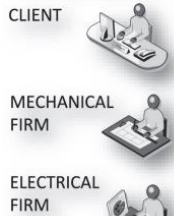
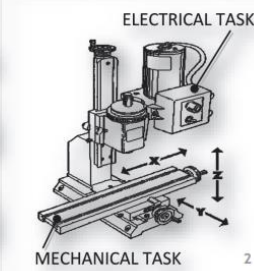


Business Process Mining Management Systems

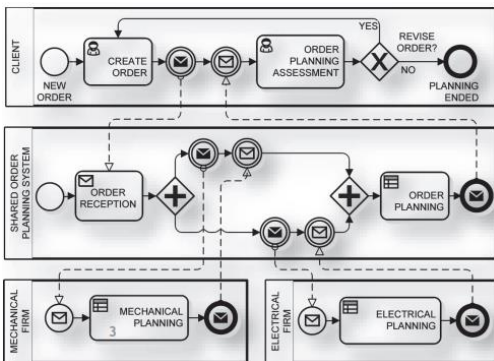
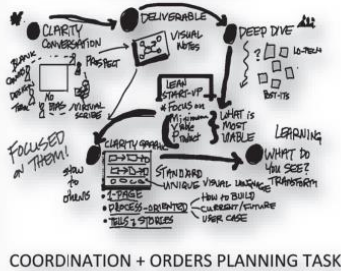
PARTICIPANTS



DESIGN AND DEVELOPMENT



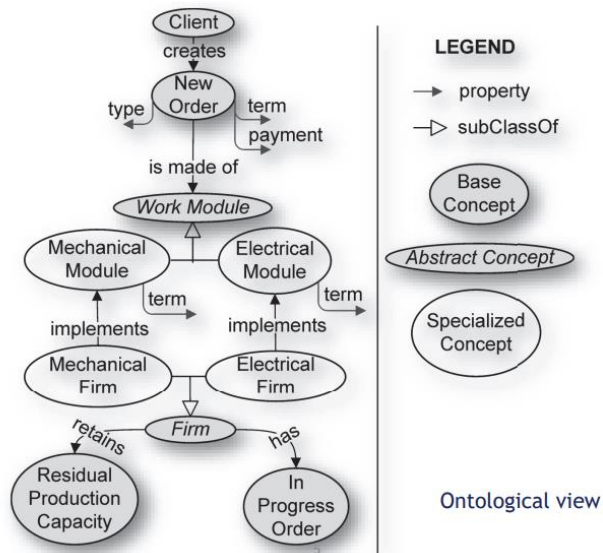
MANAGEMENT



Il task con una "tabella" in alto a sinistra indica una BRMS (Business Rule Management System) che viene chiamata dal process engine.

Si suppone che ciascun pool di una firm sia eseguito in un server privato per la firm, mentre il Planning System e il Client sono eseguiti su un server condiviso. In questo modo le BRMS di ciascuna firm sono completamente nascoste alla community.

Un ordine può essere di tipo *standard* o *innovative* (rispetto agli ordini passati), può essere performed in un periodo breve o lungo in base a diversi fattori.



Per semplicità in questo scenario l'ontologia è globalmente condivisa tra i partecipanti e le business rules sono differenti per ogni partecipante. Una ontologia può anche essere modularizzata, con l'obiettivo di evitare la condivisione di concetti privati.

- **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.

- **Formal IF-THEN rules**

TASK: MECHANICAL PLANNING	
RULE 1: If <i>newOrder.type</i> Is standard And <i>inProgressOrder</i> Is true Then <i>mechanicalModule.term</i> Is long	RULE 3: If <i>newOrder.type</i> Is standard And <i>inProgressOrder</i> Is false Then <i>mechanicalModule.term</i> Is short
RULE 2: If <i>newOrder.type</i> Is innovative Then <i>mechanicalModule.term</i> Is long	
TASK: ELECTRICAL PLANNING	
RULE 1: If <i>residualProductionCapacity</i> Is false Then <i>electricalModule.term</i> Is long	RULE 3: If <i>residualProductionCapacity</i> Is true And <i>newOrder.payment</i> Is fast Then <i>electricalModule.term</i> Is short
RULE 2: If <i>residualProductionCapacity</i> Is true And <i>newOrder.payment</i> Is slow And <i>newOrder.type</i> Is innovative Then <i>electricalModule.term</i> Is long	RULE 4: If <i>residualProductionCapacity</i> Is true And <i>newOrder.payment</i> Is slow And <i>newOrder.type</i> Is standard Then <i>electricalModule.term</i> Is short
TASK: ORDER PLANNING	
RULE 1: If <i>mechanicalModule.term</i> Is long Then <i>newOrder.term</i> Is long	RULE 3: If <i>mechanicalModule.term</i> Is short And <i>electricalModule.term</i> Is short Then <i>newOrder.term</i> Is short
RULE 2: If <i>electricalModule.term</i> Is long Then <i>newOrder.term</i> Is long	7

Collaborative Analytics

- Le BRs sono solitamente progettate in accordo ad obiettivi che sono misurabili tramite i KPIs (Key Performance Indicators), per ogni azienda e per la comunità stessa. Per questo motivo l'usabilità the data flow collegati al workflow è un requisito fondamentale.
- Le collaborative analytics hanno anche lo scopo di aggregare i dati in modo da non renderli pubblici esplicitamente.
- In generale lo scopo di questa tecnica è comparare la performance con altre aziende.

