



Bonita Workflow Tutorial

BPM Manager
miguel.valdes-faura@bull.net

Who am I ?

Speaker's qualification

- Miguel Valdes-Faura is the BPM Manager at Bull R&D
 - Attached to the R&D Java Middleware team located in Grenoble (France)
 - Managing an international team of 10 persons
 - 2 located at Bull R&D Phoenix
 - 8 located at Bull R&D France

- Miguel is also member of the ObjectWeb College of Architects
 - Leading the Bonita workflow open source project
 - Leading cooperations with the Chinese Open Source community, Eclipse and Jboss

My day to day work...

R&D and project management activities

- Product roadmap and product architecture
- Team management: developments, CCB, meetings, synchronization, deadlines...
- Industrial commitment
- Technical documentation: user oriented, articles...

and also...

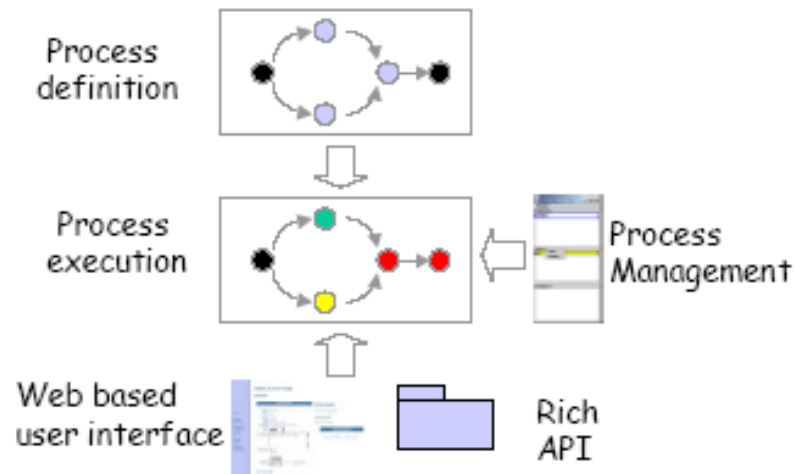
- Pre-Sales on workflow and Java Middleware
- Open source “Evangelisation” !
- Marketing activities and product valorisation
- Advanced training
- Partners relationships: eXo, Jboss, Eclipse...

Agenda, Bonita Workflow Engine

- **Bonita introduction and workflow basics**
 - Community and Objectweb relationship
 - Project history
 - Architecture
- **Bonita Highlights**
 - J2EE “Quality of Service”
 - XPDL standard support
- **Bonita basics**
 - Activities, Workflow data, transitions, sub processes, users, roles...
 - Hooks, performers, mappers and initiators
 - Iterations
 - Versioning
- **Workflow development**
 - Bonita API's
 - Bonita stand alone client
 - Use Case: The Approval Workflow sample
 - Developing hooks, mappers, performers and initiators
- **Portal integration**
 - eXo platform integration: workflow portlets
 - ECM features

Workflow basics (1/2)

- Middleware component handling processes execution and control
 - Task automation
 - User task assignment
 - Process execution and history follow up



Workflow basics (2/2)

- An environment to define, execute and monitor long duration multi-step business processes
 - Both people and software agents can participate
 - Processes can span organizational boundaries
- Key objectives
 - Process automation
 - Process control
 - Improved information sharing

