the browser
31. Enter a deposit and SUBMIT
32. At Step 2, a customer with more than the deposit will be shown


Edit expression

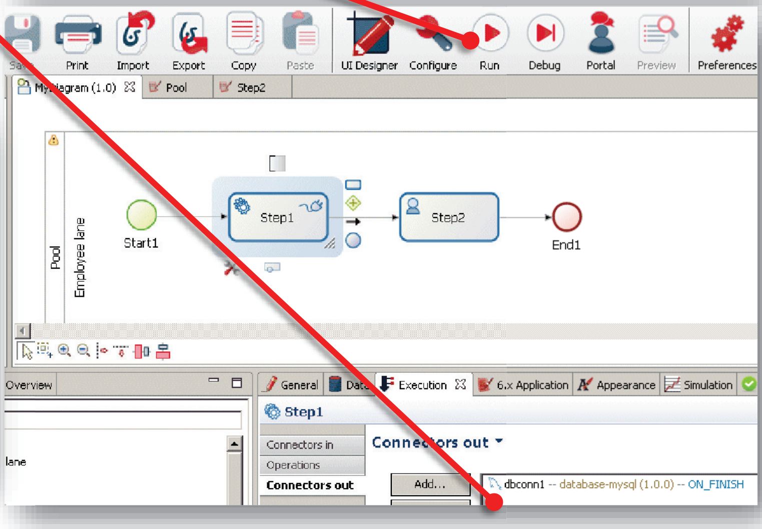
Expression type: Connector output, Constant, Parameters, Script, Variable

Name * resultset   Interpreter GROOVY

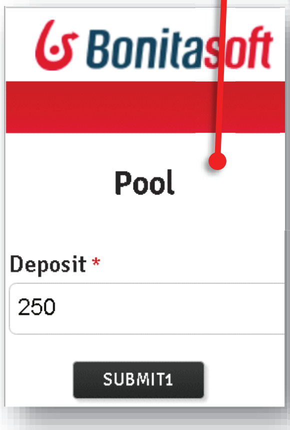
```
if (resultset.next())
   return resultset.getString("customer");
else
   return "none";
```

[✓] Automatic dependencies resolution

Return type java.lang.String

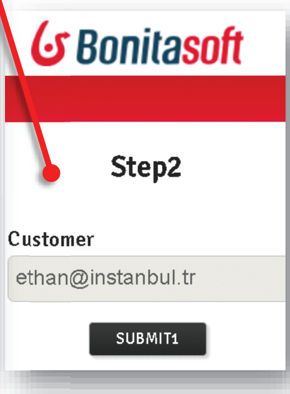


dbconn1 -- database-mysql (1.0.0) -- ON_FINISH


Bonitasoft

Pool

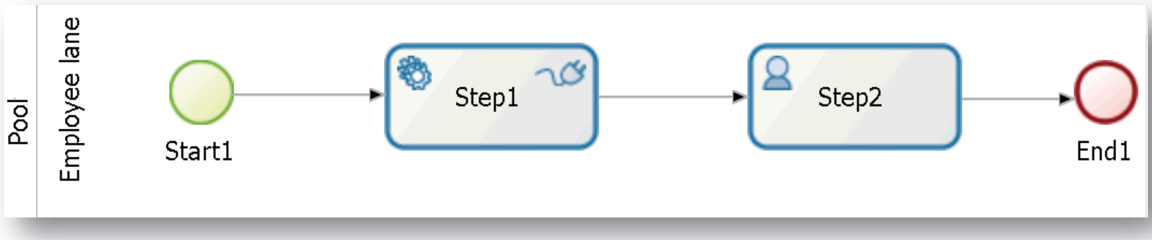
Deposit * 250


Bonitasoft

Step2

Customer ethan@instanbul.tr

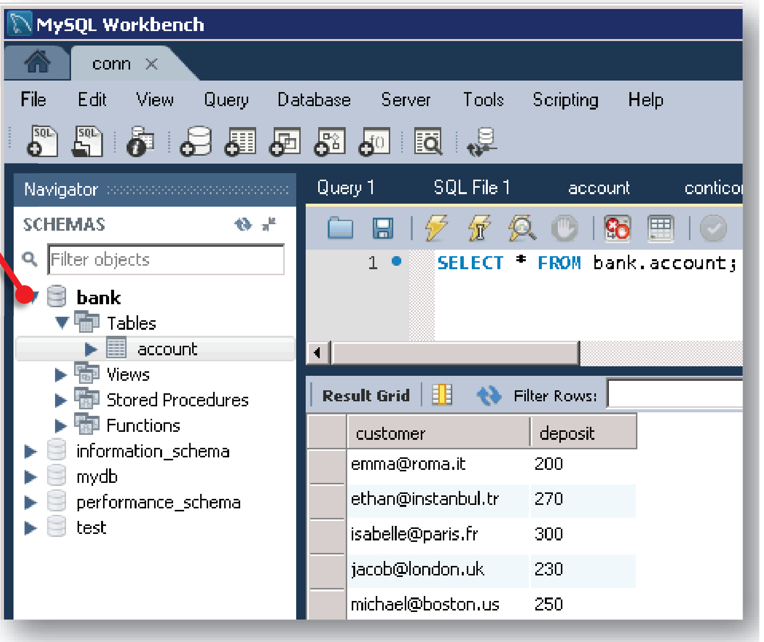
Bonita BPM: Database and Web Service connectors



1. **Create the diagram above (for detailed steps see the first tutorial):**
2. New Diagram > complete the flow with the toolkit leaving the default task types.
3. Select *Step1* > *General* Tab > *Task type:* Service.
4. Select *Step2* > *General* Tab > *Task type:* Human.
5. Click on *Save* in the cool bar.
6. **Create the process variables:**
7. Select *Pool* > *Data* Tab > *Process Variables: Add* > Name: *customer* > *Finish & Add* > Name: *deposit* > *Finish*
8. ***Create the pool form***
9. Select *Pool* > Tab *Execution* > Instantiation form > 6.x
10. Tab *6.x Application* > *Add* > Select Tab *Process variables* > Select *deposit*, and *mandatory* > *Finish*
11. ***Create the Step2 form***
12. Select *Step2* > Tab *Execution* > form > 6.x
13. Tab *6.x Application* > *Add* > Select Tab *Process variables* > Select *customer*, and *read only* > *Finish*

Bonita BPM: Database connector

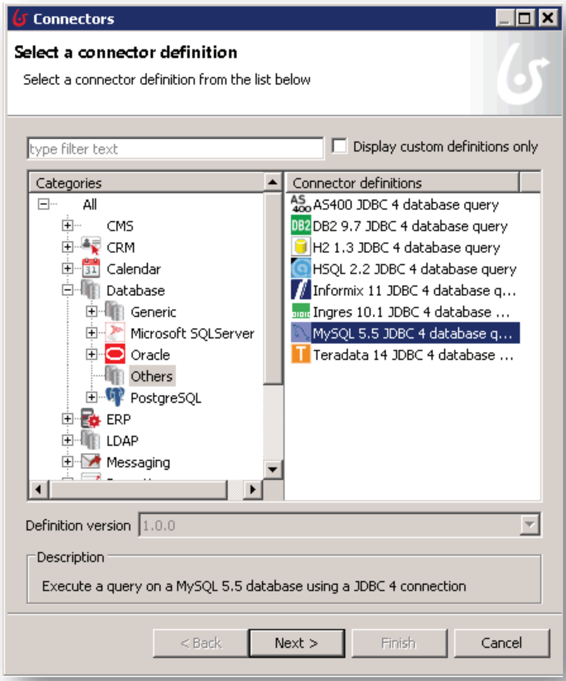
14. **Create the MySQL Database:**
15. 1st method: import the file *bank-dump.sql* into a MySQL server.
16. 2nd method: download the file www.iet.unipi.it/m.cimino/wdis/res/dbms.zip and extract it on C:\wdis. Finally, click on C:\wdis\mysqlStart
17. **Access the Database with MySQL client:**
18. Click on C:\wdis\mysqlClient6.1 > Click on the "+" icon close to MySQL connections > enter a name and click OK.
19. Select the *bank* schema > *Tables* > *account* > right click > Select rows.
20. **Create the DB Connector:**
21. On Bonita, select *Step1* > Tab *Execution* > Connectors out (*) > *Add* > Categories: *Database* > *Others* > *Connector definition* > *MySQL 5.5* JDBC 4… > Next
22. Name: *dbconn1* > *Next.* Enter URL: *jdbc:mysql://localhost:3306/bank* Username: *root* Password: *Next*



```
SELECT * FROM bank.account;
```

customer	deposit
emma@roma.it	200
ethan@instanbul.tr	270
isabelle@paris.fr	300
jacob@london.uk	230
michael@boston.us	250

(*) *Connectors out* are carried out at the end of the step, whereas *Connectors in* at the begin of the process.


Connectors

Select a connector definition

Select a connector definition from the list below

type filter text ☐ Display custom definitions only

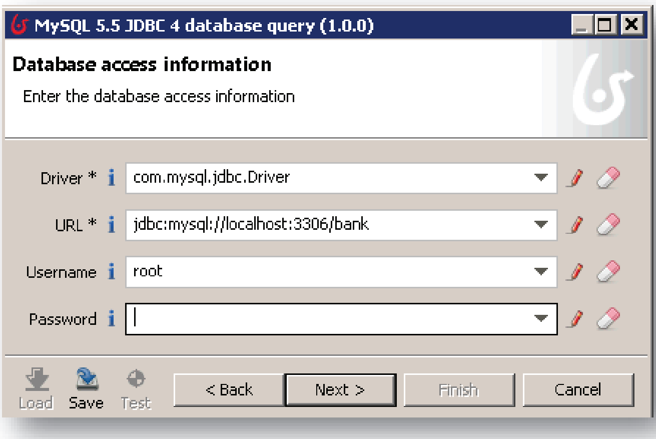
Categories	Connector definitions
All	AS400 JDBC 4 database query
CMS	DB2 9.7 JDBC 4 database query
CRM	H2 1.3 JDBC 4 database query
Calendar	HSQL 2.2 JDBC 4 database query
Database	Informix 11 JDBC 4 database q...
Generic	Ingres 10.1 JDBC 4 database ...
Microsoft SQLServer	MySQL 5.5 JDBC 4 database q...
Oracle	Teradata 14 JDBC 4 database ...
Others	
PostgreSQL	
ERP	
LDAP	
Messaging	

Definition version 1.0.0

Description

Execute a query on a MySQL 5.5 database using a JDBC 4 connection

< Back Next > Finish Cancel


MySQL 5.5 JDBC 4 database query (1.0.0)

Database access information

Enter the database access information

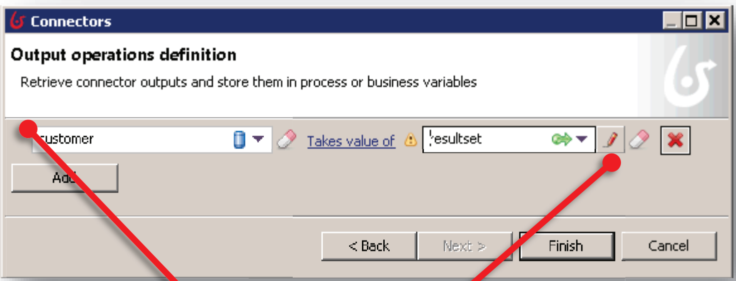
Driver * com.mysql.jdbc.Driver

URL * jdbc:mysql://localhost:3306/bank

Username root

Password

Load Save Test < Back Next > Finish Cancel


Connectors

Output operations definition

Retrieve connector outputs and store them in process or business variables

customer Takes value of resultset

Add

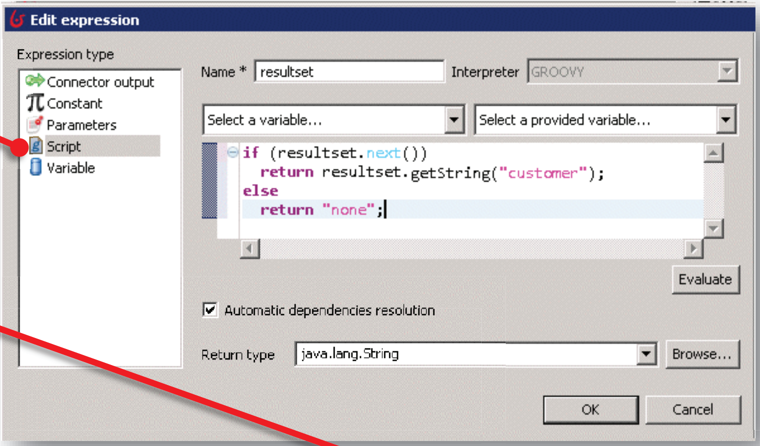
< Back Next > Finish Cancel

23. Enter the query
24. *SELECT * FROM account WHERE deposit > ${deposit};*
    (for autocompletion of variables press CTRL + SPACE)
23. Select *Next* > *Scripting Mode* > *Next* > Select target: *customer*
24. Click on the pencil icon to open the Groovy editor.

28. Expression type: *Script*
29. In the text area enter

```
if (resultset.next())
 return resultset.getString("customer");
else
 return "none";
```

28. Click on OK > Finish.
29. *Click on Start button in the coolbar*
30. The Bonita launches the browser
31. Enter a deposit and SUBMIT
32. At Step 2, a customer with more than the deposit will be shown


Edit expression

Expression type: Connector output, Constant, Parameters, Script, Variable

Name * resultset Interpreter GROOVY

Select a variable... Select a provided variable...

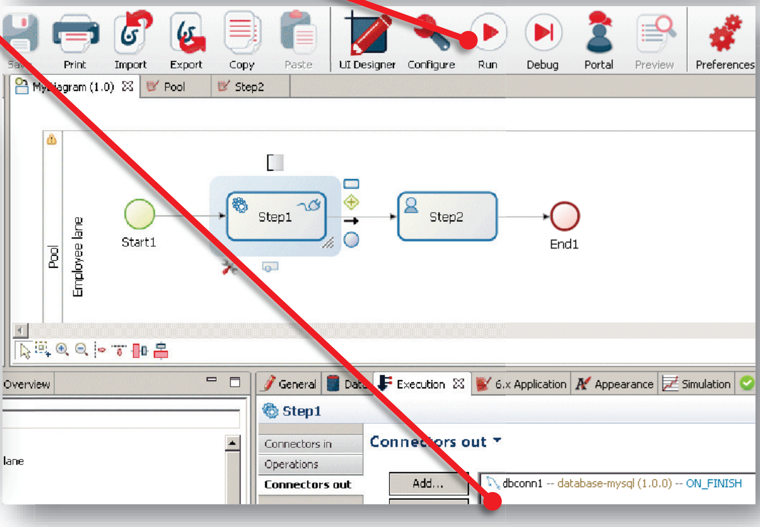
```
if (resultset.next())
    return resultset.getString("customer");
else
    return "none";
```

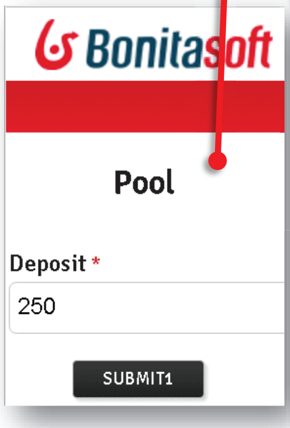
Evaluate

☑ Automatic dependencies resolution

Return type java.lang.String Browse...