"To share or not to share" dilemma = essenzialmente un buyer non vuole condividere informazioni relativi ai buoni provider, mentre non ha problemi a condividere informazioni sui cattivi buyer. Questo per mantenere un vantaggio competitivo sugli altri buyers. In ogni caso ogni buyer conosce solo un sottoinsieme di tutti i provider sul mercato, non è quindi possibile che un buyer possa fare un ranking assoluto di questi.

Come estensione del BPMN mostrato in precedenza consideriamo la capacità del Planning System di selezionare partner alternativi nei casi in cui la mechanical firm o la electrical firm non soddisfano i requisiti del cliente.

Per farlo un Order Planning Assessment task dovrebbe essere gestito anche dal Planning System. Poi sarebbe necessario aggiungere una nuova attività chiamata Select Alternative Partner.

The Collaborative AnalyticsSystem è il main pool localizzato su un server condiviso e coordina i pool dei buyers registrati. Il pool di ciascun buyer è localizzato su un server privato.

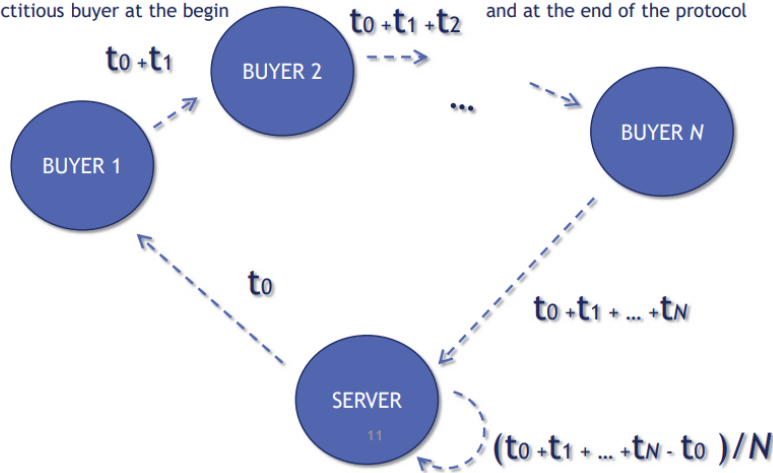
Il principale obiettivo del data flow è creare dati pubblici collettivi aggregando i dati privati dei buyers.

Il tempo medio di consegna dei venditori ad un buyer è un esempio di dato privato.

Mentre il tempo medio di consegna dei venditori a tutti i buyers è un esempio di dato collettivo.

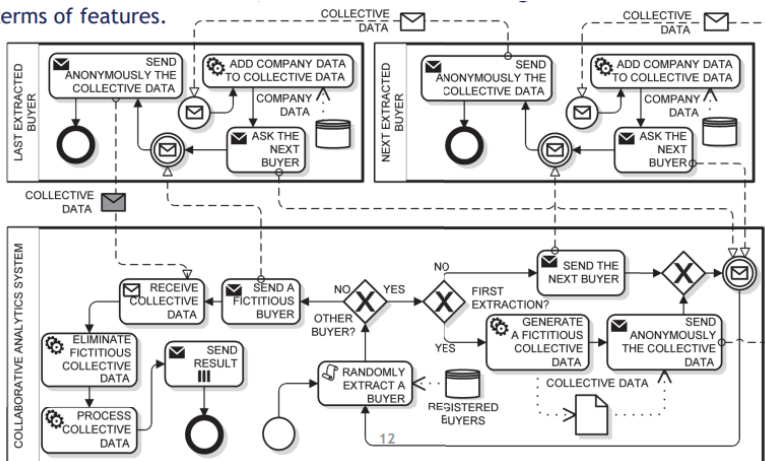
The problem: how to calculate the average without sending each term t_k to the server?

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



In generale il processo di aggregazione protegge i dati dei buyers dal diventare pubblici: il System estrae casualmente un buyer e genera un dato fittizio collettivo che è una creazione artificiale che sembra un dato reale e non può essere distinto in termini di features.

terms of features.



- Il dato collettivo è mandato in modo anonimo al buyer estratto, il quale aggiunge il proprio dato privato e chiede al sistema il next buyer.
- Il sistema estrae a caso il next buyer.
- Il buyer manda al next buyer in modo anonimo il proprio dato. In questo modo si costruisce il dato collettivo in maniera incrementale.
- Ciascun buyer non conosce la propria posizione nella sequenza perché il primo riceve dati fittizi ed il sender è sempre anonimo. L'ultimo riceve un utente fittizio che in realtà è il sistema.
- Il sistema alla fine sottrae dal dato collettivo il dato fittizio iniziale. Quindi estrae le features che servono e le manda a tutti i buyer.
- Ogni buyer può quindi comparare le sue performances rispetto a quelle collettive e mantenere solo i partner migliori.

Bonitaaaaaaaa (sta parte l'ha fatta il King di Bonita)

Mi presento, sono il king di Bonita anche detto G****o C*****i. Se mi conosci bene, se non mi conosci, a breve vedrai le mie skillz.