- J2EE based Workflow solution
- Supports collaborative & business apps



<http://bonita.objectweb.org>

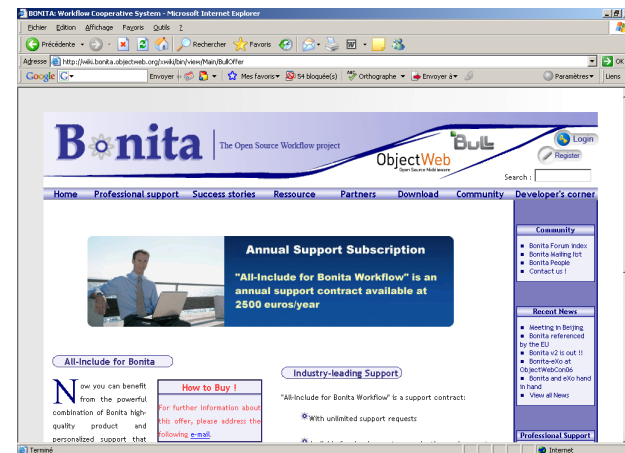
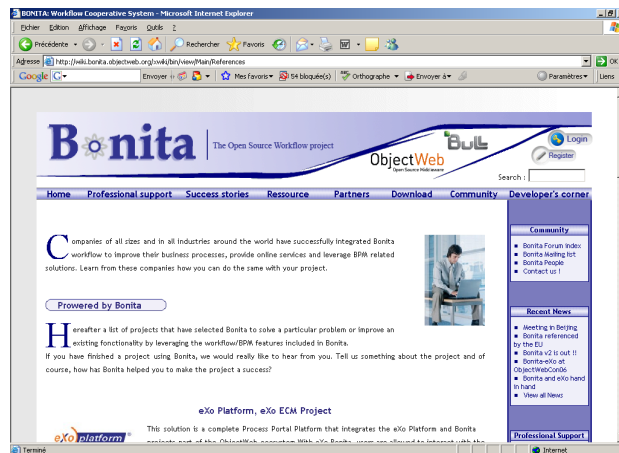
- Bonita provides out of the box workflow functionalities to define and run business processes
- LGPL license

BONITA, International recognition

- The EU Open Source Observatory references Bonita
- Communication and articles
 - Bonita v2, “The Trilogy”
 - A set of articles on Bonita v2 series published on TheServerSide community
 - Regular announcements on JavaWorld, JavaNet, InfoQ, PCQuest, TheServerSide...
- Maturity
 - End users throughout the world
 - ISV integration

BONITA's new web site !

- New web site based on xWiki's ObjectWeb component: <http://bonita.objectweb.org>



BONITA history

- Started at ECOO Team of INRIA in 2001
- First prototype using EJB1.1, BMP entity beans
- First version was developed from scratch using CMP 2.0, xdoclet code generation, struts framework, etc.
- Bonita joined ObjectWeb in February 2002
- Bull R&D leading the project from summer 2003
 - JOnAS application server support
 - Professional services and tools through the “Open Energy” offer