OK Cancel



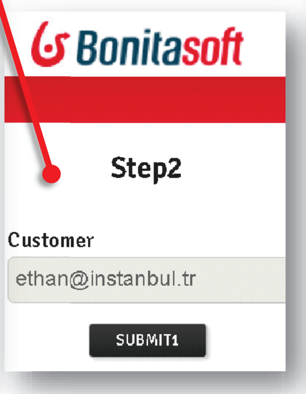

Bonitasoft

Pool

Deposit *

250

SUBMIT1


Bonitasoft

Step2

Customer

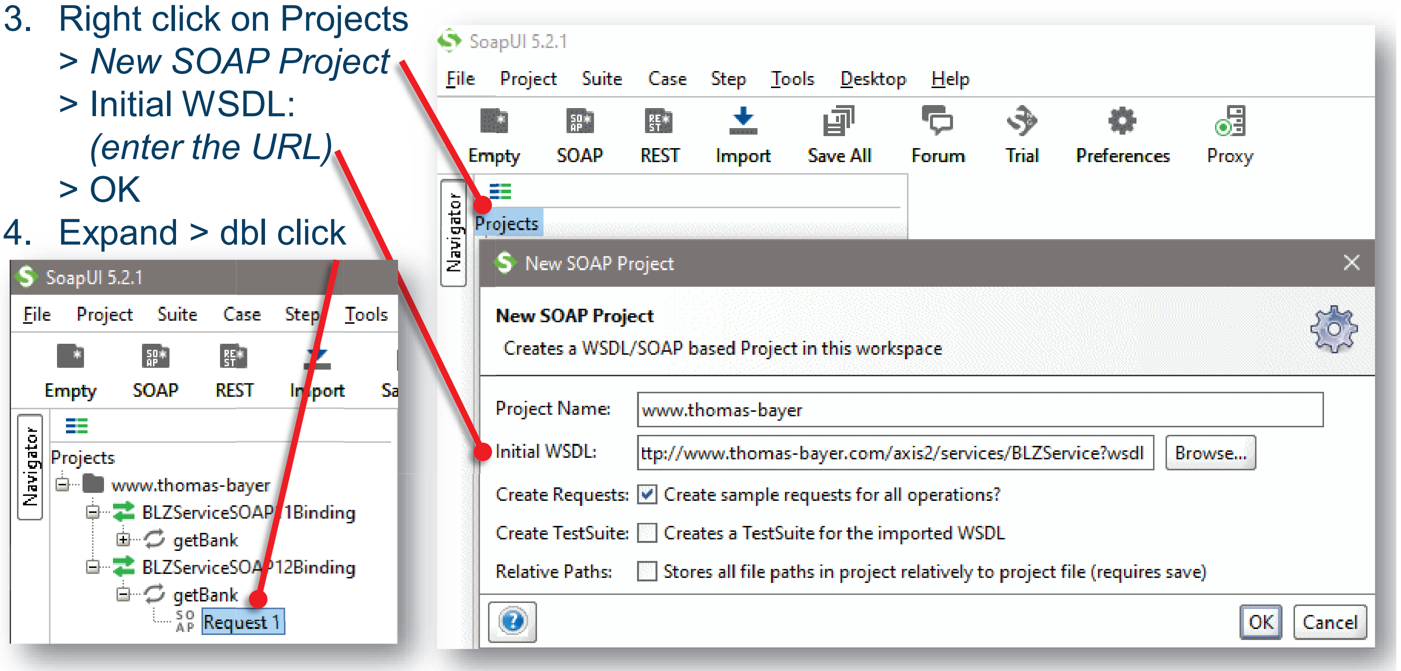
ethan@instanbul.tr

SUBMIT1

Bonita BPM: Web Service connector

Example of Web service:
http://www.thomas-bayer.com/axis2/services/BLZService?wsdl

1. Install the SOAP UI tool:
2. WIN64: http://www.iet.unipi.it/m.cimino/sse/res/SoapUI-x64-5.2.1.exe
   WIN32: http://www.iet.unipi.it/m.cimino/sse/res/SoapUI-x32-5.2.1.exe
   MACOS: http://www.iet.unipi.it/m.cimino/sse/res/SoapUI-5.2.1.dmg
   LINUX: http://www.iet.unipi.it/m.cimino/sse/res/SoapUI-x64-5.2.1.sh
3. Right click on Projects
   > *New SOAP Project*
   > Initial WSDL:
   (enter the URL)
   > OK
4. Expand > dbl click



New SOAP Project

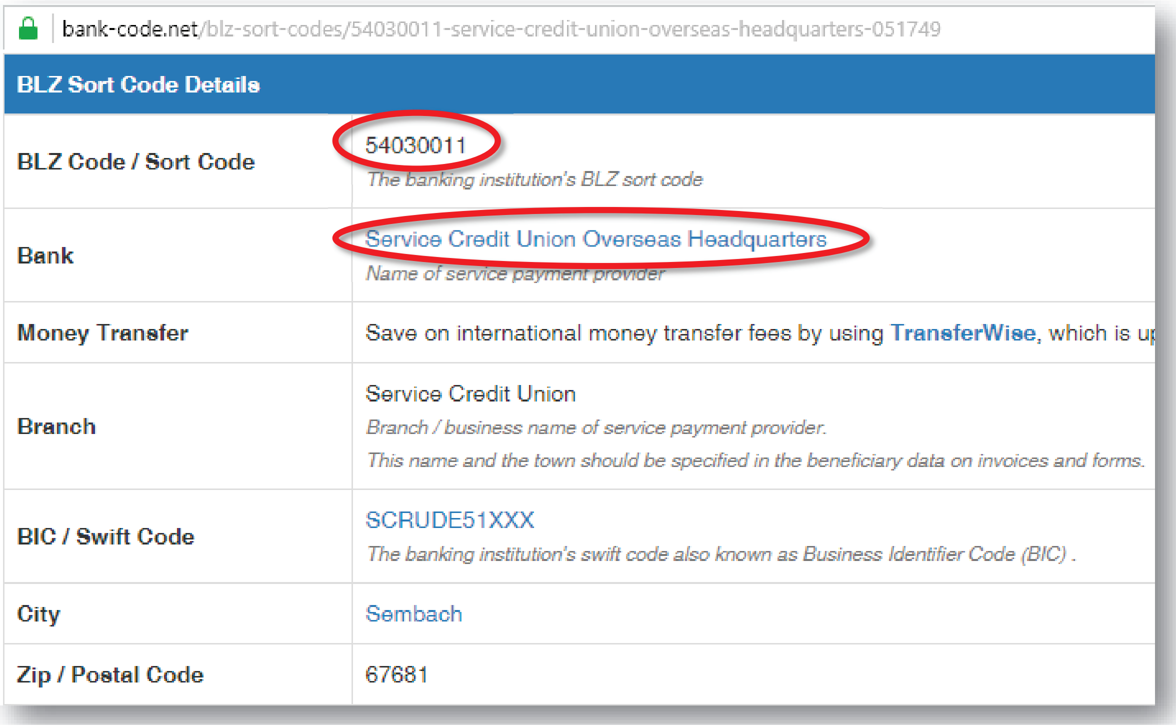
New SOAP Project
Creates a WSDL/SOAP based Project in this workspace

Project Name: www.thomas-bayer
Initial WSDL: ttp://www.thomas-bayer.com/axis2/services/BLZService?wsdl Browse...
Create Requests: ☑ Create sample requests for all operations?
Create TestSuite: ☐ Creates a TestSuite for the imported WSDL
Relative Paths: ☐ Stores all file paths in project relatively to project file (requires save)

OK Cancel

Bonita BPM: Web Service connector

- The service takes the BLZ bank code (used in Germany/Austria, ABI+CAB in Italy, incorporated into the IBAN as part of SEPA standardization) as an input
- Example:
  54030011 the BLZ of the Bank *Service Credit Union Overseas Headquarters*
  https://bank-code.net/blz-sort-codes/54030011-service-credit-union-overseas-headquarters-051749

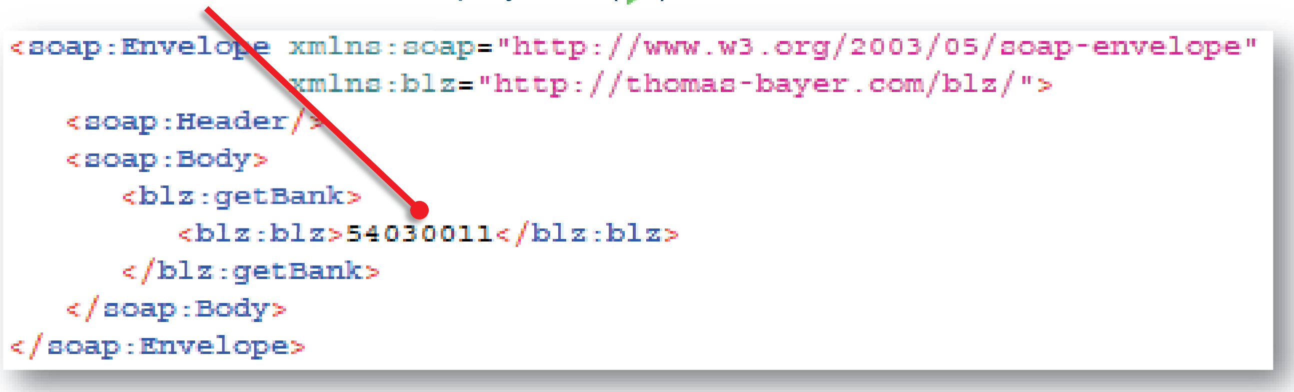


bank-code.net/blz-sort-codes/54030011-service-credit-union-overseas-headquarters-051749

BLZ Sort Code Details	
BLZ Code / Sort Code	54030011 *The banking institution's BLZ sort code*
Bank	Service Credit Union Overseas Headquarters *Name of service payment provider*
Money Transfer	Save on international money transfer fees by using **TransferWise**, which is up
Branch	Service Credit Union *Branch / business name of service payment provider.* *This name and the town should be specified in the beneficiary data on invoices and forms.*
BIC / Swift Code	SCRUDE51XXX *The banking institution's swift code also known as Business Identifier Code (BIC) .*
City	Sembach
Zip / Postal Code	67681

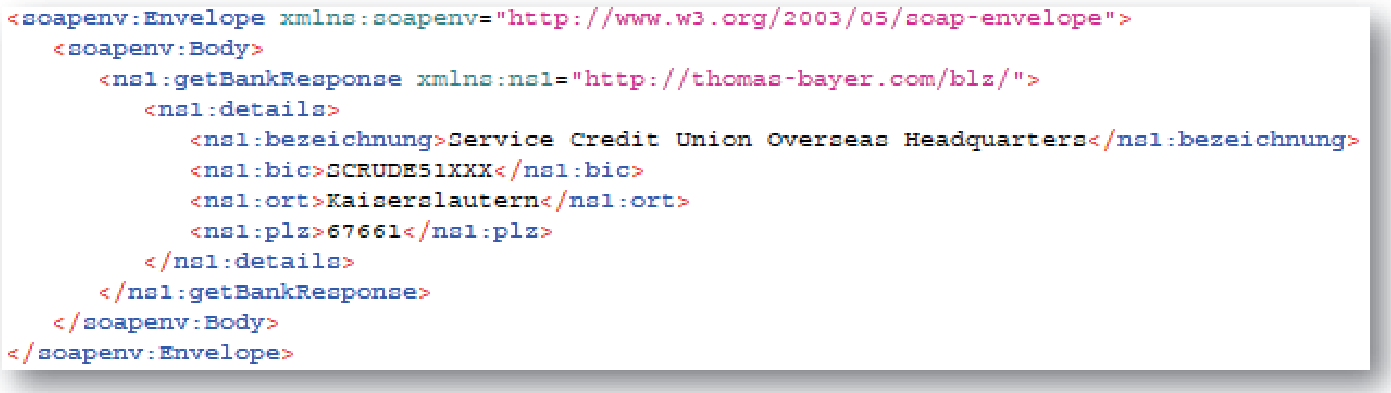
Bonita BPM: Web Service connector

Enter the code and click the play icon (▶)



```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
               xmlns:blz="http://thomas-bayer.com/blz/">
   <soap:Header/>
   <soap:Body>
      <blz:getBank>
         <blz:blz>54030011</blz:blz>
      </blz:getBank>
   </soap:Body>
</soap:Envelope>
```

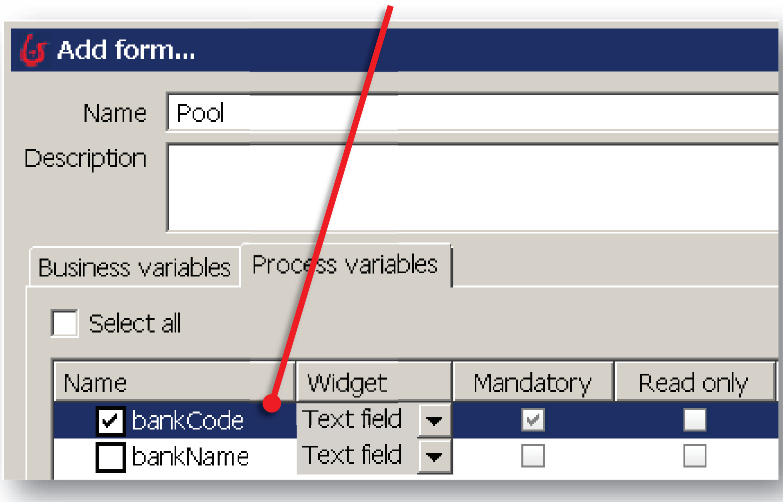
The service provides the following details: bank name (ns1:bezeichnung), BIC code (ns1:bic), place (ns1:ort), and postal code (ns1:plz)



```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
   <soapenv:Body>
      <ns1:getBankResponse xmlns:ns1="http://thomas-bayer.com/blz/">
         <ns1:details>
            <ns1:bezeichnung>Service Credit Union Overseas Headquarters</ns1:bezeichnung>
            <ns1:bic>SCRUDE51XXX</ns1:bic>
            <ns1:ort>Kaiserslautern</ns1:ort>
            <ns1:plz>67661</ns1:plz>
         </ns1:details>
      </ns1:getBankResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Bonita BPM: Web Service connector

1. **Remove the DB connector**
2. Select *Step1* > Tab *Execution* > Connectors out > *Remove*
3. **Remove the Process Variables**
4. Select *Pool* > Tab *Data* > Process variables > select *customer* > Remove > OK, select *deposit* > Remove > OK.
5. **Add the process variables *bankCode (blz), bankName (bezeichnung)***
6. *Add* > Name: *bankCode* > *Finish&Add* > Name: *bankName* > *Finish*
7. **Update the Pool form**
8. Select *Pool* > *6.x Application* > Pageflow > Select *Pool* > *Remove. Add* > *Process variables* > Select *bankCode*. Press *Finish.*



Add form...

Name: Pool

Description:

Business variables | Process variables

☐ Select all

Name	Widget	Mandatory	Read only
✓ bankCode	Text field	✓	☐
☐ bankName	Text field	☐	☐

9. **Update the Step2 form**
10. *Select Step2 > 6.x Application > Pageflow > Select Step2 > Remove. Add > Process variables* > Select *bankName* > *Finish*
11. **Add the WS connector**
12. Select *Step1* > *Execution* > *Connectors out* > *Add.*
13. *Categories: SOAP WebService > Web Service Soap1.2 > Choose the NAME > conn2 > Next*

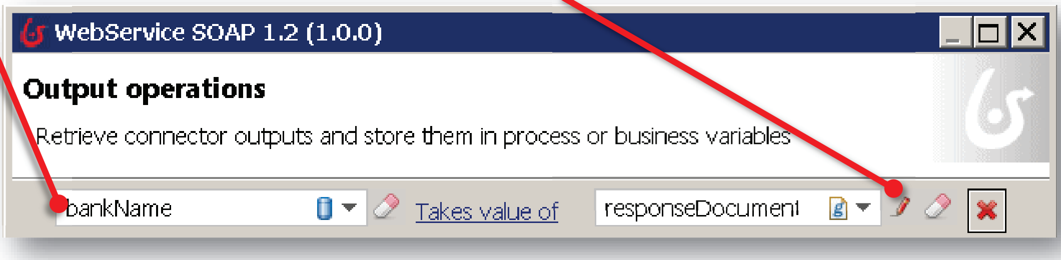
45. Name: *wsconn2* > *Next* > Enter parameters *
    Service NS: *http://thomas-bayer.com/blz/*
    Name: *BLZService*
    Press *Next*
    Port Name: *BLZServiceSOAP12Binding*
    EndPoint: *http://www.thomas-bayer.com/axis2/services/BLZService*
    Binding: *http://www.w3.org/2003/05/soap/bindings/HTTP/*
    Envelope:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
               xmlns:blz="http://thomas-bayer.com/blz/">
   <soap:Header/>
   <soap:Body>
      <blz:getBank>
         <blz:blz>${bankCode}</blz:blz>
      </blz:getBank>
   </soap:Body>
</soap:Envelope>
```

46. Next > Next > Returns body > Next > Output operations:
    (Ctrl + space to find parameters values)

____
(*) Parameters are extracted by WSDL document
http://www.thomas-bayer.com/axis2/services/BLZService?wsdl
and by using a SOAP client software such as *SoapUI*.

47. Next > Next > Returns body > Next > Output operations:
48. Select *bankName* on the left. Click on the pencil icon on the right. Edit Expression: *Script*.