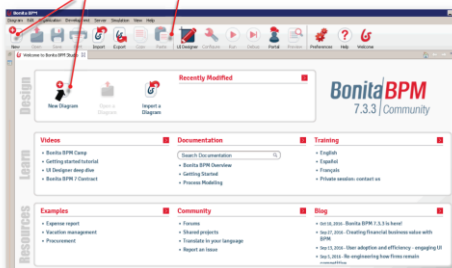
Cominciamo col dire che "Bonita BPM 7" è una applicazione per costruire applicazioni di business process-based, nel senso che si adatta ai cambiamenti del business in real time. È composta da 2 parti:

- Bonita BPM studio → development environment
- Bonita BPM platform → runtime environment

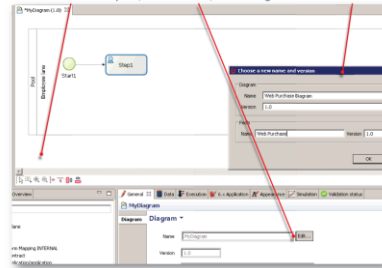
Adotta un approccio "model driven", nel senso che si fa guidare dai modelli, e in che linguaggio saranno sti modelli? Ovviamente BPMN, ma anche l'Unified Modeling Language (UML) sarebbe possibile. Questi modelli so poi processati per generare software (che hit). Per quanto visto a lezione, dovrete avere bonita in "C:\pmi".

Proseguendo, il pacco di slide ci mostra come creare un BPMN model direttamente in Bonita, dove creiamo per prima cosa il pool, poi andiamo a riempirlo. Notare che tra le domande che Marco GCU Cirilino può fare all'esame troviamo anche "Come importare un BPMN in bonita". Bene, la risposta a questo mistico quesito è che basta usare gli occhi e aprire Bonita stessa. Riporto di seguito gli step per andare a creare un BPMN a mano, non c'è molto da dire in quanto sono auto-esplicativi (e se non lo sono questa parte di roba si trova nel file 13 sul sito di Cirilino, anche detto BPMSA4x2.pdf).

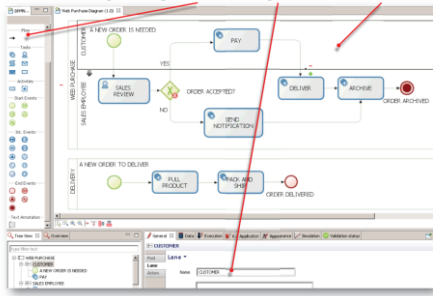
1. Select New from the Cool bar to create a new diagram



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

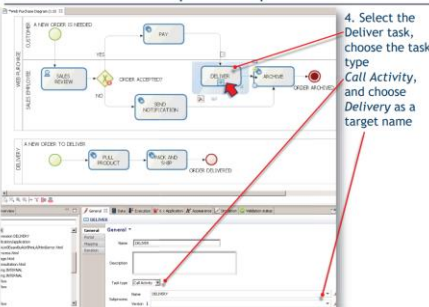


3. Create the diagram using the toolkit, configure the selected element



Bonita BPM: Connect a sub-process to a pool

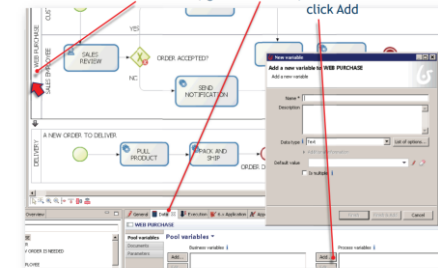
21 of 85



Bonita BPM: Add Process variables

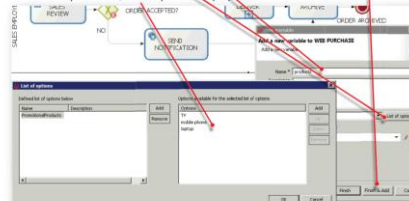
22 of 85

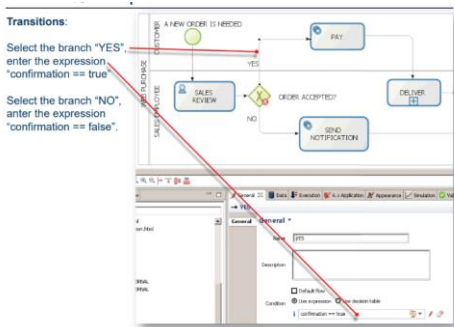
5. Select the Web Purchase Pool, go to Data Pane, on Process variable click Add



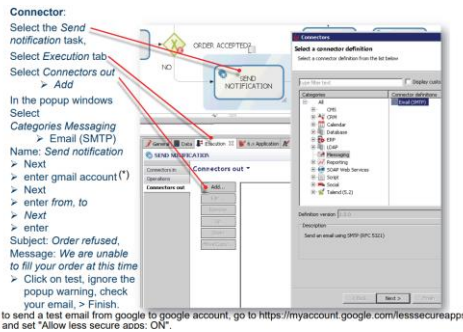
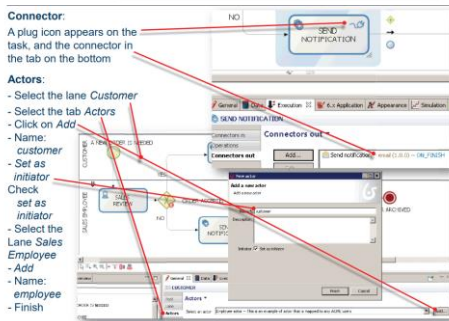
Process variables: can be used in a process and until the process instance is completed.

6. Enter customerName, leave Data type Text, and press Finish&Add enter 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's laptop.