<http://www.bull.com/integration/libre.html>

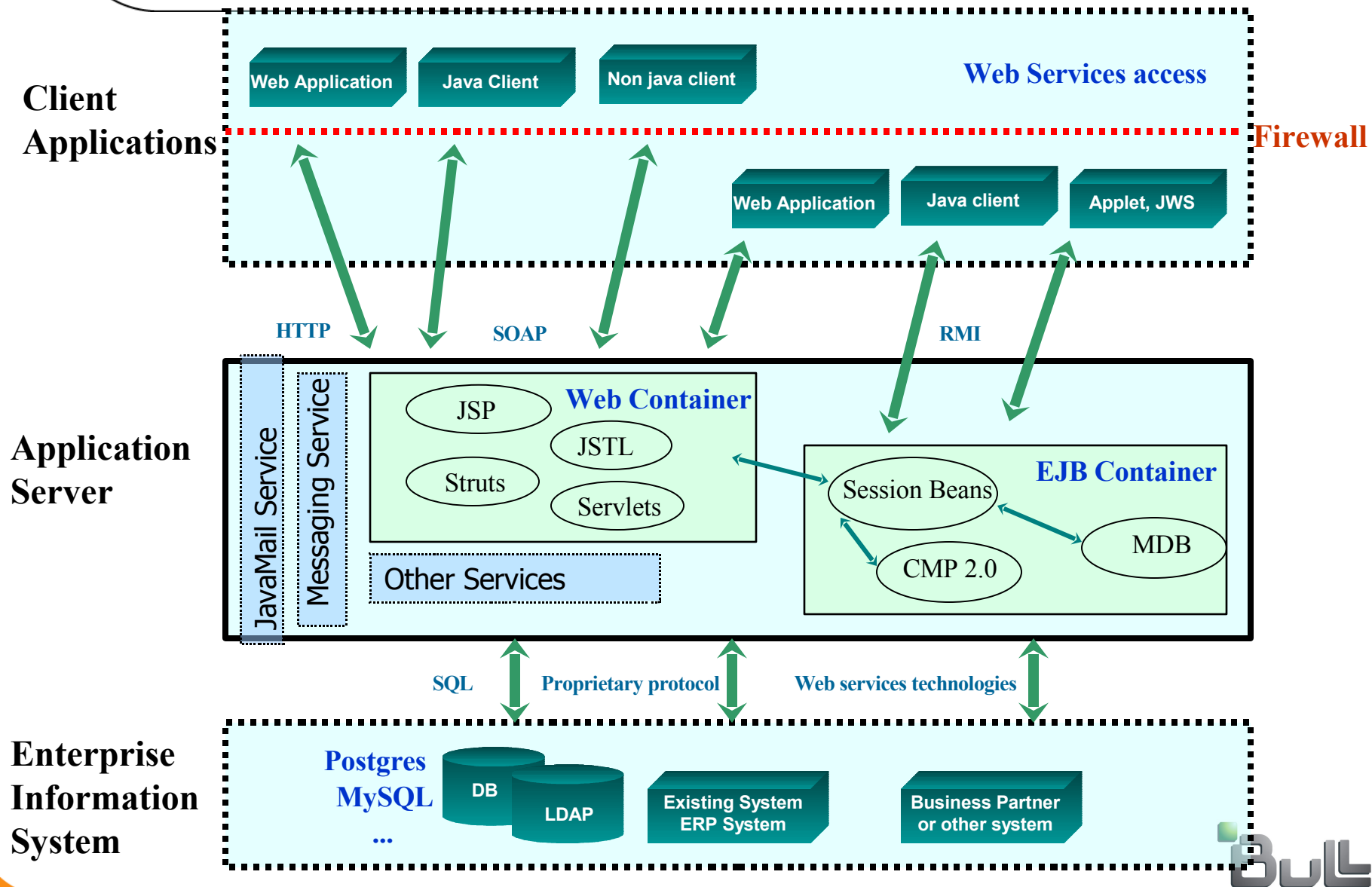
BONITA & ObjectWeb

- Active community around ObjectWeb



- Users support: mailing list, two forums, bug tracker, documentation, javadoc, user guide...
- Top 10 most downloaded projects
- Last version pre-configured for JOnAS-4.8.4

BONITA, architecture

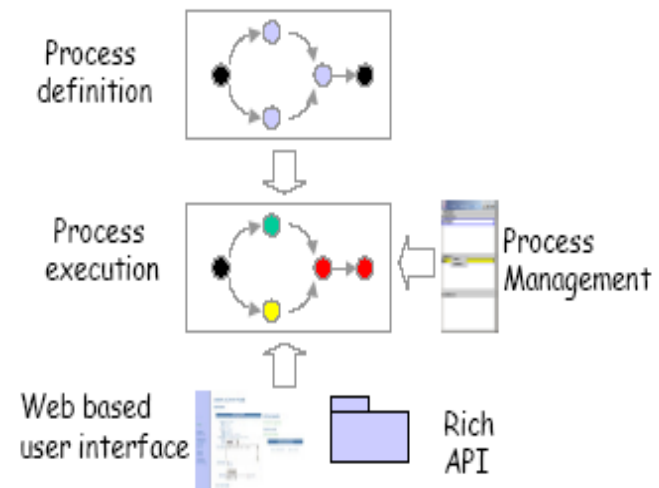


Agenda, Bonita Workflow

- Bonita introduction and workflow basics
 - Community and Objectweb relationship
 - Project history
 - Architecture
- **Bonita Highlights**
 - J2EE “Quality of Service”
 - XPDL standard support
- Bonita basics
 - Activities, Workflow data, transitions, sub processes, users, roles...
 - Hooks, performers, mappers and initiators
 - Iterations
- Workflow development
 - Bonita API's
 - Bonita stand alone client
 - Use case: The Approval Workflow sample
 - Developing hooks, mappers, performers and initiators
- Portal integration
 - eXo platform integration: workflow portlets
 - ECM features

BONITA highlights

- Benefit from the J2EE application server “Quality of Services” (QoS)
- Ability to modify the definition of a running process.
- Business and collaborative processes support
- XPDL standard compliant
- Fairly simple to use !