WebService SOAP 1.2 (1.0.0)

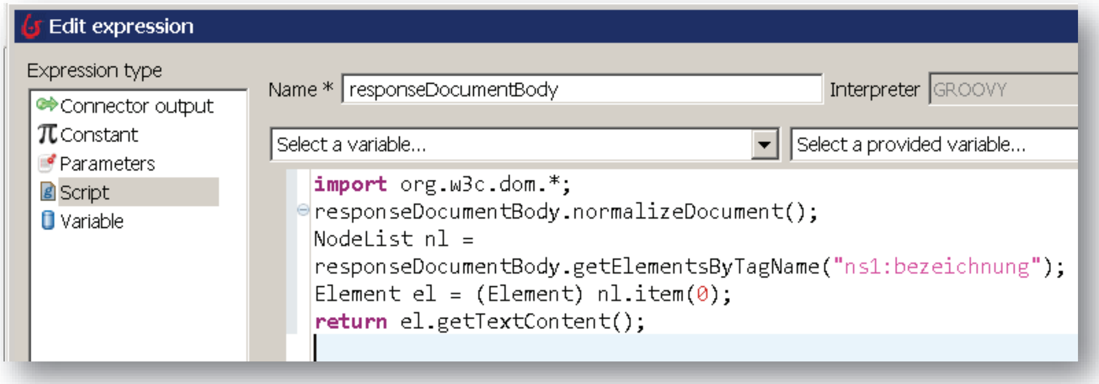
Output operations

Retrieve connector outputs and store them in process or business variables

bankName Takes value of responseDocument

49. In the text area (Ctrl + space to select parameters values if needed):

```
import org.w3c.dom.*;
responseDocumentBody.normalizeDocument();
NodeList nl = responseDocumentBody.getElementsByTagName("ns1:bezeichnung");
Element el = (Element) nl.item(0);
return el.getTextContent();
```


Edit expression

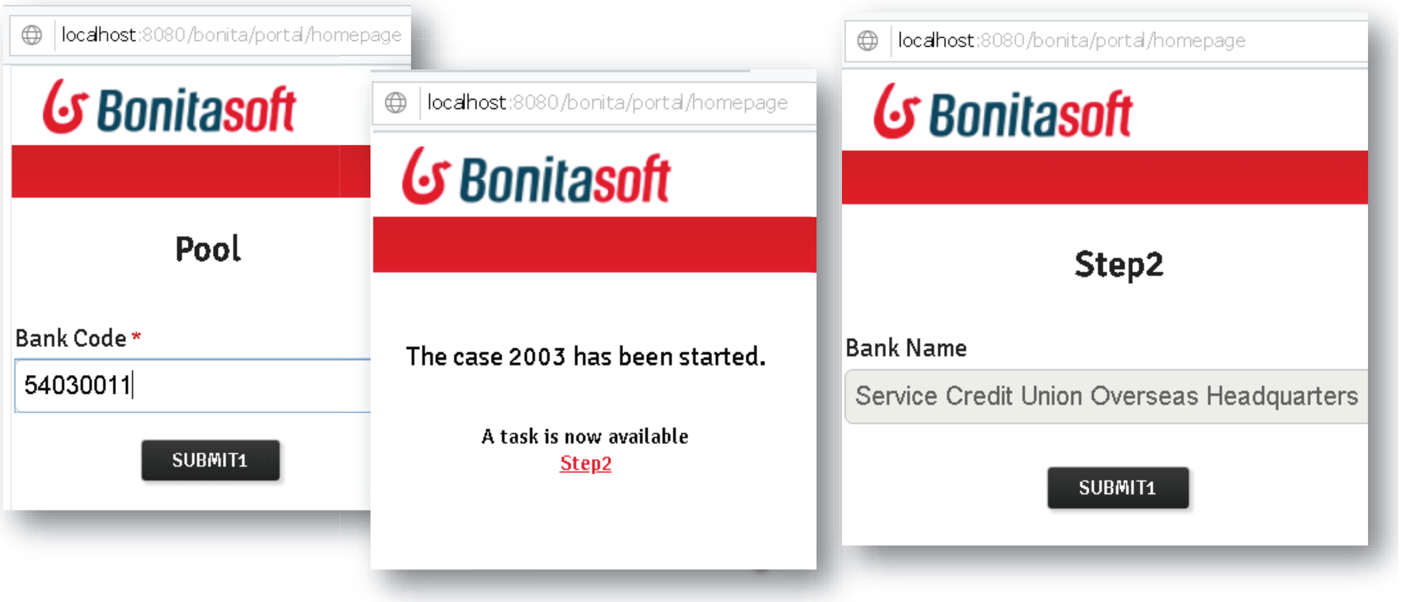
Expression type: Connector output, Constant, Parameters, Script, Variable

Name * responseDocumentBody
Interpreter GROOVY

Select a variable...
Select a provided variable...

```
import org.w3c.dom.*;
responseDocumentBody.normalizeDocument();
NodeList nl =
responseDocumentBody.getElementsByTagName("ns1:bezeichnung");
Element el = (Element) nl.item(0);
return el.getTextContent();
```

49. Click on *Start* button in the coolbar
50. The Bonita launches the browser
51. Enter *Bank Code* and SUBMIT
52. At Step 2, the Bank Name is shown
53. Note: The WS may reply with "-1" when the WS is not available (this my occur for free WS)

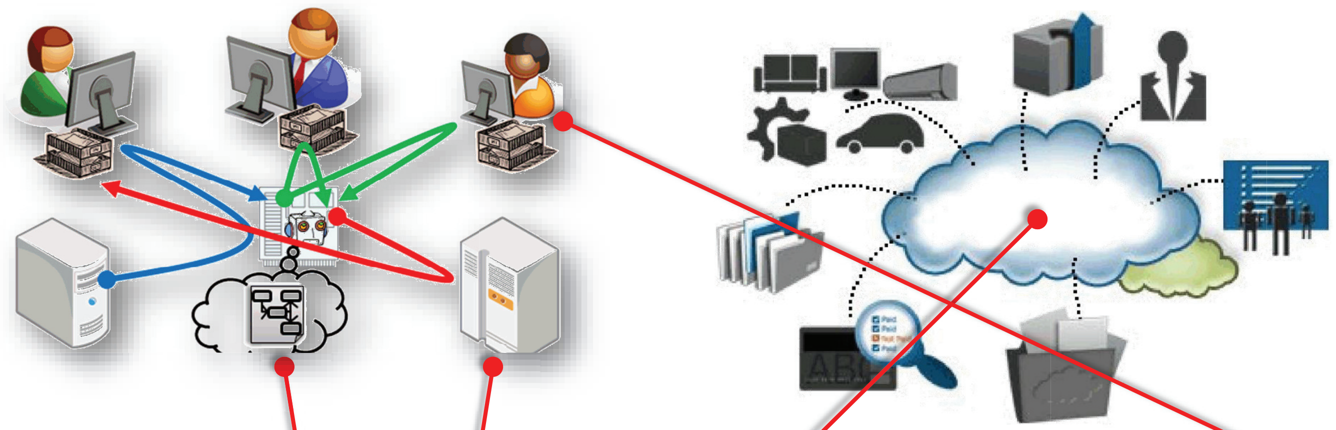

localhost:8080/bonita/portal/homepage
Bonitasoft
Pool
SUBMIT1
Bank Code *
54030011

localhost:8080/bonita/portal/homepage
Bonitasoft
The case 2003 has been started.

A task is now available
Step2

localhost:8080/bonita/portal/homepage
Bonitasoft
Step2
SUBMIT1
Bank Name
Service Credit Union Overseas Headquarters

Evolution of Enterprise System Architectures



✓ A **computer Information System (IS)** is a system, composed of people and computers systems, that processes or interprets information.

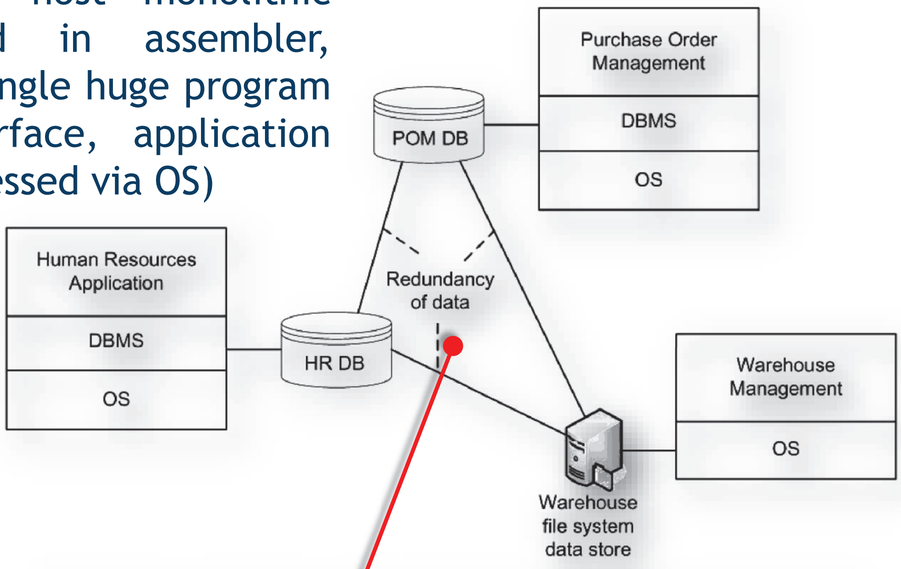
✓ An **Enterprise Information System (EIS)** is an IS which supports enterprise business processes.

✓ A business process is a collection of related, structured activities/tasks producing a specific service/product for a particular kind of customer (customer-centric perspective).

Evolution of Enterprise System Architectures

- 1st era: mainframes host monolithic applications, developed in assembler, managing all tasks in a single huge program with textual user interface, application logic, and data files (accessed via OS)

- With the advent of DBMS and GUI, lowering cost of computers, it is typical for an enterprise to have different applications: for HR, PO, and for Production Planning; each with its own DBMS.


Purchase Order Management
DBMS
OS
POM DB
Human Resources Application
DBMS
OS
HR DB
Redundancy of data
Warehouse Management
OS
Warehouse file system data store

Enterprise applications with redundant data and data dependencies

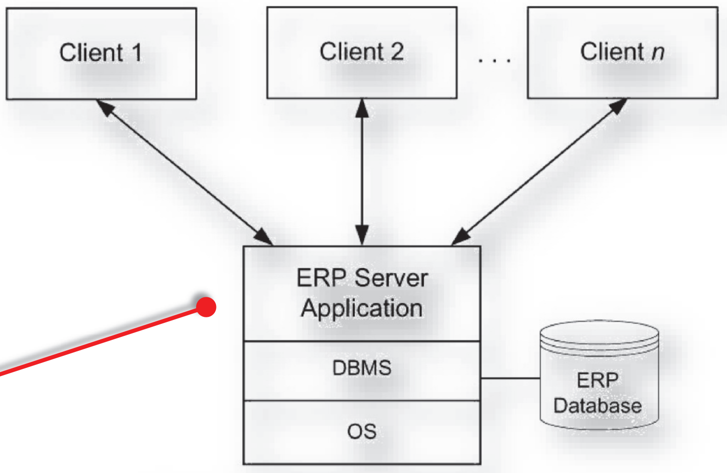
- In large enterprises with different departments, different application systems are used for the same issue.

- Dependencies between data stored in multiple systems are represented by identifiers (e.g. contract id, employee id). However, changes (e.g. a customer address) are hard to propagate without inconsistency.

Evolution of Enterprise System Architectures

- In this context the first Enterprise Resource Planning systems (**ERP**) are developed: to host disparate enterprise applications over an centralized DBMS.

- An ERP is accessed by client applications, which access an application server issuing requests to a DBMS.



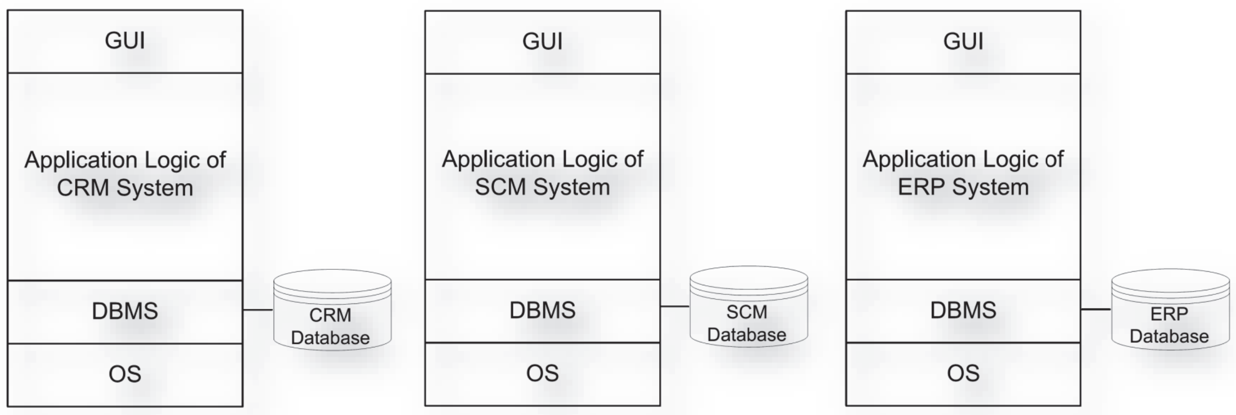
Two-tier client-server architecture

- With the growth of enterprises and new market requirements, driven by new customer needs around the year 2000, new software systems enter in the market:

- Supply Chain Management (**SCM**) systems, Customer Relationship Management (**CRM**) systems, with the purpose of supporting the planning, operation, and control of supply chains, including inventory management, warehouse management, management of suppliers (and distributors) relationships (**SRM**), and demand planning.

- New types of ISs enter the market, often developed by different vendors, hosting their own DBMS. System architects face again the problem of heterogeneous enterprise applications: for instance, call centers are not able to know the complete status of the customer.

- This unsatisfactory situation is called "siloed applications": while application systems can be physically connected by a local network, they are not logically integrated; manual integration made by the user consumes considerable resources and is error-prone.


GUI
Application Logic of CRM System
DBMS
OS
CRM Database
GUI
Application Logic of SCM System
DBMS
OS
SCM Database
GUI
Application Logic of ERP System
DBMS
OS
ERP Database

Siloed enterprise applications

- Unfortunately, due to the large complexity of the systems at hand, the same approach used with ERPs, i.e., to re-implement systems functionality in an integrated way, is not feasible in the new context.

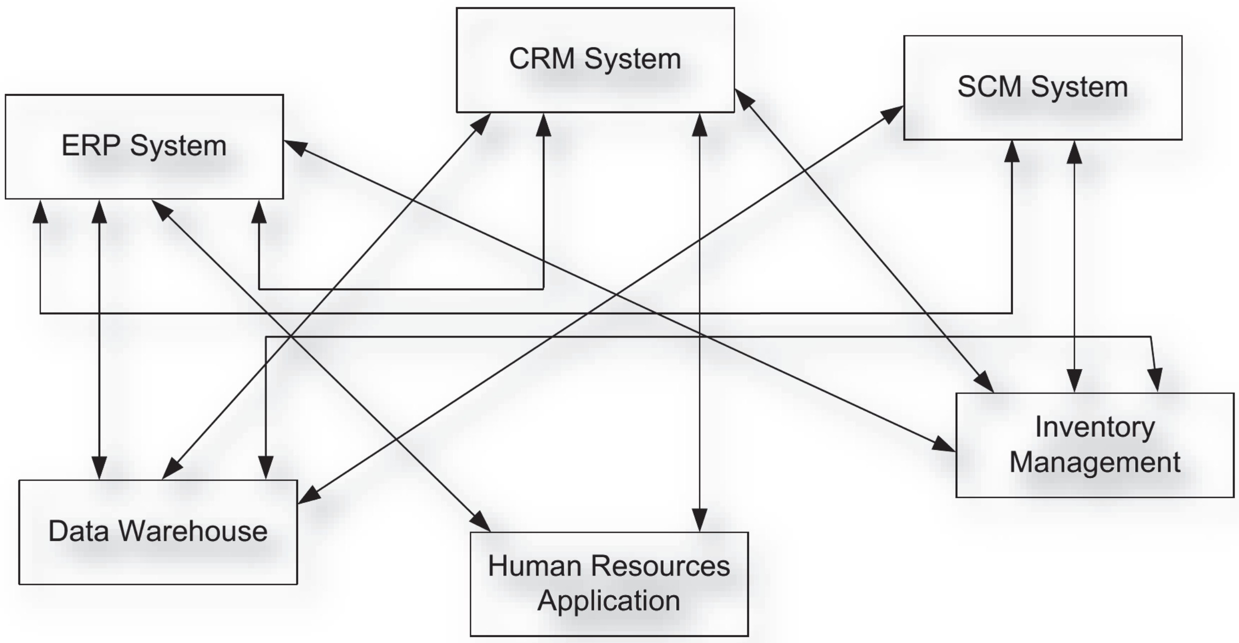
- This leads to new middleware systems: **EAI** (Enterprise Application Integration) systems; in EAI, a system performs certain steps and transfers control to another system, which takes results and continues operation.