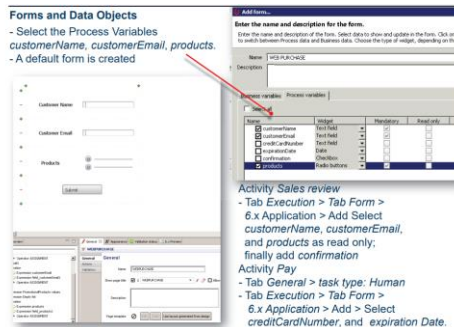
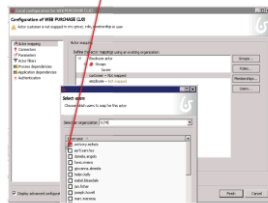
Fatta tutta stammerda aggiungiamo ora i rami ai gateway (a sinistra) e maneggiamo i "connector" (sotto); questi servono per automatizzare azioni essenzialmente, tipo l'invio di una mail automatica sull'acettazione di un ordine. Aggiungiamo poi il mapping degli attori, i forms e i data objects. Queste son tutte cose che saranno visibili accedendo alle pagine automaticamente generate da Bonita tramite interfaccia web.



Commentato [GC1]: Interessante procedura assolutamente non exploitabile

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



Bonita BPM Use Case

Andiamo quindi a chillarcela sull'interfaccia web. Per prima cosa c'è da cliccare RUN nella "cool bar" dentro Bonita. Aprire due browser differenti, e puntare a <http://localhost:8080/bonita/login.jsp>. Se gli step precedenti sono stati eseguiti correttamente non c'è beghe e si può accedere al browser con gli username/password precedentemente scelti (customer login = "anthony.nichols", employee login = "april.sanchez", per entrambi password "bpm").

Tasks **Cases** **Processes**

Search

START

Name: WEB PURCHASE (1.0)

Version: 1.0

No description

1 of 1

My cases

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

- fill the Web Purchase form and click on submit

Customer Name *

Customer Email *

Products *

TV

mobile phone

laptop

Submit

- Select the task and press take

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

SALES REVIEW

Customer Name

Customer Email

Confirmation

Products

TV

mobile phone

laptop

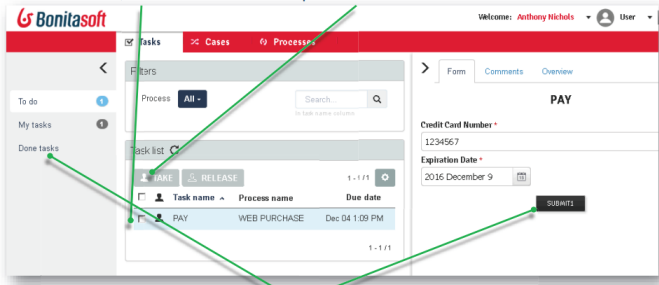
Submit

- 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

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



Fatto ciò, si dovrebbe vedere nei done tasks la cronologia dei tasks.

- 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

Per provare ad utilizzare i connettori al DB, seguire i successivi steps :

1. Per prima cosa, creare il seguente BPMN dentro BONITHA (seguendo gli steps dell'esempio precedente su come fare)

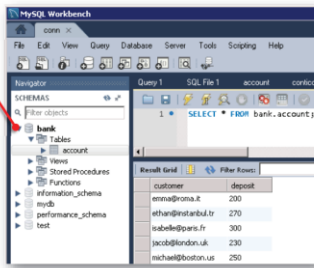


2. New Diagram > complete the flow with the toolkit leaving the default task types.
3. Select *Step1* > General Tab > Task type: Service.
4. Select *Step2* > General Tab > Task type: Human.
5. Click on Save in the cool bar.
6. **Create the process variables:**
7. Select *Pool* > Data Tab > Process Variables: 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

- 2.
3. Prima di fare il prossimo step, consiglio di scaricare il mysql (autocontenuto) fornito dal sito di Cirlino; motivo è che i prossimi steps poi li fa con quello e a qualcuno del corso non funzionava usando la versione di Mysql che di solito usiamo.

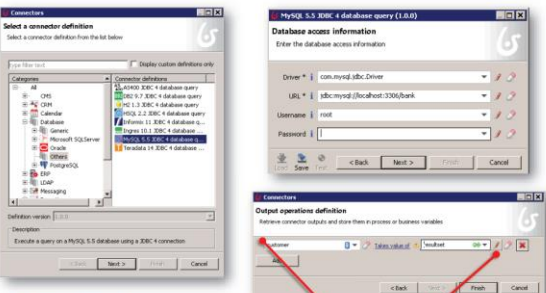
14. **Create the MySQL Database:**
15. 1st method: import the file *bank-dump.sql* into a MySQL server.
16. 2nd method: download the file www.iet.unipi.it/m.cimino/wdis/res/dbms.zip and extract it on C:\wdis. Finally, click on C:\wdis\mysqlStart
17. **Access the Database with MySQL client:**
18. Click 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.



- 4.