J2EE (QoS), web services and messaging

J2EE Workflow Web Services

- Both ingoing and outgoing web services calls are supported:
 - The workflow API is accessible through web services calls
 - External web services can be reached via Hooks

Messaging

- Awareness infrastructure
 - Bonita **J**ava **M**essage **S**ervice Module
 - Bonita Instant Messaging Service
 - Bonita Mail Service

Workflow users and roles

- User Management
 - DB by default (HSQL DB)
 - LDAP, Active Directory and other users repositories are supported thru the User API
- Workflow users and roles
 - Workflow roles are defined at project level
 - Users filling a role could change over workflow instances
 - Both explicit and programmatic modes are available for users assignment

J2EE (QoS), Security

- J2EE Authentication mechanism through JAAS
 - Standard way to configure the J2EE application security
 - Standard Java clients and Web applications can reach Bonita through login modules

```
import javax.security.auth.login.LoginContext;
import hero.client.test.SimpleCallbackHandler;

...

public class MyWorkFlowClass {

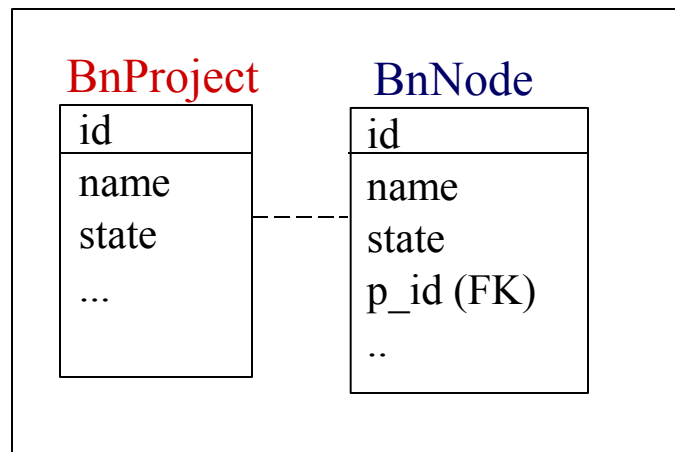
    static public void main(String[] args) throws Exception {
        // User Admin authentication
        char[] password={'t','o','t','o'};
        SimpleCallbackHandler handler = new SimpleCallbackHandler("admin",password);
        LoginContext lc = new LoginContext("TestClient", handler);
        lc.login();

        ...
    }
}
```

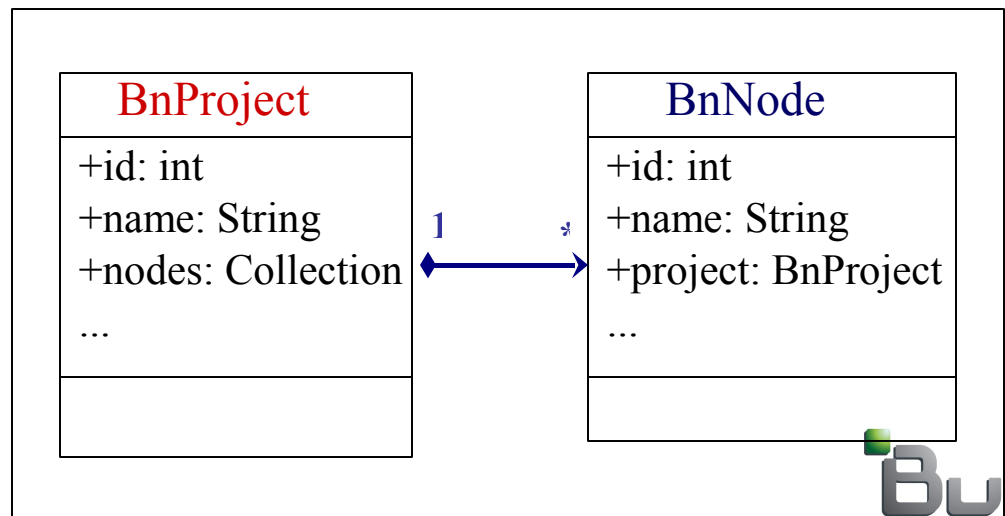
J2EE (QoS), Java-relational DB mapping

- Bonita entities are mapped using **C**ontainer-**M**anaged-**P**ersistence:
 - The container automatically maps Bonita fields to a corresponding database table with ease
 - Fields mapped in Bonita: simple objects, serializable objects, entities references and Collections