- EAI technology can be used to cope with syntactic and semantic differences between data (data integration): e.g. the customer address is represented in one system by the attribute "CAddr" and in the other system by the attribute "StreetAdrC"; e.g. in one system the attribute "Price" includes value-added tax, in the other system it is excluded.

- In enterprise computing changes are abundant, and a system architecture should support changes in an efficient and effective manner.

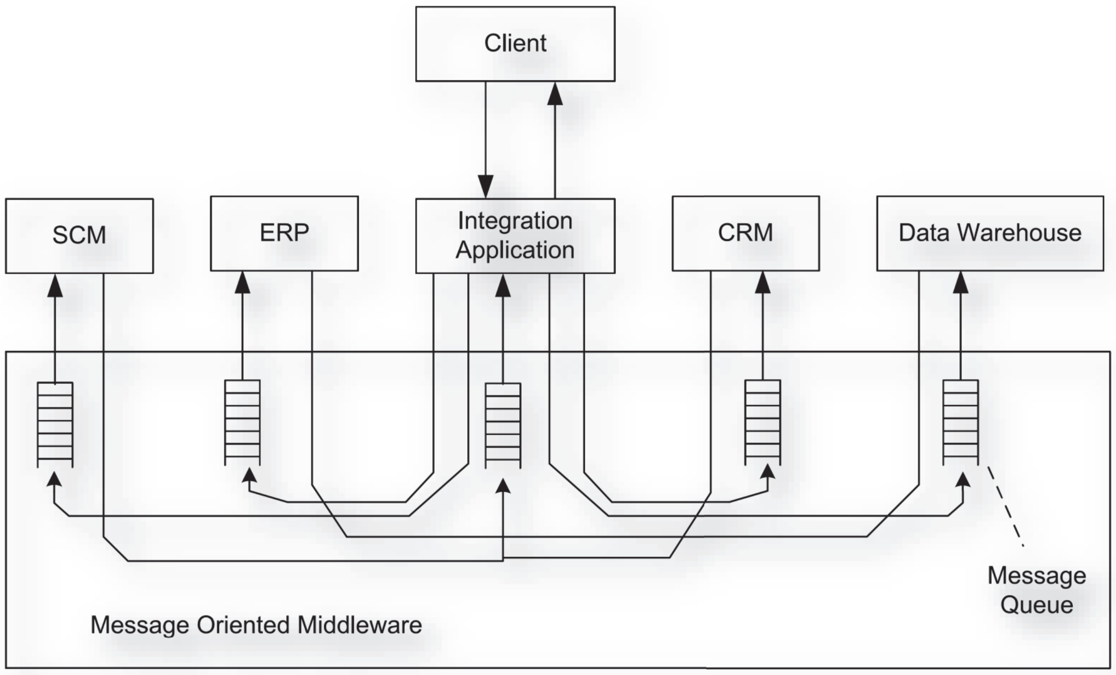
- Early inventions in the EAI architecture are: (i) message oriented middleware; (i) application adapters; (iii) message broker with declarative rules.

Evolution of Enterprise System Architectures



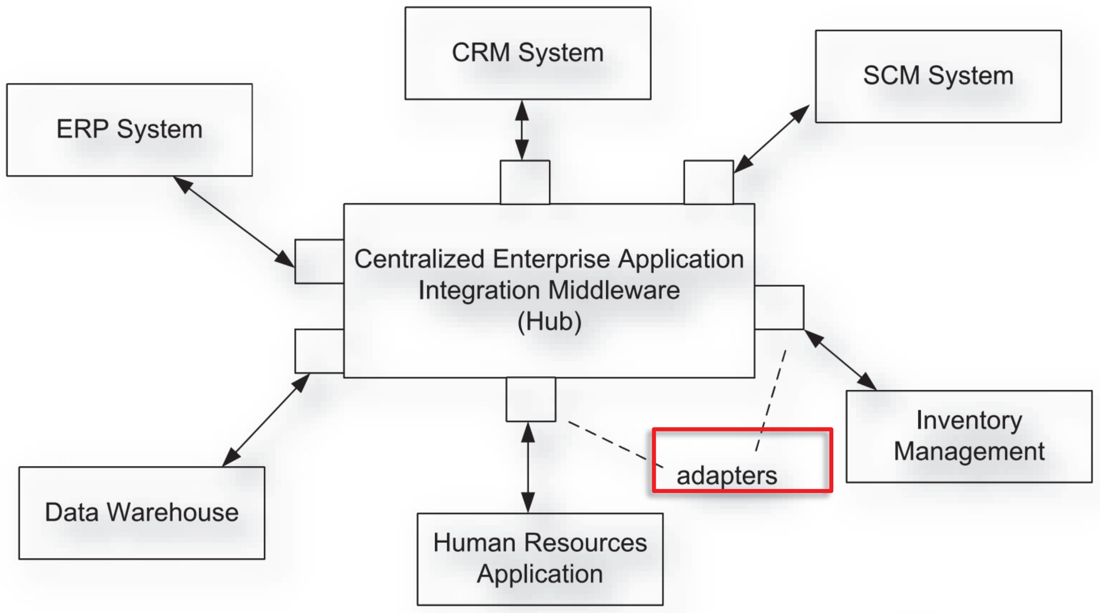
Early enterprise application integration: hard-wiring of application systems results in $N \times N$ problem

Evolution of Enterprise System Architectures



Message-oriented middleware for reliable communication between applications. Senders of messages encode receivers, and process logic is encoded in applications

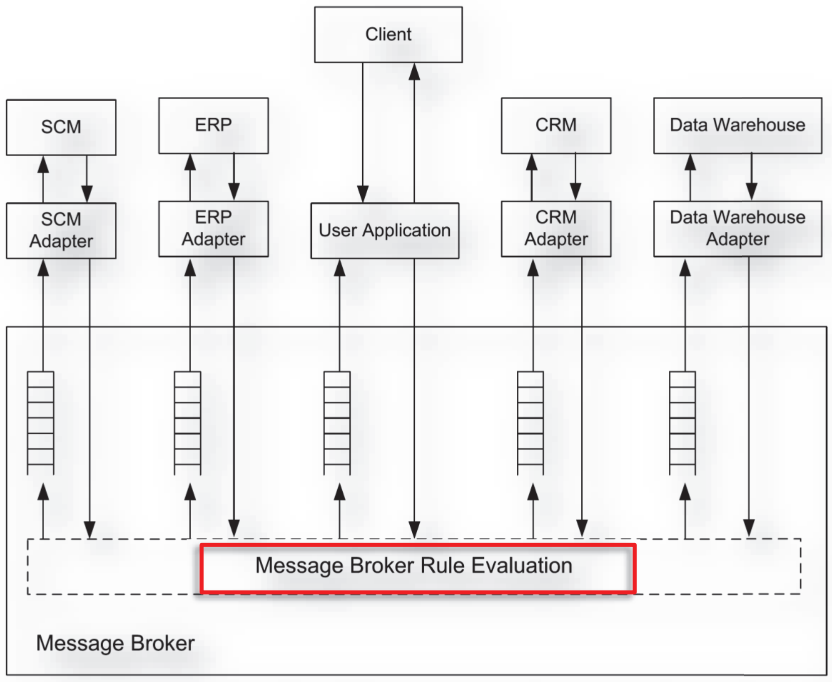
- Guarantee of message delivery. But the problem of point-to-point communication still exists (response to change is not improved): each sender of a message needs to encode the receiver in the integration application.


CRM System
SCM System
ERP System
Centralized Enterprise Application Integration Middleware (Hub)
Inventory Management
adapters
Data Warehouse
Human Resources Application

Hub-and-spoke enterprise application integration architecture

- The sender of a message does not encode the receiver, since the message structure and content is used to automatically detect the receiver or receivers of a message (content-based routing). No *N x N* connections. Each application requires the development of a dedicated adapter.
- Adapters of application systems are used to perform message transformations (data mapping between the applications), and to handle data heterogeneity issues

- Message brokers are used to define rules for communication between applications, in a declarative way, in the central hub.
- Applications can link to message brokers via publish/subscribe mechanism: applications can subscribe to certain types of messages and can publish messages, the enterprise application integration hub uses realize the relaying of messages.



Message broker with declarative rules that de-couples senders from receivers and eases response to change

- Drawback: the message broker contains considerable application logic in rules. This approach requires a global data model hosted by the message broker via programming and low-level configuration of adapters.
- Data integration is typically performed using data mapping tools allowing the mapping of data structures of the application to data structures of the message broker.

From Application Integration to Process Orientation

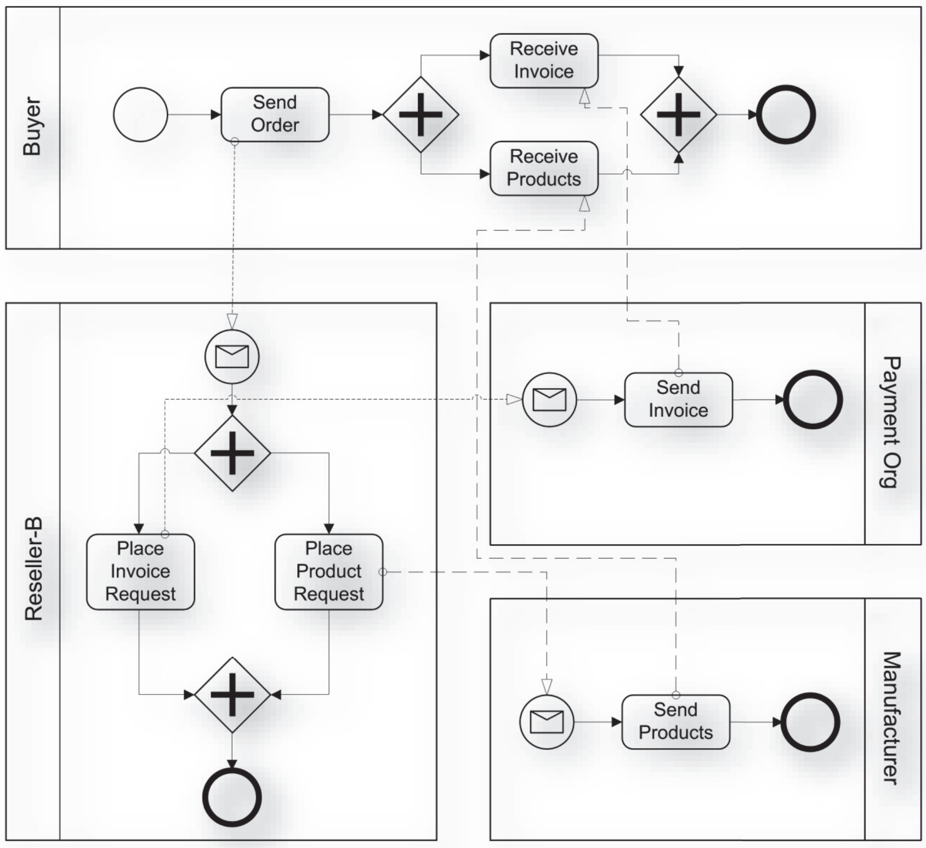
• In typical enterprise application integration scenarios, the functionality of the integrated applications is organized by **a sequence or partial order of steps, realizing a process.** This process consists of activities that are executed, under business constraints, to achieve an overall business goal.

• While in enterprise application integration discussed so far these process structures are embedded in rules hosted by the message broker, an explicit representation of processes is more appropriate. **Workflow management** is the fundamental invention in the evolution of information systems.

• In parallel, another factor emerges from business administration rather than from software technology: **process orientation (PO)**

• PO is based on a critical analysis of **Taylorism** (small-grained activities conducted by highly specialized personnel), which was good until 80s, when products were typically assembled in a few steps of a simple nature (⇒ transfer of work between companies does not introduce delays, no information on previous steps is required)

• In modern business organizations, that mainly process information, the steps during a business process are often related to each other, context information on the complete case is required during the process, and the transfer of work between companies causes a major problem.



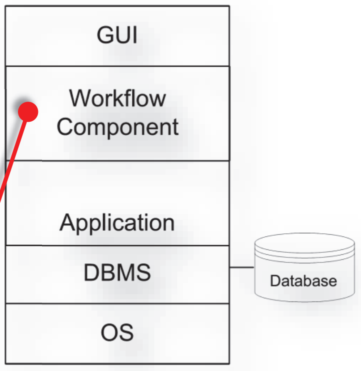
Example of business-to-business collaboration through interacting business processes

Evolution of Enterprise System Architectures

• The important achievement of workflow management is the explicit representation of process structures in process models and the controlled enactment of business processes according to these models.

• The **model-driven approach** facilitates a high degree of flexibility, because process models can be adapted to fulfill new requirements, and the modified process models can immediately be used to enact business processes.

• A **workflow management system (WfMS)** is a software system that defines, creates, and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants, and, where required, invoke the use of IT tools and applications.

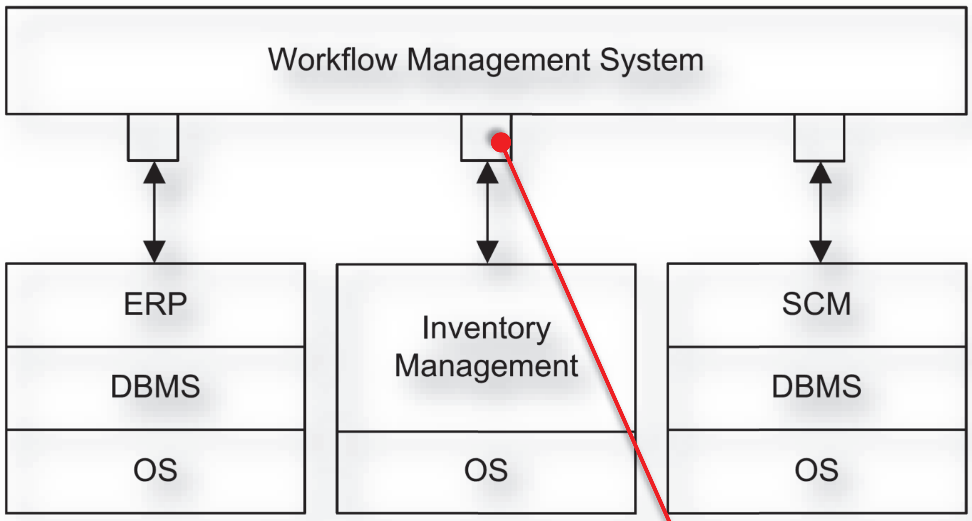


Single-application workflow systems achitecture

• Today, most enterprise application systems, such as ERP, embed a workflow engine (called workflow component) to facilitate the flexible customization of business processes within these systems.

Evolution of Enterprise System Architectures

• In the case of multiple-application workflows, a dedicated workflow management system makes sure that the application systems are invoked as specified in the process model.

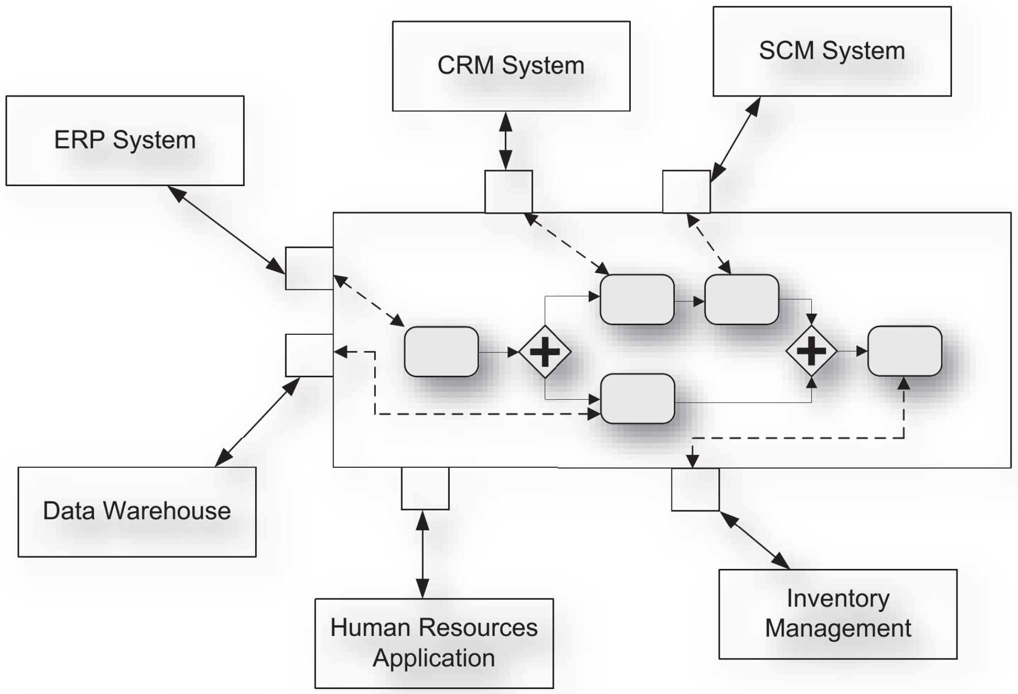


Multiple-application workflow systems architecture

• In addition, data transfer between application systems is also handled by the workflow management system, by using adapters.