5. Possiamo ora a collegare Bonitha al DB tramite connector



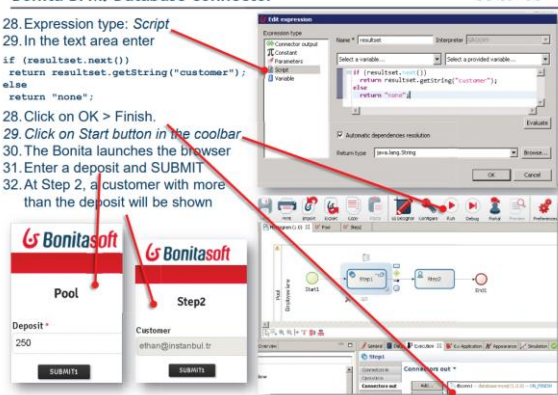
23. Enter the query
 24. `SELECT * FROM account WHERE deposit > ${deposit};`
 (for autocompletion of variables press CTRL + SPACE)
 23. Select **Next > Scripting Mode > Next > Select target: customer**
 24. Click on the pencil icon to open the Groovy editor.

Bonita BPM: Database connector 35 of 85

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 toolbar
 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



- 6.

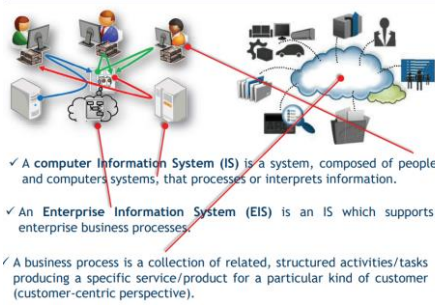
Bonita BPM: Database and Web Service connectors

Questa roba richiede SOAP UI che non ha mai fatto a lezione e non è nel registro; visti i poteri conferitomi da Thalia Cruz io skippo.

Modeling Enterprise systems (dove “enterprise”=“impresa”)

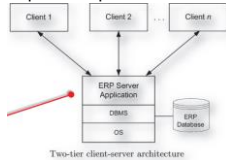
Vediamo per prima cosa generalmente il landscape:

- Questi “enterprise information systems” sono solitamente formati da 3 livelli, i quali sono
 - infrastruttura IT
 - Solitamente globalmente distribuita
 - Applications
 - Coordinano e servono i business processes
 - business processes
 - Dovrebbero essere definiti in maniera chiara come un set di attività (=tasks) coordinate (workflow)
- Un task può essere definito come un set di steps dettagliati di un use case. Lo use case rappresenta il goal di un utente, il quale supporta uno o più tasks del business process.



Uscendo da questa parentesi inutile e passando alla vera e propria **evoluzione dell'Enterprise system architecture**, vediamo i passi:

1. In prima era avevamo applicazioni monolitiche sviluppate in assembler, senza GUI e con sola CLI, una vera merda. Queste gestiscono il tutto quindi lo pigliano pure nel culo perché son più appesantite che il Longo dopo una carbonara da 180g di Primo.
2. Avvento dei DBMS e GUI, abbattimento costi dei computer. Ora si scinde in due le applicazioni; è tipico per HR (human resources) averne una, per PO (Purchase Order management) un'altra e infine una per Production Planning. Ognuna con proprio DBMS. Qui c'è tutte le beghe del Ducange, quindi avendo dati storiati in più sistemi collegati da dipendenze sugli ID, c'è le beghe se devo propagare updates e tutta sta roba
3. Sviluppo dei primi **ERP** ("Enterprise Resource Planning systems"); con questi si cerca di sviluppare applicazioni di impresa disparate sopra lo stesso DBMS centralizzato

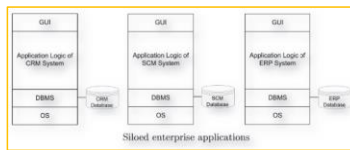


Two-tier client-server architecture

4. Intorno agli anni 2000, ci so nuovi bisogni sul mercato quindi si sviluppano nuovi softwares; si parla di
 - a. Supply Chain Management (**SCM**) systems
 - b. Customer Relationship Management (**CRM**) systems

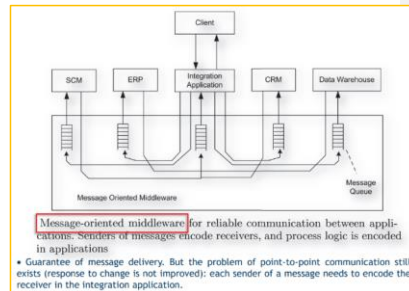
Lo scopo è quello di supportare operazioni sul supply chain, il planning e tutte ste belle robe.

5. Nonostante questo c'è ancora beghe perché si parla di "siloed applications" che non so distribuite. Ci vuole una roba tipo quella del Puliafio e Vallahi
6. Si arriva quindi ad aggiungere **middleware systems**; **EAI** (**Enterprise Application Integration systems**). Questi servono essenzialmente per integrare sistemi diversi, per esempio:
 - a. Adattano apps diverse (e.g. in una app ho attributo "username" e nell'altra "name"). Si fa aggiungendo "adapters" tra app diverse
 - b. Si basano su scambio di messaggi (quelle di prima no)
 - c. Introducono message broker (publish/subscribe mechanism). La su bega è che c'ha molta logica



Siloed enterprise applications

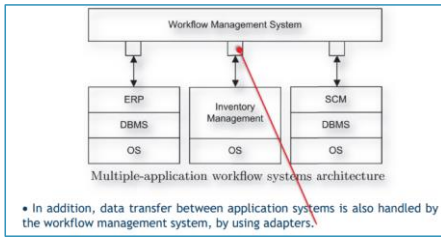
7. Da **Application Integration a Process Orientation**. Qui si tocca il top: mentre la app integration fa schifo perché si hanno sequenze parzialmente ordinate di step per realizzare un processo, con la PO ci si appoggia al Taylorismo. Il signor Taylor ci dice che è bono avere attività piccole e granulari fatte da personale specializzato, ciò porta a vantaggi quale il parallelismo. Il tutto è vero fino agli anni 80, in quanto nelle organizzazioni di business moderne le cose son più complicate viste e.g. interdipendenze tra steps, lavoro trasferito tra compagnie diverse e così via.