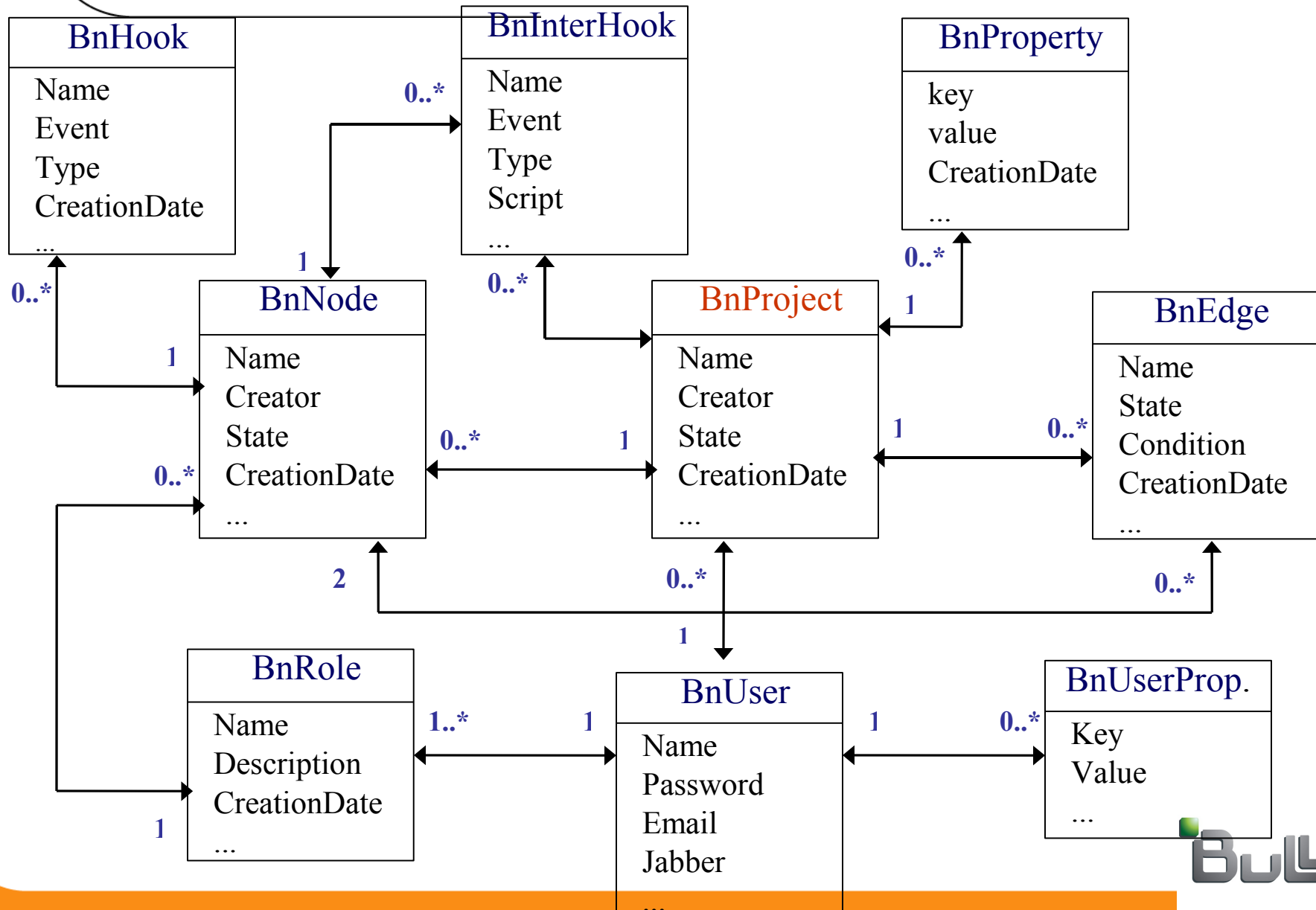
Database schema



Object Model

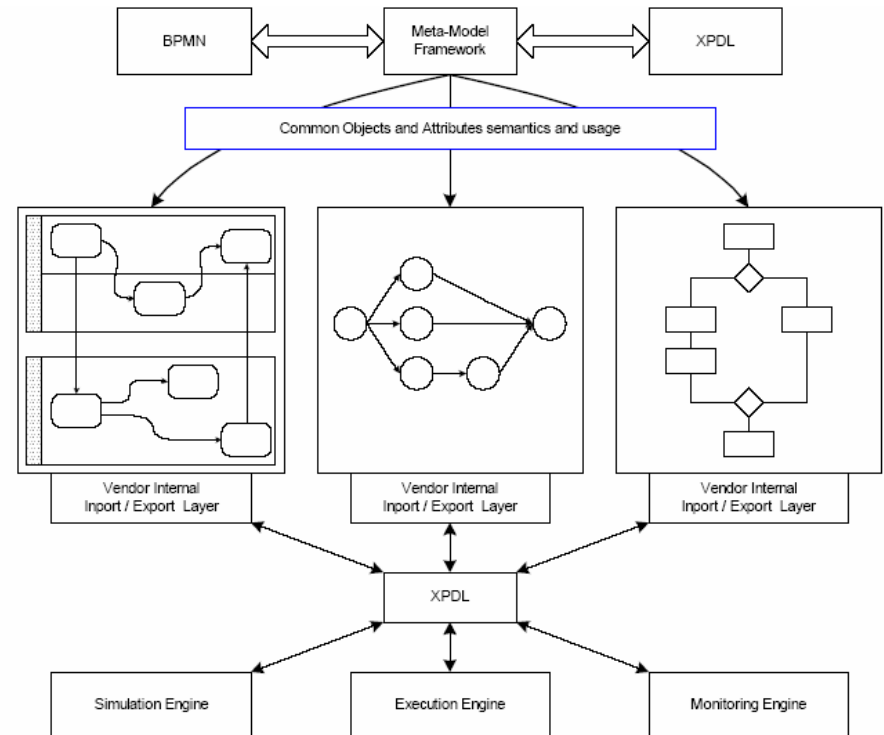


J2EE (QoS), Java-relational DB mapping



XPDL Overview

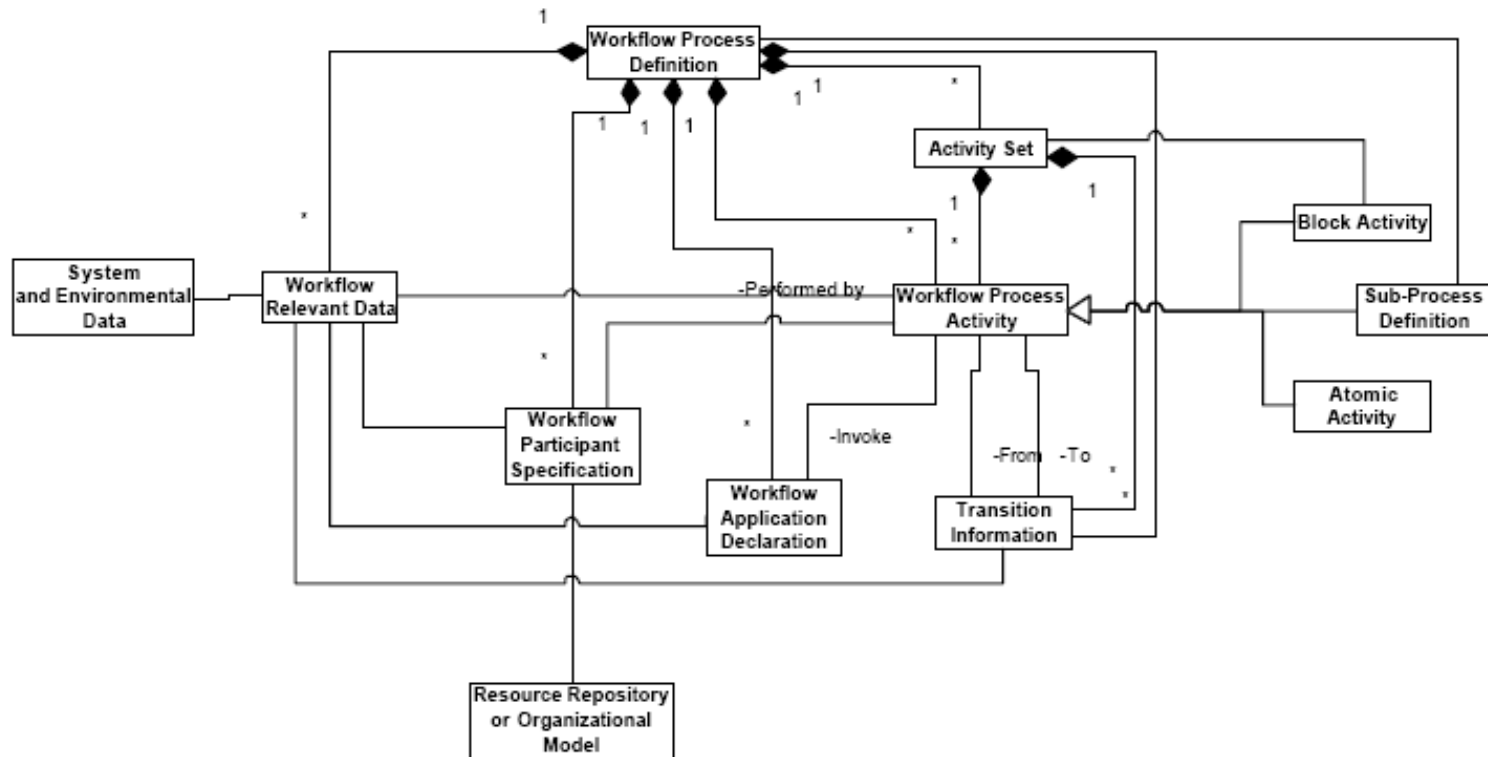
- XML Process Definition Language (XPDL)
- XML based language defined by the WfMC organization to allow workflow interoperability
- A common set of workflow entities defining a workflow process