Evolution of Enterprise System Architectures

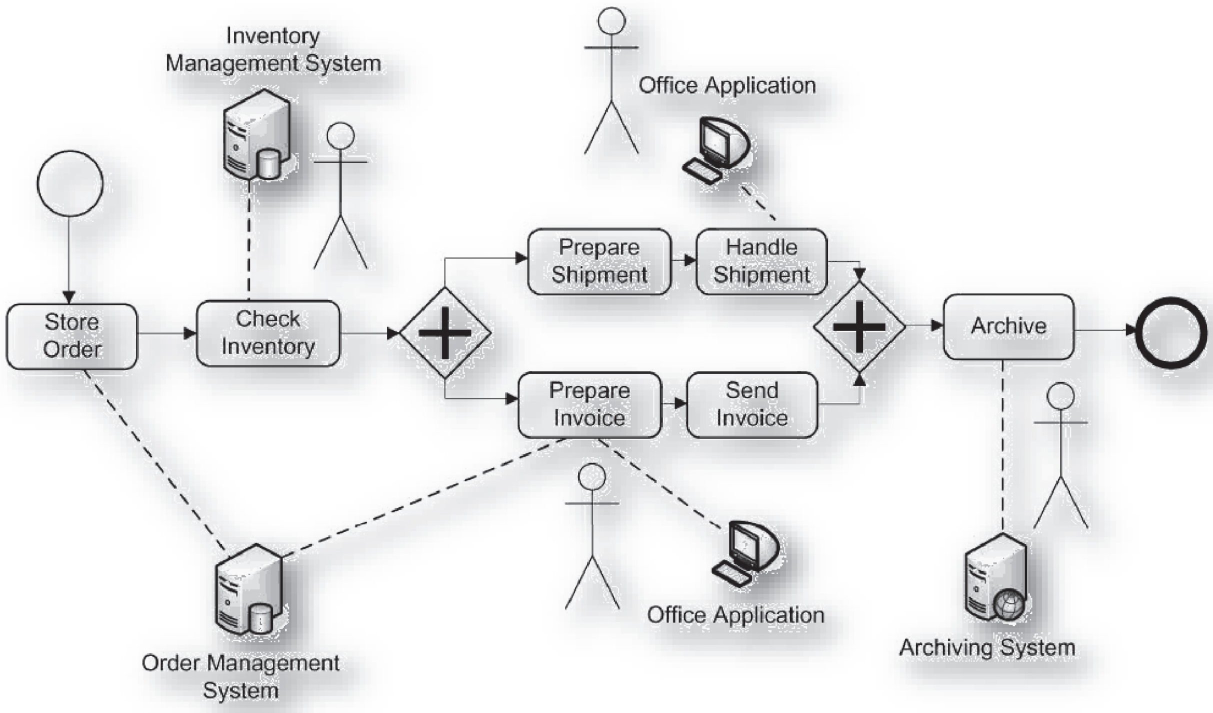
- 1st type of Wf: Systems Workflow, which consists of activities implemented entirely by software systems without any user involvement



System workflow integration scenario; a process model defines if and when enterprise applications are invoked

Evolution of Enterprise System Architectures

- 2nd type of Wf: human interaction workflow, in which humans are actively involved and interact with information systems.



Sample human interaction workflow

Process Support Without Workflow Systems

• Not all environments ask for a WfMS. In cases where no changes to the process structure are envisioned, a coding of the process flow can be an attractive and adequate choice: e.g. store procedures in database administration, print workflow in publishing environments.

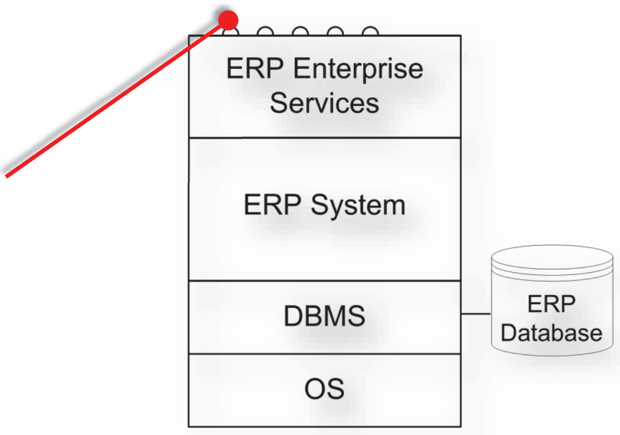
• Business processes are also realized in online shops, such as train reservation systems, where steps of an interaction process are graphically represented to guide the user interaction. Since this type of interaction process can be realized using Web page design, a WfMS is not required.

• ERPs realize literally thousands of BPs, which can be customized to fit particular needs. In most cases, the BPs are realized within the system, without integration issues. In some cases, if the predefined BPs cannot be tailored to fits the needs, integrated process modeling can be used for new processes.

• One of the major trend both in business engineering and software technology is **Service-Oriented Architecture (SOA**) implemented by **Web Services (WS)**.

• A WS is a software whose operations are provided, in a platform independent format (XML), by a host to any another host of the WWW. In a WS, web technology such as the HTTP protocol, originally designed for human-to-machine communication, is utilized for machine-to-machine communication, to invoke software operations and transfer machine readable data (XML).

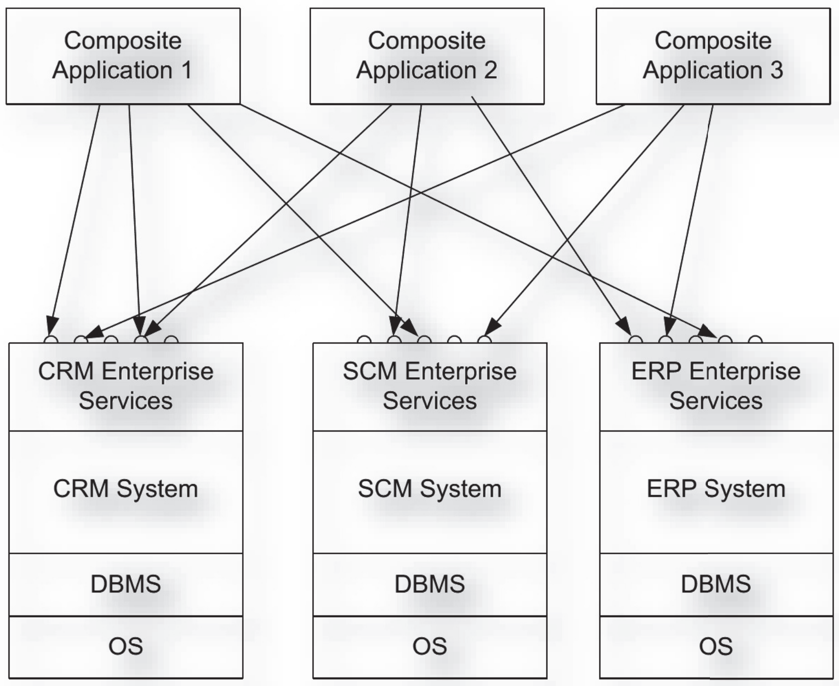
• The functionalities of an enterprise application system can be provided through services (depicted by semicircles in figure) via XML-based standardized interfaces.
Thus, complex applications can be dynamically built on top of existing functionalities.



Service-enabled application system

Evolution of Enterprise System Architectures

- **Composite applications** invoke enterprise services that provide the functionality of the underlying back-end systems. User interaction is realized by dedicated graphical user interfaces that sit on top of composite applications.

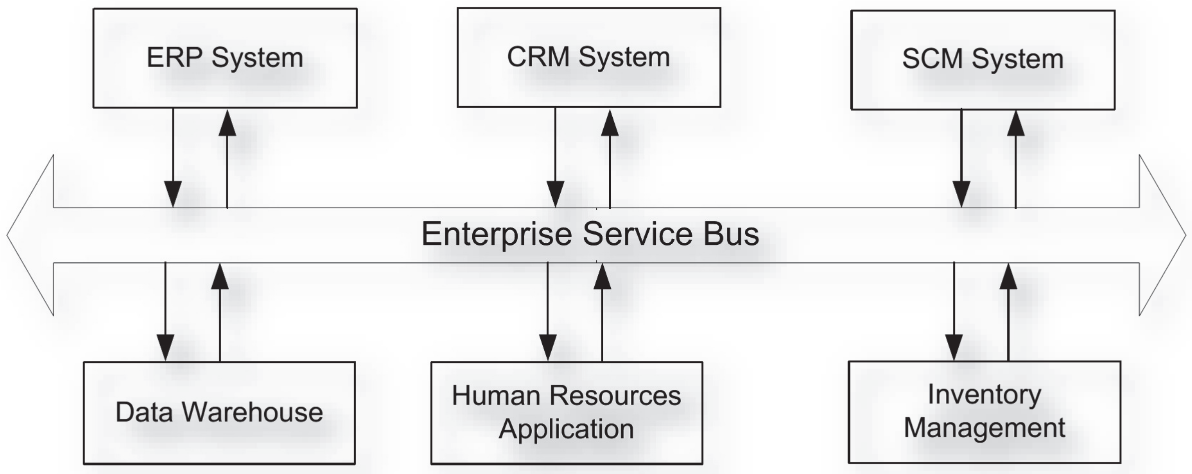


Enterprise systems expose functionality through enterprise services

Evolution of Enterprise System Architectures

- Modern EAI middleware provides Web Services interfaces to the enterprise applications.

- The term **Enterprise Service Bus (ESB)** means that each enterprise application is attached to the bus, which acts as an application independent integration middleware.



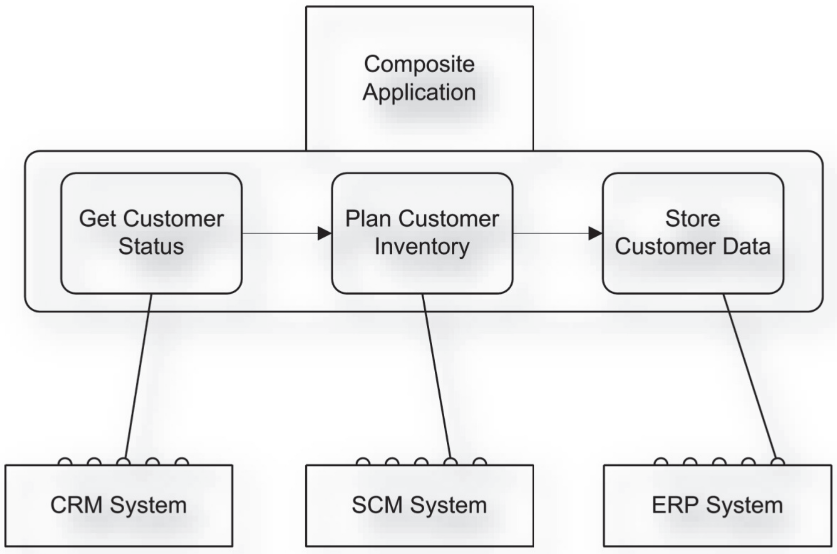
Enterprise service bus

Evolution of Enterprise System Architectures

• The structure of composite applications can in many cases be expressed as a business process.

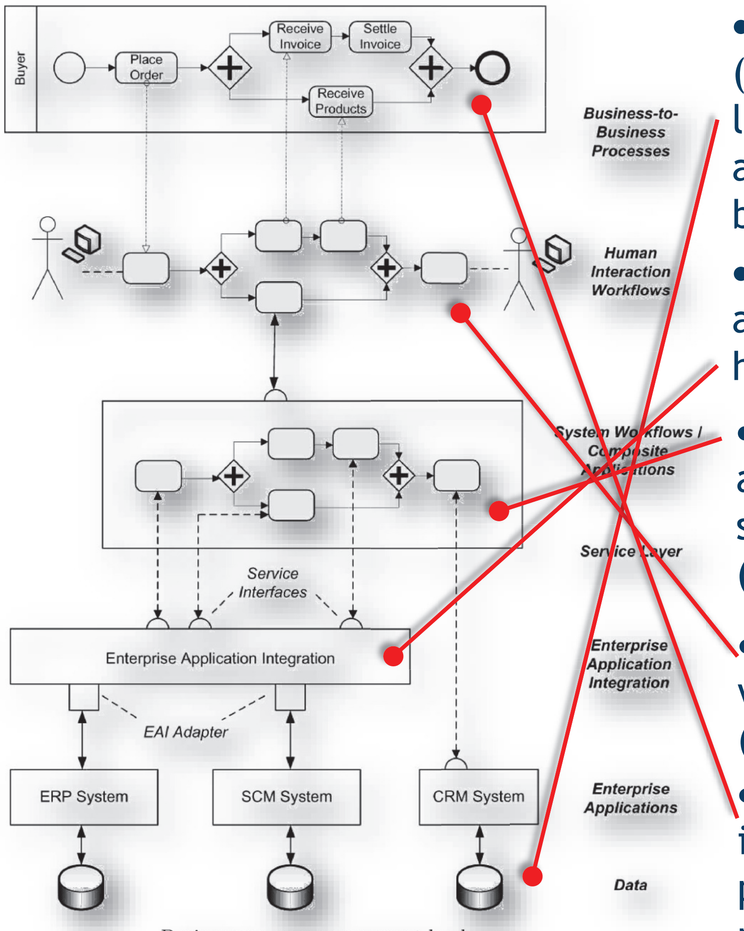
• The activities of these processes are implemented by invoking enterprise services. Additional execution constraints like conditional execution can be represented by business process models

• Enterprise services can also be used to realize business interactions of multiple enterprises (multiple pools).



Using service composition to realize composite applications

Evolution of Enterprise System Architectures


Buyer
Place Order
Receive Invoice
Settle Invoice
Receive Products
Business-to-Business Processes
Human Interaction Workflows
System Workflows / Composite Applications
Service Layer
Service Interfaces
Enterprise Application Integration
EAI Adapter
Enterprise Application Integration
ERP System
SCM System
CRM System
Enterprise Applications
Data

Business process management landscape

• The business process management (BPM) architecture is shown. At the lowest level, heterogeneous applications, such as ERP and CRM, but also tailor-made applications.

• Integration issues are covered by an EAI middleware, via adapters for heterogeneous applications.

• The functionalities of enterprise applications are provided through services to the system workflow (**service tasks**)

• The activities of human interaction workflows can be then associated (**user tasks**)

• Finally, activities in human interaction workflows can also be part of a business-to-business process interaction.

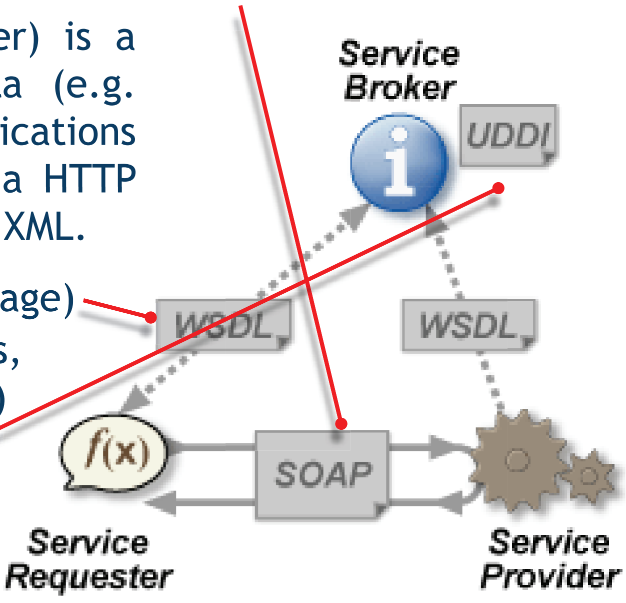
BP Management: Introduction to Web Service

- A **Web Service** (WS) is a software system designed to support interoperable **machine-to-machine** interaction over a network.
- Many companies today offer software on the Web as a service: *Google, Yahoo, Amazon, eBay*.
- Governments collect a lot of data opening up access to data via Web Services (WS). WS technology represents an important way for businesses to communicate with each other and with clients as well.