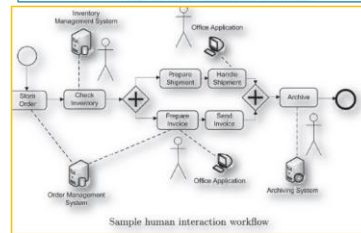
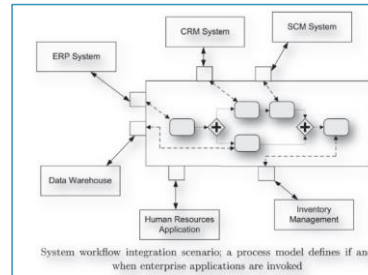
Message-oriented middleware for reliable communication between applications: Senders of messages encode receivers, and process logic is encoded in applications

• 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.

La conquista dell'evoluzione del workflow management è come punto ultimo la rappresentazione esplicita di strutture di processi nei "process models"; con questi si usano bene i *model-driven approaches*, che so il top in quanto flessibili e adattabili in base ai business processes. Oggi, in molte imprese abbiamo un **WfMs** (**workflow management system**), che se ne occupa.



Mostriamo a sinistra uno schemino col WfMs e i suoi adapter per i vari sistemi.



Ci sono due tipi di WfMs :

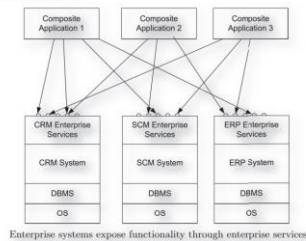
- System Workflows.** Consistono di attività implementate interamente in sistemi puramente software, senza coinvolgimento di utenti umani
- Human interaction workflows.** Come prima ma con umani coinvolti

Concludiamo sui WfMs, dicendo che non servono sempre; in casi dove non sono previsti cambiati alle strutture dei processi, un loro uso può essere adeguato, in caso contrario non troppo. Stessa cosa per business di tipo "online", dove l'interazione utente/customer viene automatizzata. Anche in questo caso un WfMs non è necessario.

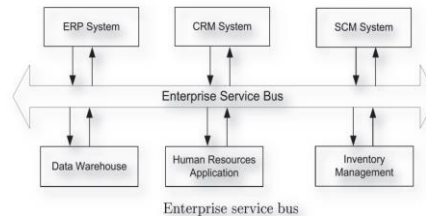
Service oriented architecture (SOA) e Web Services (WS)

Rimanendo sulla Evolution delle Enterprise system architectures, vediamo ora il trend più moderno ed attuale, che è quello delle **SOA (service oriented architecture)**, implementate da **WS (web services)**. Un WS è un software che fornisce operazioni in formato indipendente dalla piattaforma su cui si trova e con cui comunica, spesso usando XML. Spesso usa inoltre http (ideato inizialmente per human-machine communication) per realizzare machine-machine communication e per invocare operazioni su altre macchine. Per realizzare tutto questo le interfacce sono standardizzate e basate su XML. Inoltre, si possono realizzare applicazioni composite, aggiungendo servizi sul top di altri, oppure rimuovendone alcuni se non ci sono dipendenze col resto.

- 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.

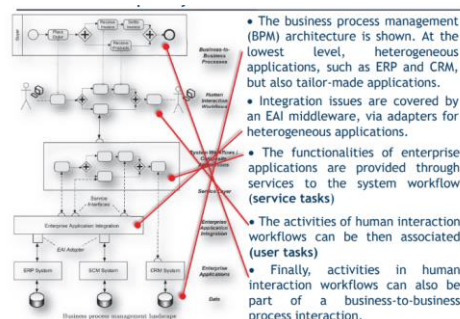
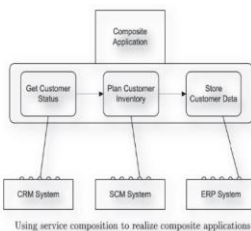


- The term Enterprise Service Bus (ESB) means that each enterprise application is attached to the bus, which acts as an application independent integration middleware.



Arrivando al dunque, il punto qui è che le attività/task dei processi sono implementati invocando servizi dell'impresa; quindi, si può realizzarci essenzialmente sopra dei BPM. Che hit!

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



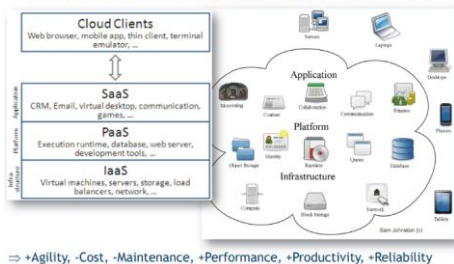
- The business process management (BPM) architecture is shown. At the lowest level, heterogeneous applications, such as ERP and CRM, but also tailor-made applications.
- Integration issues are covered by an EAI middleware, via adapters for heterogeneous applications.
- The functionalities of enterprise 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.