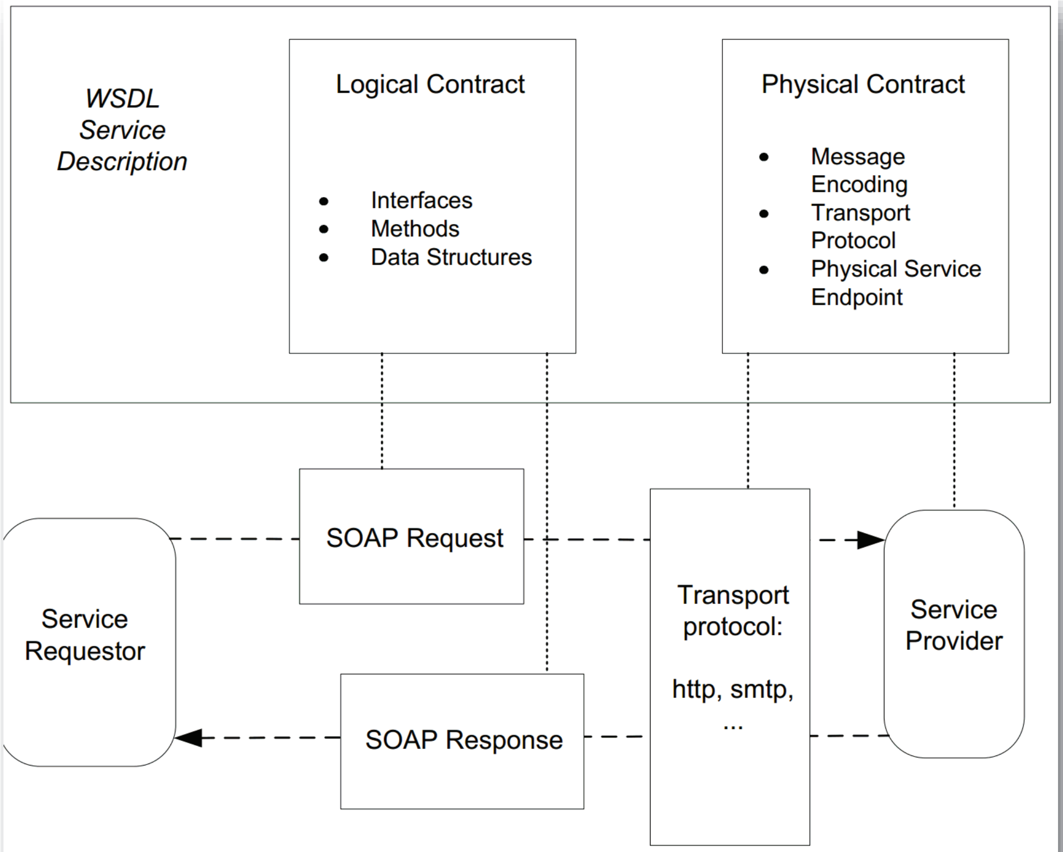
- http://www.webservicex.net/

BP Management: Introduction to Web Service

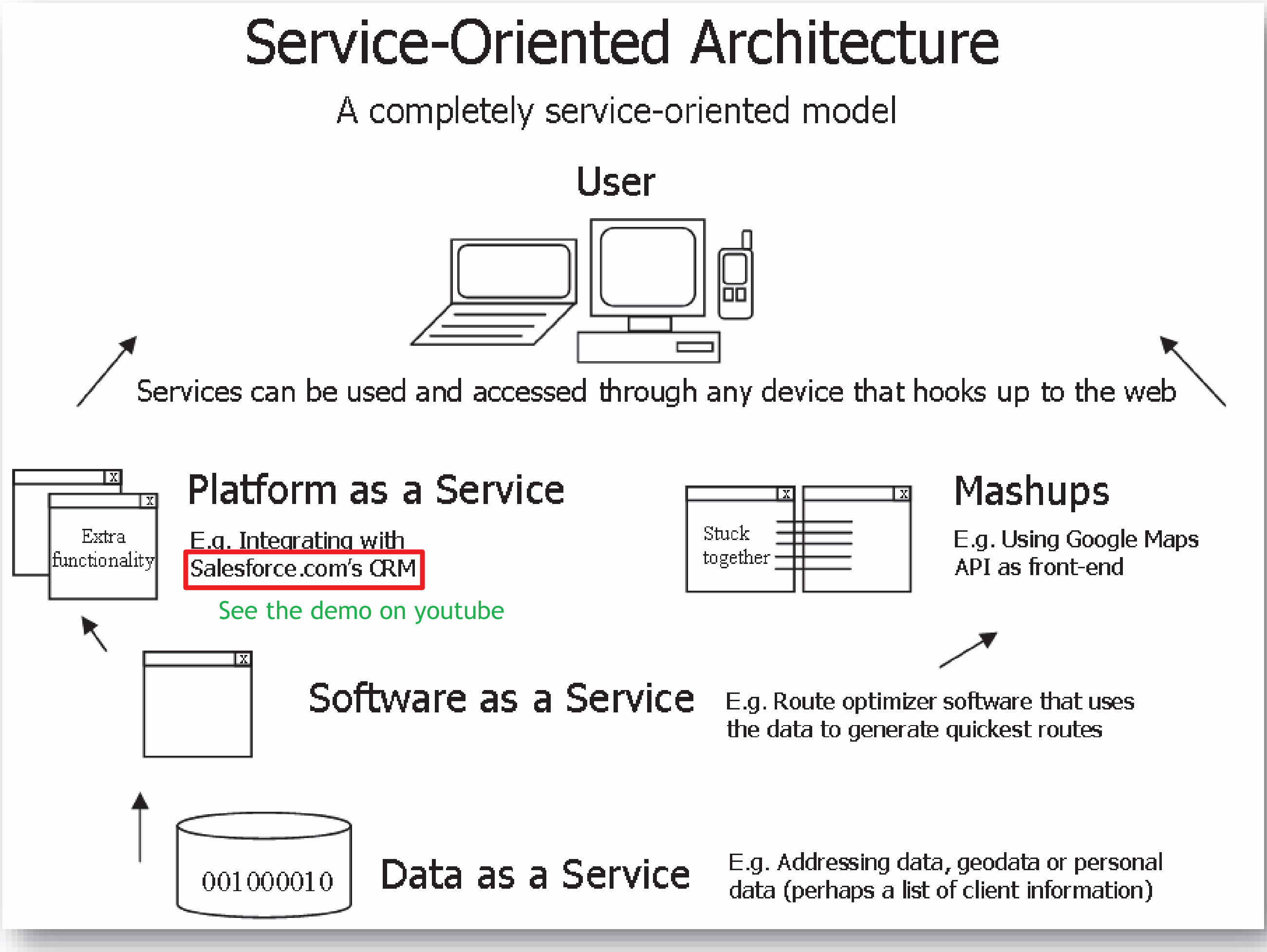
- E.g. a purchase-and-ordering WS communicates to an inventory WS that specific items need to be reordered. Many WS can be chained to implement complicated workflows.
- W3C-XML protocols are used to interact with a WS.
- **SOAP** (Simple Object Access Protocol) over **HTTP** is used to exchange XML messages between the Requester and the Provider of the service.
- REST (REpresentational State Transfer) is a lightweight method to transfer data (e.g. between simple devices/client applications and a WS provider). IT is based on a HTTP request-response message, in JSON or XML.
- WSDL (Web Services Description Language) specifies the WS interface (operations, input/outputs, types, endpoints, ecc.)
- UDDI (Universal Description, Discovery and Integration) provides information to search and access the WS via a Registry



- The WS provider is responsible for preparing a WSDL file of the service
- The WS requestor need access to the WSDL file
- The requestor sends a SOAP message to the service endpoint using information in the WSDL
- The provider invokes the software and returns a SOAP message

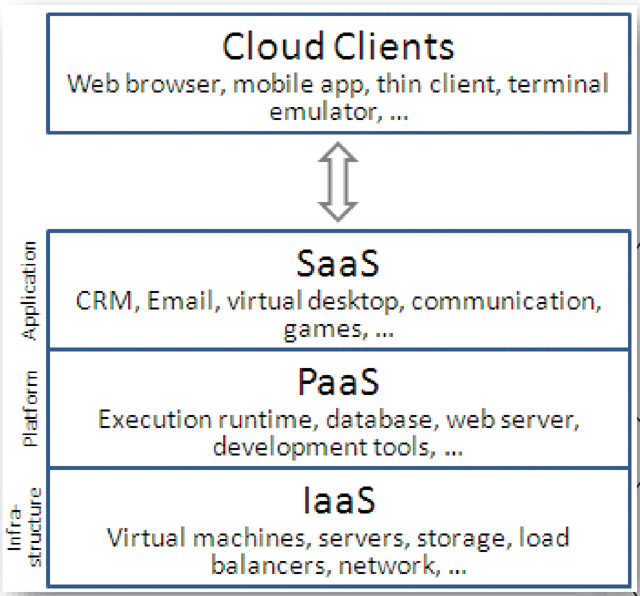


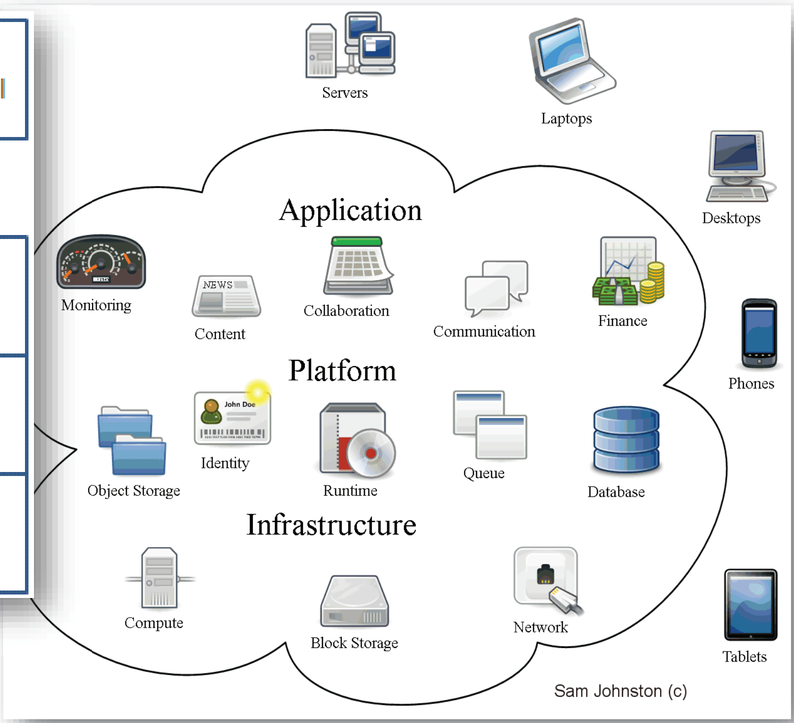
Role of WSDL in Web service invocation


Service-Oriented Architecture
A completely service-oriented model
User
Services can be used and accessed through any device that hooks up to the web
Platform as a Service
Extra functionality
E.g. Integrating with Salesforce.com's CRM
See the demo on youtube
Mashups
Stuck together
E.g. Using Google Maps API as front-end
Software as a Service
E.g. Route optimizer software that uses the data to generate quickest routes
Data as a Service
001000010
E.g. Addressing data, geodata or personal data (perhaps a list of client information)

BP Management: Introduction to Web Service

- The Service-Oriented Architecture (SOA) is one of the enablers of Cloud Computing: Internet-based computing providing on demand resources

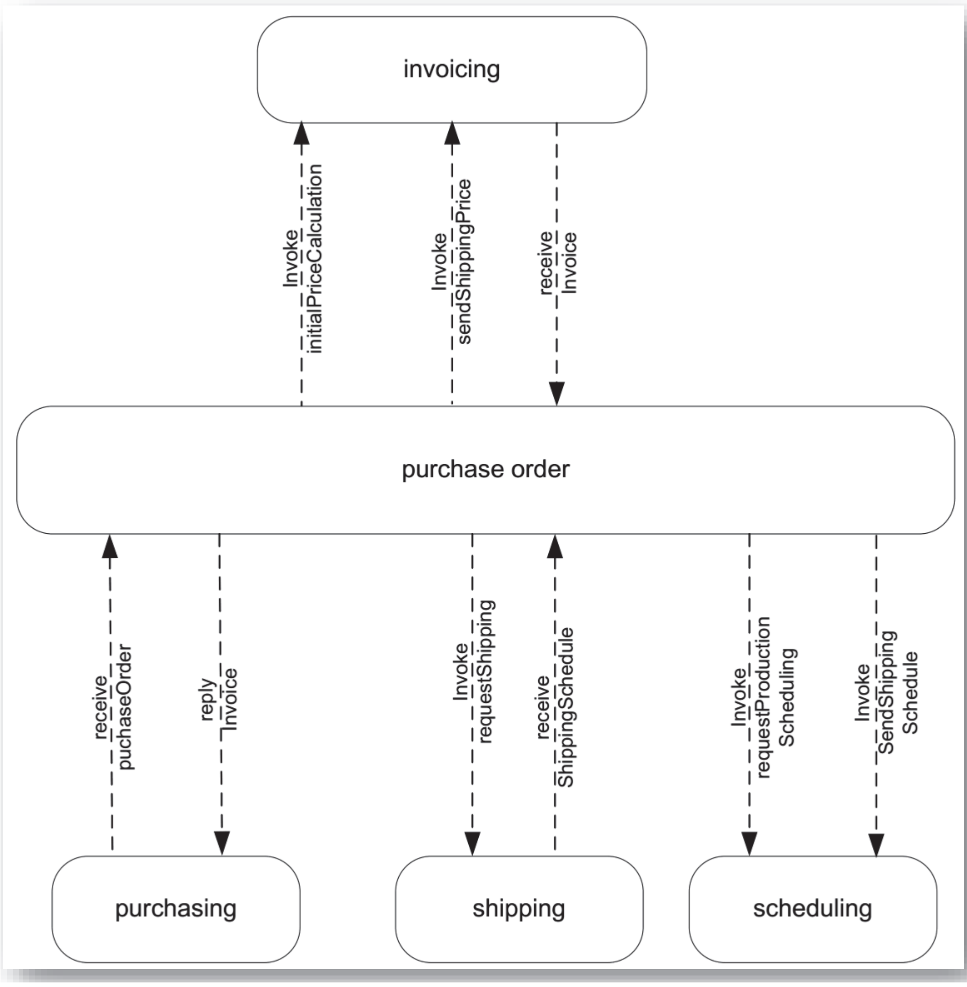

Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...
SaaS
CRM, Email, virtual desktop, communication, games, ...
PaaS
Execution runtime, database, web server, development tools, ...
IaaS
Virtual machines, servers, storage, load balancers, network, ...
Application
Platform
Infra-structure



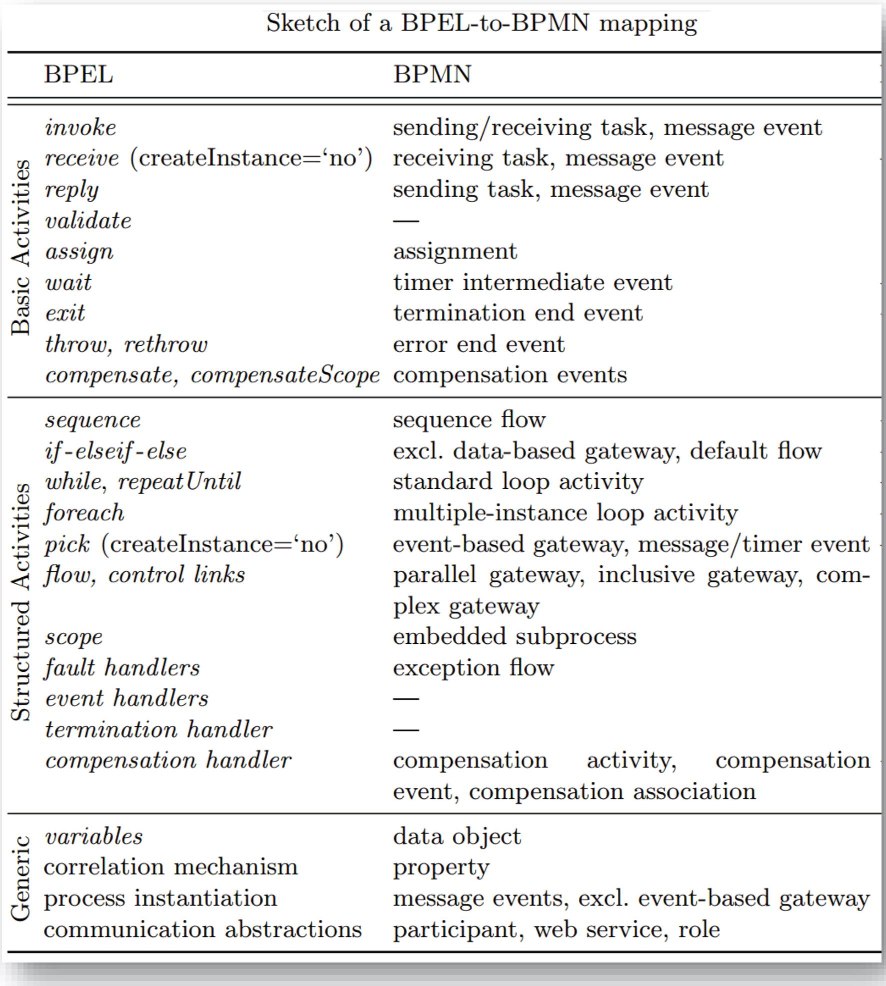
⇒ +Agility, -Cost, -Maintenance, +Performance, +Productivity, +Reliability

BP Management: Introduction to Web Service

- WS composition: describes how a set of services are related to each other. It is an implementation of system workflows.
- **WS-BPEL**: Business Process Execution Language for WS, is a related XML standard
- Example of a high-level purchase order composition showing the communication with each WS.

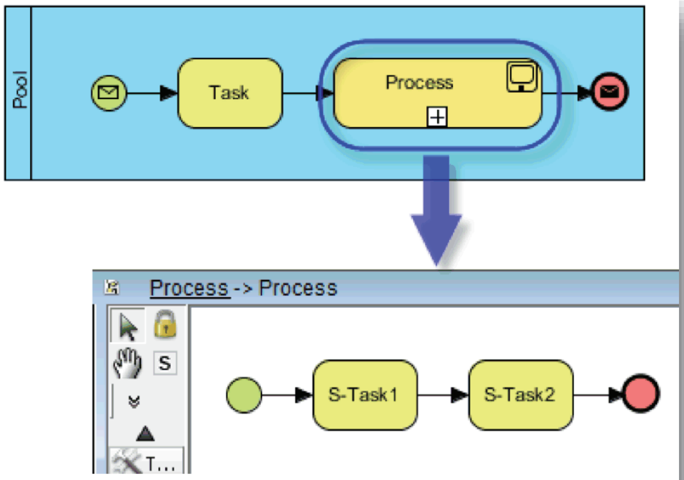


- BPEL can be generated from BPMN, under some limitation
- BPMN is a graph-oriented language in which control and action noted can be connected almost arbitrarily
- BPEL is a mainly block-structured language, an extension of imperative programming languages
- Since BPEL offers loops, if-then-else, XML data types, it is Turing complete


Sketch of a BPEL-to-BPMN mapping

	BPEL	BPMN
Basic Activities	*invoke*	sending/receiving task, message event
	receive (createInstance='no')	receiving task, message event
	reply	sending task, message event
	validate	—
	assign	assignment
	wait	timer intermediate event
	exit	termination end event
	throw, *rethrow*	error end event
	compensate, *compensateScope*	compensation events
Structured Activities	*sequence*	sequence flow
	if-elseif-else	excl. data-based gateway, default flow
	while, *repeatUntil*	standard loop activity
	foreach	multiple-instance loop activity
	pick (createInstance='no')	event-based gateway, message/timer event
	flow, *control links*	parallel gateway, inclusive gateway, complex gateway
	scope	embedded subprocess
	fault handlers	exception flow
	event handlers	—
	termination handler	—
	compensation handler	compensation activity, compensation event, compensation association
Generic	*variables*	data object
	correlation mechanism	property
	process instantiation	message events, excl. event-based gateway
	communication abstractions	participant, web service, role

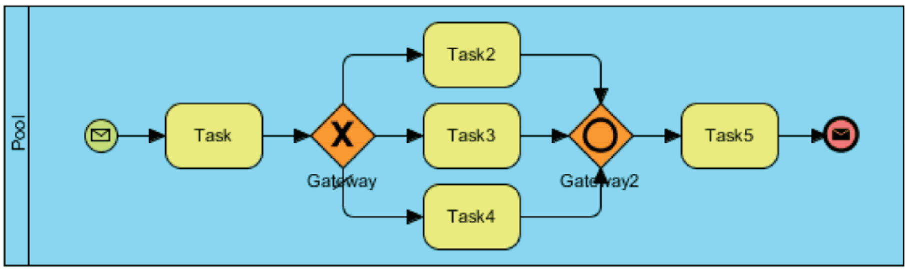
- Examples of BPMN-to-BPEL transformations made by the Visual Paradigm modeler suite
- In the BPEL, the flow modeled in cub-process is merged to the ordinary flow: the activities *STask1* and *STask2* are modeled in the sub-process diagram, following *Task*



```
<?xml version="1.0" encoding="UTF-8"?>
<process name="BusinessProcessDiagram1" targetNamespace="http://BusinessProcessDiagram1" xmlns="...
   <partnerLinks>
      <partnerLink myRole="provider" name="Pool" partnerLinkType="Pool:PartnerLinkType"/>
   </partnerLinks>
   <variables>
      <variable messageType="Pool:Message" name="Variable"/>
   </variables>
   <sequence>
      <receive createInstance="yes" operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType"..
      <empty name="Task"/>
      <empty name="STask1"/>
      <empty name="STask2"/>
      <reply operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
   </sequence>
</process>
```

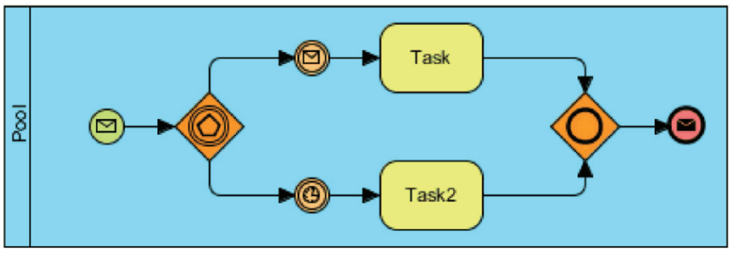
- Example 2: the BPMN XOR gateway is translated into a BPEL switch

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="BusinessProcessDiagram1" targetNamespace="http://BusinessProcessDiagram1" xmlns="http://schemas.xmlsoap.org/ws
business-process/" xmlns:Pool="http://mypool" xmlns:tns="http://BusinessProcessDiagram1" xmlns:xsd="http://www.w3.org/2001/XMLSche
  <partnerLinks>
    <partnerLink myRole="provider" name="Pool" partnerLinkType="Pool:PartnerLinkType"/>
  </partnerLinks>
  <variables>
    <variable messageType="Pool:Message" name="Variable"/>
  </variables>
  <sequence>
    <receive createInstance="yes" operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
    <empty name="Task"/>
    <switch name="Gateway">
      <case condition="true()">
        <sequence>
          <empty name="Task2"/>
        </sequence>
      </case>
      <case condition="false()">
        <sequence>
          <empty name="Task3"/>
        </sequence>
      </case>
      <otherwise>
        <sequence>
          <empty name="Task4"/>
        </sequence>
      </otherwise>
    </switch>
    <empty name="Task5"/>
    <reply operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
  </sequence>
</process>
```



- Example 3: the BPMN event-driven XOR gateway becomes a BPEL pick, which provides two branches, each one with a condition. The branch that has its condition satisfied first is executed.

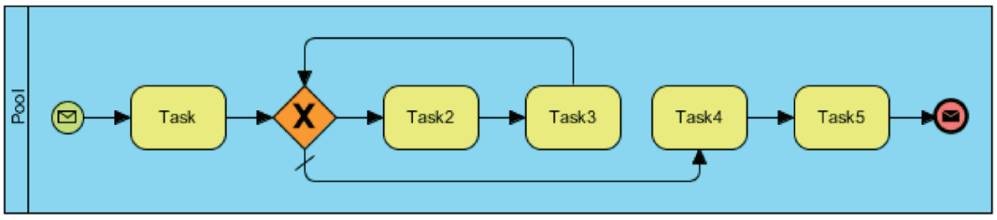
```
<?xml version="1.0" encoding="UTF-8"?>
<process name="BusinessProcessDiagram1" targetNamespace="http://BusinessProcessDiagram1" xmlns="...
business-process/" xmlns:Pool="http://b" xmlns:tns="http://BusinessProcessDiagram1" xmlns:xsd="...
  <partnerLinks>
    <partnerLink myRole="provider" name="Pool" partnerLinkType="Pool:PartnerLinkType" partnerRole="requester"/>
  </partnerLinks>
  <variables>
    <variable messageType="Pool:Message" name="Variable"/>
  </variables>
  <sequence>
    <receive createInstance="yes" operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
    <pick createInstance="no">
      <onMessage operation="continue" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable">
        <sequence>
          <empty name="Task"/>
        </sequence>
      </onMessage>
      <onAlarm for="'PT5S'">
        <sequence>
          <empty name="Task2"/>
        </sequence>
      </onAlarm>
    </pick>
    <invoke inputVariable="Variable" operation="performCallback" partnerLink="Pool" portType="Pool:RequesterPortType"/>
  </sequence>
</process>
```



BP Management: Introduction to Web Service

• Example 4: in BPMN having a sequence flow back into a gateway, forming a loop, means in BPEL to repeat the flow while a condition is satisfied.

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="BusinessProcessDiagram1" targetNamespace="http://BusinessProcessDiagram1" xmlns="http://schemas.xmlsoap.org/ws/...
business-process/" xmlns:Pool="http://mypool" xmlns:tns="http://BusinessProcessDiagram1" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <partnerLinks>
    <partnerLink myRole="provider" name="Pool" partnerLinkType="Pool:PartnerLinkType"/>
  </partnerLinks>
  <variables>
    <variable messageType="Pool:Message" name="Variable"/>
  </variables>
  <sequence>
    <receive createInstance="yes" operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
    <empty name="Task"/>
    <while condition="false()">
      <sequence>
        <empty name="Task2"/>
        <empty name="Task3"/>
      </sequence>
    </while>
    <empty name="Task4"/>
    <empty name="Task5"/>
    <reply operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
  </sequence>
</process>
```



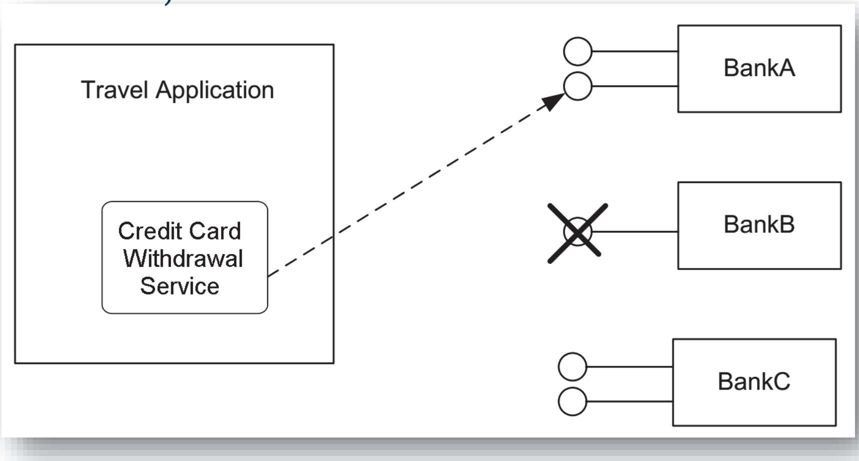
BP Management: Introduction to Web Service

- Let us consider an example of a **travel application**, allowing customers to select trips, make reservations, and confirm reservations by providing credit card information.
- To allow this composition of services, the travel application invokes a credit card withdrawal service provided by a bank. In static binding the two services are bound at **development time.** However, this is not effective in environments with dynamic service landscape

• In dynamic binding, service implementations can be discovered at runtime: the application (or a supporting middleware) asks the service registry for a list of suitable services, selects and binds to one of them.

• Service matchmaking: the process of selecting a set of services that fit a service request. It is made by the service broker.

• It depends on rich semantic annotation of services

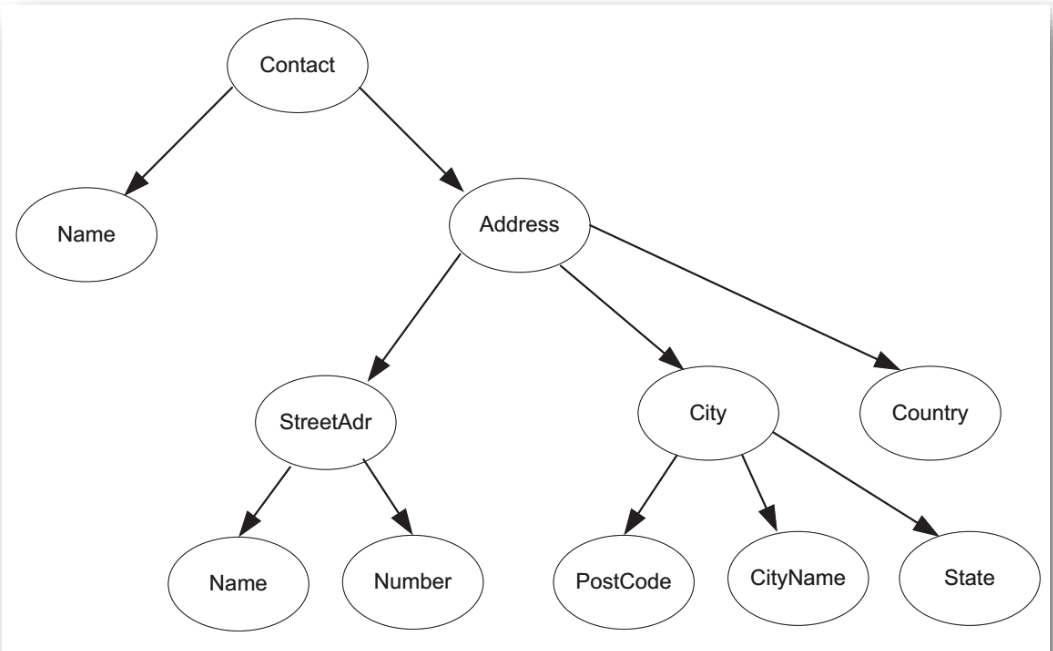


BP Management: Introduction to Web Service

- Services can be composed in a correct way if they operate on the same domain concepts. The simplest case occurs when consequent services operate on the same domain concept: e.g. a service returning customer data can be combined with a service taking customer data as an input.
- Since web services are usually developed independently of each other, the WSDL data types in most cases do not match.
- Typically syntactic differences are solved by system architects and software developers, using data mapping techniques. With compositions of many services this approach introduces a considerable overhead due to heterogeneous data types.
- Even with similar parameter names like *europrice* and *price*, the user of the service cannot be sure that the price is really the euros currency. For example there might be semantic differences: one service returns 120 and the other 118. Since the concept of price is not agreed upon the providers, the price 120 includes value-added tax (VAT), while the price 118 does not.
- To solve this problem, data should be semantically annotated by using **Semantic Web** standard, to be automatically compared and integrated. (RDF, resource Description Framework, OWL: ontology Web langauge)

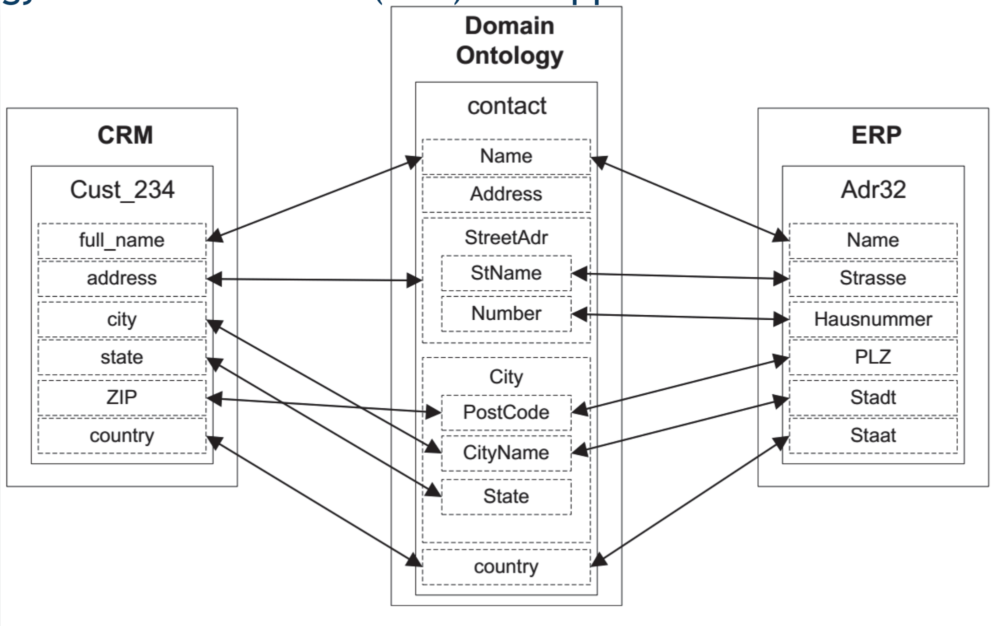
BP Management: Ontologies and Data Mapping