Tornando ai WS, sono sistemi di software il cui design supporta machine-machine interactions sopra una network. Questo è il top sia per far comunicare business diversi, sia per farli comunicare coi clienti stessi. Per comunicare coi Web Services, usiamo protocolli che abbiamo già visto a Cloud/Pulitafo/Vallelata, e che non hanno nessun senso di essere riproposti anche in questo esame. Vediamoli!

- **SOAP (simple object access protocol)**, lavora sopra http per scambiare XML messages tra chi richiede e chi fornisce il servizio
- **REST (representational state transfer)**, uguale al SOAP ma lightweight. Si basa su messaggi di richiesta/risposta in HTTP, encodati però in JSON/XML
- **WSDL (web services descriptor language)**, specifica la interfaccia del WS, quindi come comunicarci, gli endpoints,...
- **UDDI (universal description, discovery and integration)**, fornisce informazioni per la ricerca e accesso dei WS tramite un registro

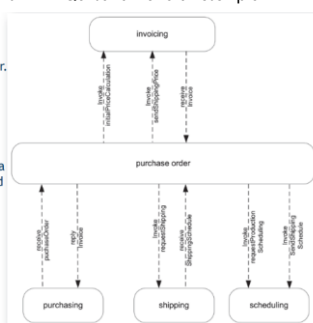
Il meccanismo poi è il solito visto a Cloud, con il provider che deve preparare un WSDL file per il servizio che vuole attivare, il richiedente poi riceve questo WSDL e sa quindi come sono fatte le interfacce del servizio e i loro indirizzi. Poi le invoca con SOAP/REST. Qui non parla di un broker né di come il richiedente ottenga effettivamente il WSDL, a cloud c'era un broker.

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



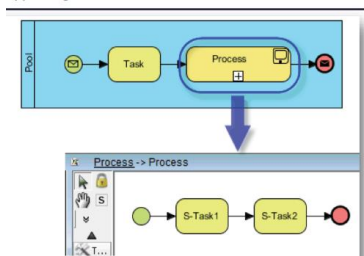
Il top qui come sappiamo è fare composizione di web services, arrivando proprio alle service oriented architectures. Qui in questa materia strana si parla di **"WS composition"**: questa descrive come i servizi sono legati tra loro. Si parla anche di **WS-BPEL**: è il *business process execution language* per i WS, legato allo standard XML. Qui sotto metto un esempio

- 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.

Questi BPEL possono essere generati dal BPMN se si introducono limitazioni; il BPMN è un grafo essenzialmente, mentre il BPEL essenzialmente è una struttura a blocchi. Inoltre il BPEL offre loops, if-then-else e i tipi di XML quindi è "Turing completo" (???). Le trasformazioni da BPMN a BPEL lo sapete con cosa si fanno? Esatto, con **Visual Paradigm** (era facile come risposta).



Questo sulla sinistra è un esempio di trasformazione di Visual paradigm; il processo viene scisso nelle componenti, che vengono nel BPEL (la foto in XML qui sotto) aggiunte al flusso regolare di esecuzione dopo "Task".

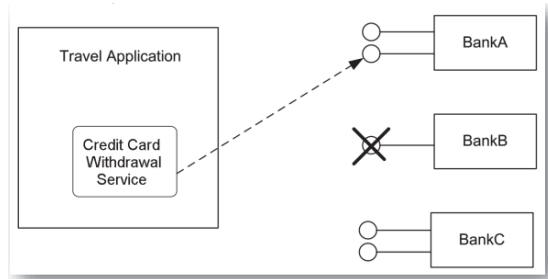
```
<?xml version="1.0" encoding="UTF-8"?>
<process name="BusinessProcessDiagram1" targetNamespace="http://BusinessProcessDiagram1" xmlns="...">
  <partnerLink myRole="provider" name="Pool" partnerLinkType="Pool:PartnerLinkType"/>
  <partnerLinks>
  </partnerLinks>
  <variables>
    <variable messageType="Pool:Message" name="Variable"/>
  </variables>
  <sequence>
    <receive createInstance="yes" operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType">
      <empty name="Task"/>
      <empty name="S-Task1"/>
      <empty name="S-Task2"/>
    </receive>
    <reply operation="perform" partnerLink="Pool" portType="Pool:ProviderPortType" variable="Variable"/>
  </sequence>
</process>
```

Ulteriori trasformazioni che Visual Paradigm esegue: (ci sono esempi ma li salto in quanto inutili)

- Lo XOR gateway dei BPMN (cioè quello con la "X" classico e le condizioni sui rami) vien trasformato in BPEL switch (se vuoi è a pagina 45 ma non ha senso recuperarlo, c'è solo un XML dimmerda)
- Lo XOR event-driven dei BPMN diventa un *BPEL pick*, che fornisce due rami, ognuno con una condizione. Il pick è una struttura per cui il ramo in uscita che verifica per primo la condizione, è il primo ad essere eseguito (esattamente come succede per il gateway event driven)
- Nei BPMN, riportare un sequence flow dentro un gateway (quindi creando un loop), viene tradotto in BPEL in un *while condition = false*, dove semplicemente si ripete l'operazione finché non si passa a true.