- A domain ontology is associated with a set of stakeholders, who need to agree on the domain ontology. In an ontology, concepts are represented by ellipses, and the relationships are represented by directed arcs.
- Example of simple domain ontology for contacts (XML standards: RDF, OWL)



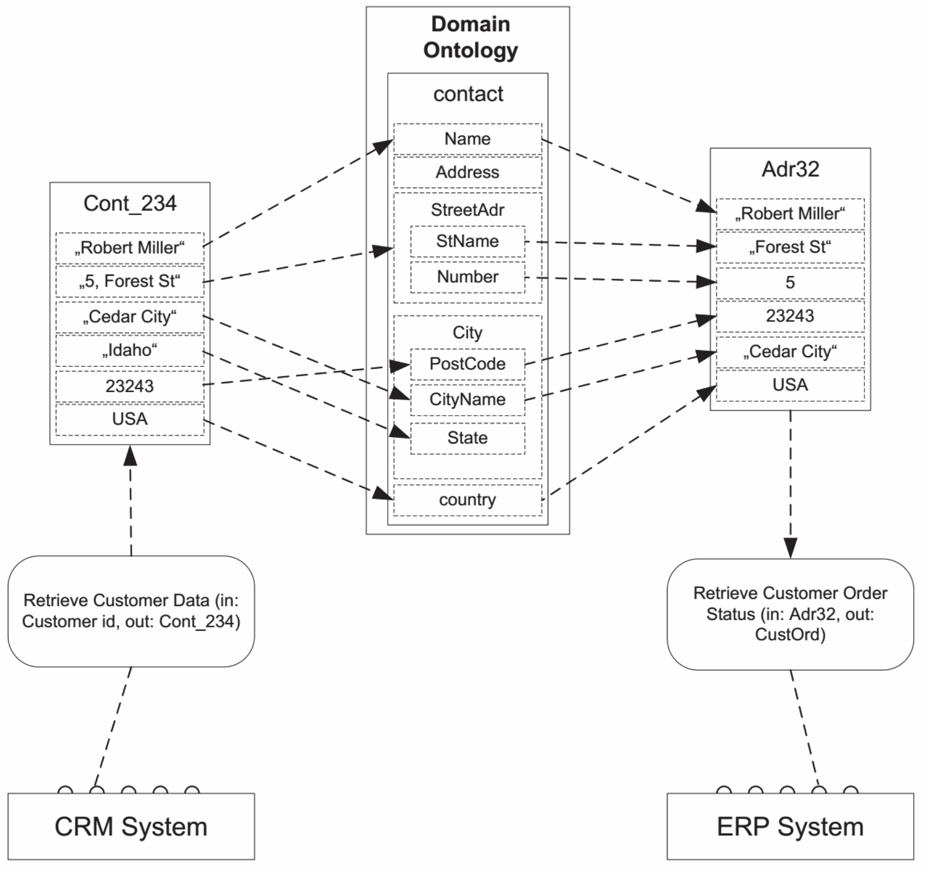
Domain ontology for contacts

- The Contact domain ontology can be used to integrate a Customer Relationship Management (CRM) and an Enterprise Resource Planning (ERP) with different data structures.
- E.g. the *full_name* field of the CRM is mapped to the *Name* concept in the domain ontology. The field *Strasse* (ERP) is mapped to *StName*.
- If each data field is mapped to the domain ontology, the mapping of the data between two system can be achieved automatically at runtime.



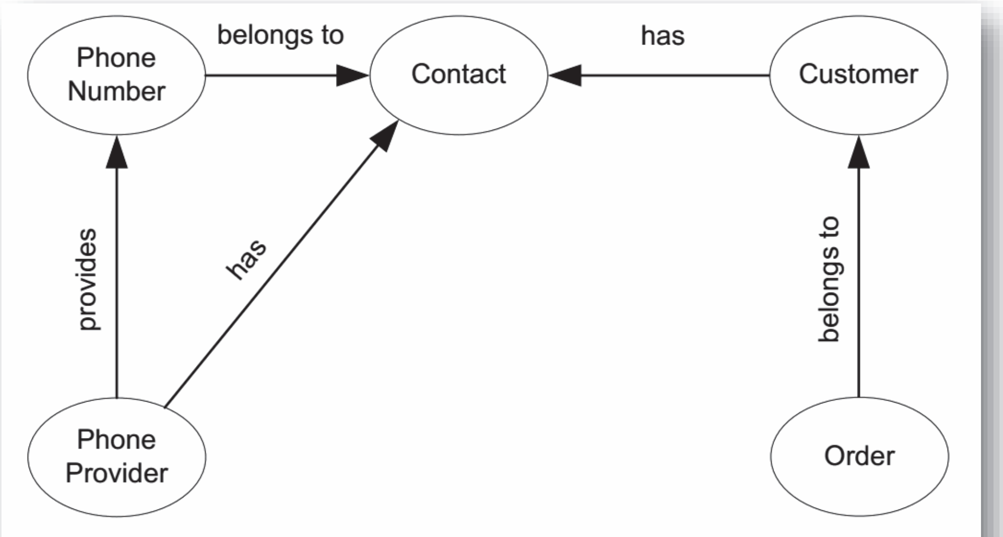
Domain ontology facilitates data mapping

- A service of the CRM returning a parameter of data type Cont_234 can be fed into a service taking a parameter of data type Adr32 if the appropriate ontology is provided.



Domain ontology used for automatic data mapping of CRM customer data to ERP customer data

- Let us consider a call center domain, where phone calls by customers come in and call center agents serve these calls using an ERP and a CRM software systems.
- In a call center environment, a customer calls to request certain information. Using the phone number of the incoming call, the CRM gets hold of the customer address, which is, in turn, fed to the ERP to provide information on the customer.
- Another service may take a phone number as input and provide the address of the phone provider as output.



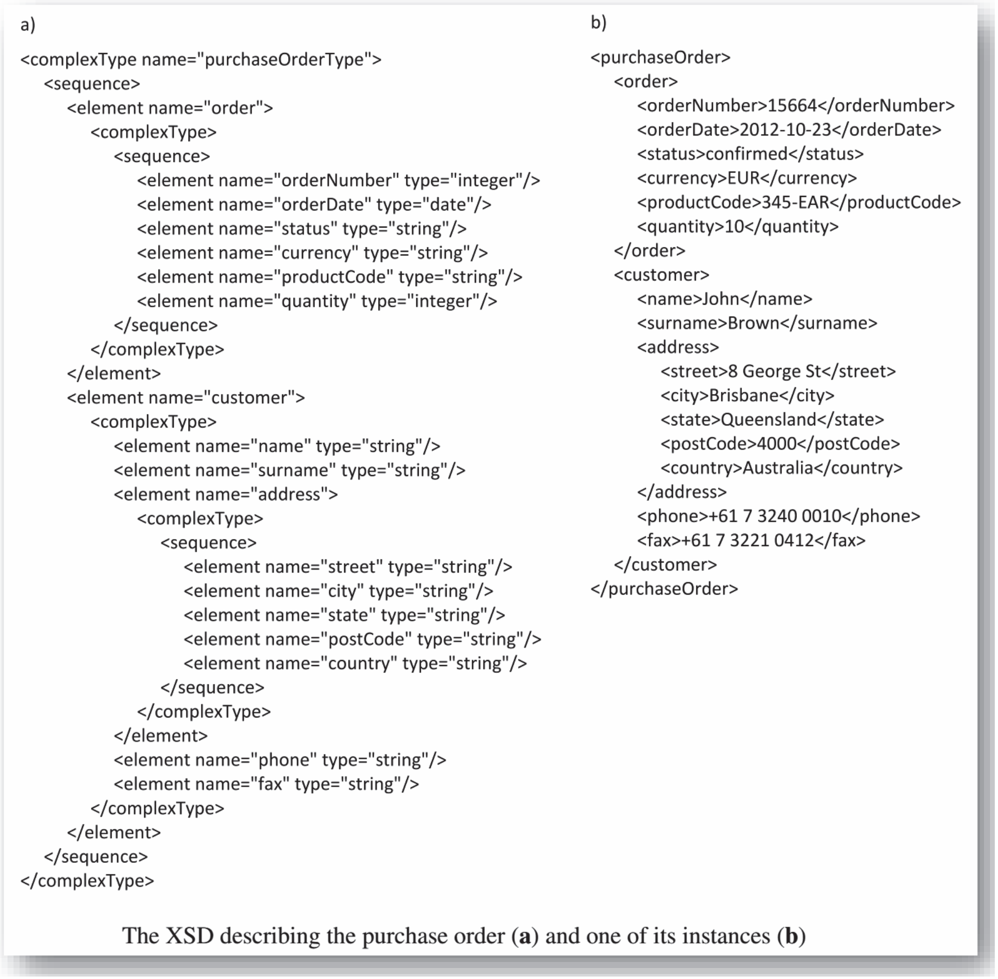
Domain ontology of call centre example

Summary of aspects to make a process executable

- **Process variables** are managed by the BPMS engine to allow data exchange between process elements. E.g. the purchase order in the order fulfillment process, represents a process variable.
- The **lifetime of a process variable** is confined to the life of the process instance in which the variable is created, and is only visible to the process level in which it is defined and to all its sub-processes. This means that a variable defined in a sub-process is not visible in the parent process.
- We need to assign a data type to each process variable to allow BPMS to interpret and manipulate these variables. In BPMN, the type of each process variable can be specified as an XSD (XML Schema definition) type.
- The type of a variable can be simple or complex. Simple types are strings, integers, doubles (numbers containing decimals), Booleans, dates, times, etc. E.g. The object Stock availability can be represented as a process variable of type integer (representing the number of available units of a product).

Making processes executable on BPMS: Process variable

- Complex types are hierarchical compositions of other types. A complex type can be used for example to represent a business document, such as a purchase order or an invoice.



a)

```
<complexType name="purchaseOrderType">
  <sequence>
    <element name="order">
      <complexType>
        <sequence>
          <element name="orderNumber" type="integer"/>
          <element name="orderDate" type="date"/>
          <element name="status" type="string"/>
          <element name="currency" type="string"/>
          <element name="productCode" type="string"/>
          <element name="quantity" type="integer"/>
        </sequence>
      </complexType>
    </element>
    <element name="customer">
      <complexType>
        <element name="name" type="string"/>
        <element name="surname" type="string"/>
        <element name="address">
          <complexType>
            <sequence>
              <element name="street" type="string"/>
              <element name="city" type="string"/>
              <element name="state" type="string"/>
              <element name="postCode" type="string"/>
              <element name="country" type="string"/>
            </sequence>
          </complexType>
        </element>
        <element name="phone" type="string"/>
        <element name="fax" type="string"/>
      </complexType>
    </element>
  </sequence>
</complexType>
```

b)

```
<purchaseOrder>
  <order>
    <orderNumber>15664</orderNumber>
    <orderDate>2012-10-23</orderDate>
    <status>confirmed</status>
    <currency>EUR</currency>
    <productCode>345-EAR</productCode>
    <quantity>10</quantity>
  </order>
  <customer>
    <name>John</name>
    <surname>Brown</surname>
    <address>
      <street>8 George St</street>
      <city>Brisbane</city>
      <state>Queensland</state>
      <postCode>4000</postCode>
      <country>Australia</country>
    </address>
    <phone>+61 7 3240 0010</phone>
    <fax>+61 7 3221 0412</fax>
  </customer>
</purchaseOrder>
```

The XSD describing the purchase order (**a**) and one of its instances (**b**)

Making processes executable on BPMS: Task variable

- Internal variables of each task, called data inputs and data outputs in BPMN, need to refer to an XSD type defining their structure. Differently from process variables, they are only visible within the task (or sub-process) in which they are defined.
- E.g. a data input for task “Check stock availability” in order to store the content of the purchase order.
- The association between data objects and task data inputs/outputs is defined via a data mapping. In most cases, the BPMS will automatically create all the tedious data mappings between data objects and tasks.
- BPMN relies on XPATH as the default language for expressing data assignments, other languages can be used like Java Universal Expression Language (UEL) or **Groovy**.
- E.g. *Activiti BPM* supports UEL, *Bonita Open Solution* and *Camunda Fox* support **Groovy** while BizAgi’s BPM Suite supports its own expression language.

Making processes executable on BPMS: Service Task

- A service task specifies how to communicate with the external application that will execute the task. It is required is that the external application provides a service interface that the service task can use.
- A service interface contains one or more service operations, each describing a particular way of interacting with a given service. For example, a service for retrieving inventory information provides two operations: one to check the current stock levels and one to check the stock forecast for a given product.
- An operation can either be in-out or in-only, thus expecting a request/response message or request only. Each message of a service operation needs to reference a message in the BPMN model, so that it can be assigned an XSD data type.
- For each interface, a concrete implementation is defined: which communication protocols are used b the service and where the service is located in the network. By default, BPMN uses Web service technology to implement service interfaces, and relies on SOAP/REST and WSDL to specify this information.

Making …: Send/Receive Tasks, Message/Signal Events

- A send task is a special case of the service task: it sends a message to an external service using its data input, but there is no response. A receive task waits for an incoming message and uses its data output to store the message content.
- A receive task can be used to receive the response of an asynchronous service which has previously been invoked with a send task. The asynchronous service is provided by the consumer.
- Accordingly, in the send task the producer process acts as the service requester sending a request message to the consumer. In the receive task the roles get swapped: the producer acts as the service provider to receive the response message from the consumer.
- This pattern is used for long-running interactions, where the response may arrive after a while. The drawback of using a synchronous service task in place of a send-receive is that this task would block the process to wait for the response message.
- Message and signal events work exactly like send and receive tasks

Making …: Script task and User task

- For **script task**, provide the snippet of code that will be executed by the BPMS, in a programming language such as JavaScript or Groovy.
- The task data inputs store the parameters for invoking the script while the data outputs store the results of script execution.
- For **user task,** specify the rules for assigning work items of this task to process participants at runtime, the technology to communicate with participants and the details of the user interface to use.
- Also, define data inputs to pass information to the participant, and data outputs to receive the results. Process participants are members of a resource class, sharing certain characteristics, holding the same role or belonging to the same department or unit.
- Specify the implementation technology used to offer the work item to the selected participant(s): (i) how to reach the participant (e.g. via email or worklist notification), (ii) how to render the content of the task data inputs on screen (e.g. via web forms organized through screenflows), (iii) the strategy to assign the work item to a single participant out of the assignment expression (e.g. assign it to the order clerk with the shortest queue or randomly).

Making …: Task, Event and Sequence Flow Expressions

- To write expressions for the attributes of tasks and events, and for the sequence flows including conditions. E.g. in a loop task we need to write a boolean expression implementing the condition “until response approved”. For timer events, e.g. “Friday afternoon”, it can be provided a temporal expression in the form of a precise date or time, a relative duration, or a repeating interval.
- These expressions can be linked to data elements and instance properties so as to be resolved dynamically at execution. For example, we can set an order confirmation timeout based on the number of line items in an order.
- To write a Boolean expression to capture the condition attached to each sequence flow following an (X)OR-split. For example, the condition “product in stock” after the first XOR-split in the order fulfillment example can be implemented as an XPATH expression.
- There is no need to assign an expression to a default sequence flow, since this arc will be taken by the BPMS engine if the expressions assigned to all other arcs emanating of the same (X)OR-split are false.

Making processes executable on BPMS: system binding

- The most BPMS-specific properties to configure in order to make a process model executable are those of user tasks and those to link the executable process with the enterprise systems (system binding).
- BPMSs offer a range of predefined service task extensions, called service *adapters or connectors*: performing a database lookup, sending an email notification, posting a message to Twitter or setting an event in Google Calendar, reading or writing a file and adding a customer in a CRM system.
- Each adapter comes with a list of parameters that we need to configure. BPMSs provide wizards with capabilities to auto-discover some of the parameter values. For instance, to use a database lookup we need to provide the type of the database server (e.g. MySQL, Oracle DB) and the server's URL, the schema to be accessed, the SQL query to run and the credentials of the user authorized to run the query.
- E.g. instead of implementing “Check stock availability” as a service task, generic database lookup adapter can be used if available. The task “Notify unavailability to customer” and “Request shipping address” can be implemented via email adapters, without dedicated email services.