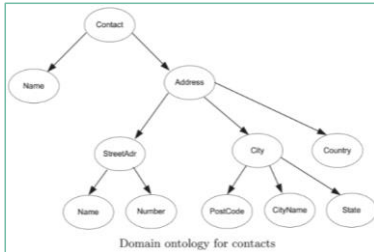
Esempio di sviluppo di applicazione coi WS – travel application

Qui dobbiamo consentire ai clienti di selezionare viaggi, far prenotazioni, confermarle e pagare con carta. Riportiamo la struttura del pagamento; essenzialmente la applicazione qui comunica a Runtime con un *service broker*, che le comunica il *service registry*, poi ne sceglie uno che meglio rispetta le sue preferenze.

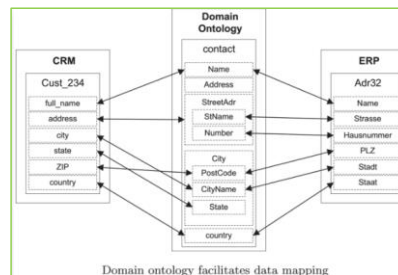


Ci son problemi poi relativi alla standardizzazione dei dati, in quanto per esempio i prezzi possono essere in sterline o in euro eccetera. I dati per risolvere questo problema dovrebbero quindi essere annotati con un "**Semantic Web standard**", di modo che la loro comparazione possa essere automatica.

Vediamo le **ontologies** e i **data mapping**: una domain ontology contiene concetti rappresentati come ellissi, e le relazioni tra questi rappresentate con archi diretti. Vediamone un esempio per i contatti per i customer di questa applicazione.



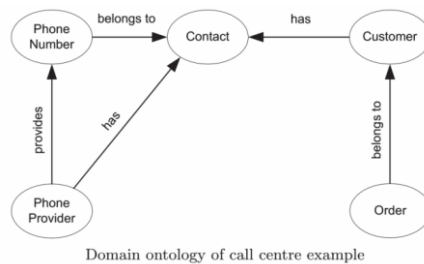
Questa può essere usata per integrare il **CRM** (Customer Relationship Management) e il **ERP** (Enterprise Resource Planning) con altre data structures. Per fare ciò vedi la foto sottostante, dove appunto si mostra il **data mapping** realizzato tramite l'uso della **ontology**: per notare come questa è usata fai attenzione per esempio a "Strasse" in ERP che diventa "StName" nella ontology, con stesso meccanismo per "address" in CRM. Questo per tutti gli attributi.



Se tutti i campi sono mappati bene alla ontology, allora il mapping tra dati di sistemi diversi può essere automaticamente ottenuto a runtime.

Ulteriore esempio – call center

I clienti chiamano il call center, e gli agenti servono le chiamate usando dei sistemi software ERP e CRM. Vediamo la ontology per capire il funzionamento del call center.



Come rendere un processo eseguibile in BPMS

Summary degli aspetti di cosa ci vuole **per rendere un processo "executable"**:

- Le "process variables" sono gestite dal BPMS engine per consentire scambio di dati tra gli elementi di process. Un esempio di process variable è "purchase order" in un "order fulfillment" process.
- La lifetime di una di queste process variables è confinata al tempo di vita del processo in cui questa è creata, ed è solo visibile a livello di questo processo e i suoi sotto-processi. NON visibile ai suoi genitori.
- Bisogna assegnare un data type ad ogni process variable, così che il BPMS possa manipolarle correttamente. Si fa con i tipi XSD (XML schema definition).
 - o può essere semplice (string, integer, boolean, date, time...)...
 - o ...oppure complesso (composizioni gerarchiche di altri tipi)
- Le variabili dentro ogni task (chiamate data input e data output in BPMN) devono riferirsi ai tipi XSD che definiscono la loro struttura. Al contrario delle variabili di processo, queste sono visibili solo dentro il task (o sotto task)
- Le associazioni tra data objects e task data input/output è definita tramite data mapping. Nella maggior parte dei casi sono fatti automaticamente da BPMS questi mappings.
- Un "service task" specifica come comunicare con la applicazione esterna che eseguirà il task; è richiesto quindi che la app esterna fornisca un'interfaccia che il task possa utilizzare
- Per ogni interfaccia, una implementazione concreta è necessaria, quindi specificando quali protocolli usare, ecc ... Di default, BPMN usa la tecnologia dei Web Services, e si basa su SOAP/REST e WSDL.
- Gestire bene i send/receive tasks (sono casi speciali di service tasks):
 - o Per i send, non c'è bisogno di attendere risposta
 - o I receive al contrario sono bloccanti finché non ricevo qualcosa

Con questi si possono quindi creare pattern di tipo produttore/consumatore. Il drawback è che il receiver ha tempi morti di attesa

- I message/signal events funzionano allo stesso modo dei precedenti
- Per i task di script, fornire lo snippet di codice che verrà eseguito
- Per i task di tipo user, fornire le user interfaces e i modi per comunicare con l'utente. Definisci anche i data inputs per passare informazioni all'utente
- Per i sequence flow, definisci dove necessario gli attributi degli stessi (per esempio, sui gateway definisci correttamente le condizioni per i rami, oppure le condizioni sui loop se ci sono)
- Come ultimo step, definisci se necessario le configurazioni per rendere eseguibile il task utente che devono interfacciarsi con l'enterprise system; questo si fa con "adapters" oppure "connectors", e li abbiamo visti prima in Bonita, quando abbiamo parlato di connessione al DB per esempio. Sono molto funzionali e spesso necessari (e.g. vedi l'invio delle e-mail, serve pefforza).