XPDL Overview

■ XPDL Meta-Model

- Top-level entities contained within a Process Definition
- It could be extended by means of the extended attributes



XPDL Overview

■ XPDL entities description

Package	Workflow Process	Activity	Transition	Application	Data Field (Workflow Relevant Data)	Participant
- Id	- Id	- Id	- Id	- Id	- Id	- Id
- Name	- Name	- Name	- Name	- Name	- Name	- Name
- Description	- Description	- Description	- Description	- Description	- Description	- Description
- Extended Attributes	- Extended Attributes	- Extended Attributes	- Extended Attributes	- Extended Attributes	- Extended Attributes	- Extended Attributes
- XPDL Version	- Creation Date	- Automation Mode			- Data Type	- Participant Type
- Source Vendor ID	- Version	- Split				
- Creation Date	- Author	- Join				
- Version	- Codepage	- Priority				
- Author	- Country Key	- Limit				
- Codepage	- Publication Status	- Start Mode				
- Country Key	- Priority	- Finish Mode				
- Publication Status	- Limit	- Deadline				
- Conformance Class	- Valid From Date					
- Priority Unit	- Valid To Date					

XPDL Overview

■ XPDL Extended Attributes

- Additional information (user or vendor specific) included within a process definition
- To express any additional entity characteristics which need to be exchanged between systems

■ Workflow relevant data defined at activity level:

```
<DataField Id="Decision" Name="Decision">
  <DataType>
    <EnumerationType>
      <EnumerationValue Name="Yes" />
      <EnumerationValue Name="No" />
    </EnumerationType>
  </DataType>
  <InitialValue>No</InitialValue>
  <ExtendedAttributes>
    <ExtendedAttribute Name="PropertyActivity" />
  </ExtendedAttributes>
</DataField>
```

Agenda, Bonita Workflow

- Bonita introduction and workflow basics
 - Community and Objectweb relationship
 - Project history
 - Architecture
- Bonita Highlights
 - J2EE “Quality of Service”
 - XPDL standard support
- **Bonita basics**
 - **Activities, Workflow data, transitions, sub processes, users, roles...**
 - **Hooks, performers, mappers and initiators**
 - **Iterations**
- Workflow development
 - Bonita API's
 - Bonita stand alone client
 - Use case: The Approval Workflow sample
 - Developing hooks, mappers, performers and initiators
- Portal integration
 - eXo platform integration: workflow portlets
 - ECM features

Bonita basics...

■ Two processes types:

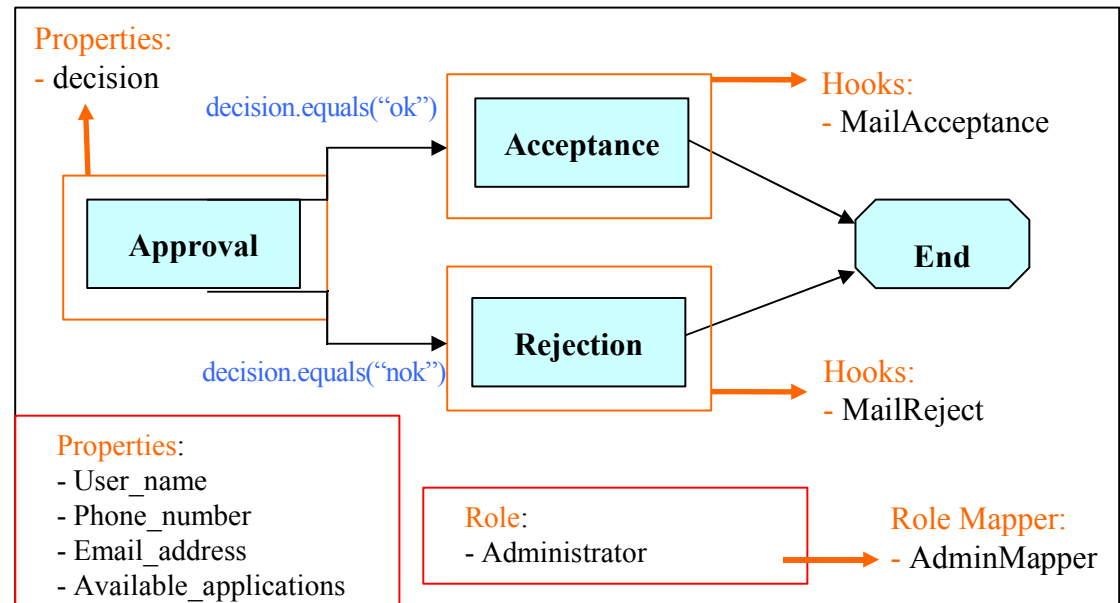
- Models
- Cooperatives

■ Advanced features:

- RoleMapper
- PerformerAssigns
- Hooks

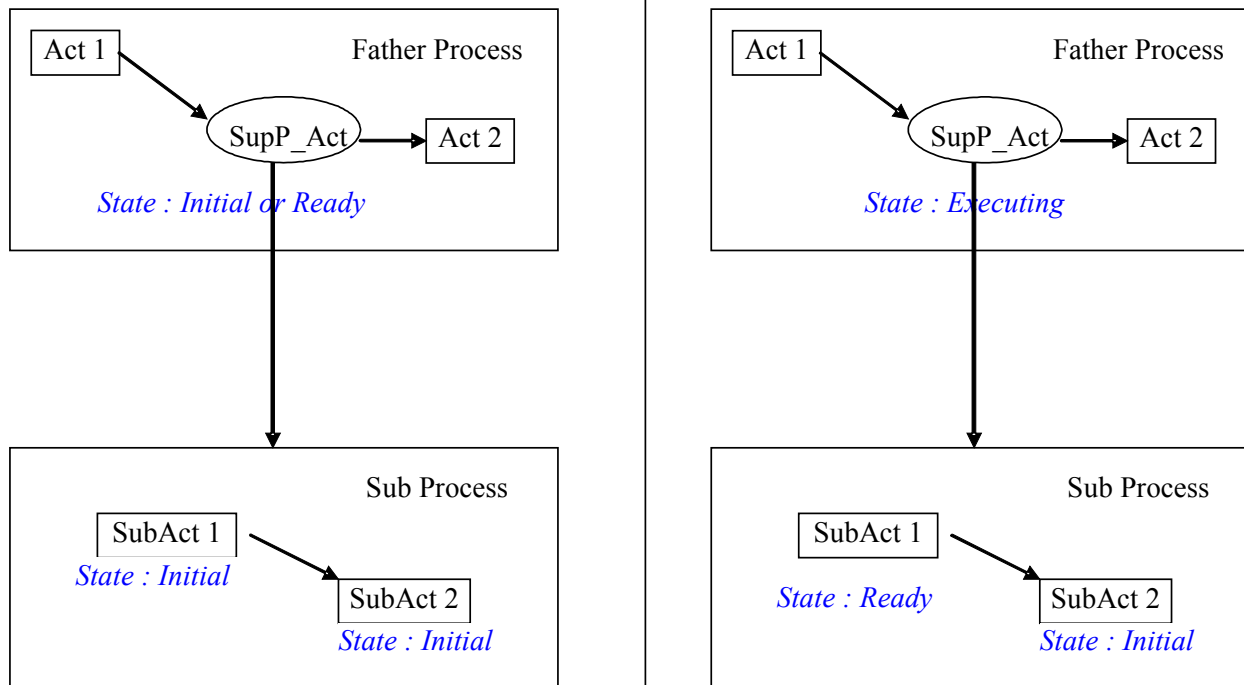
■ Workflow entities

- Activities
- Transitions/Edges
- Conditions
- SubProjets
- Iterations
- Properties
- Users
- Roles



Bonita basic, sub processes

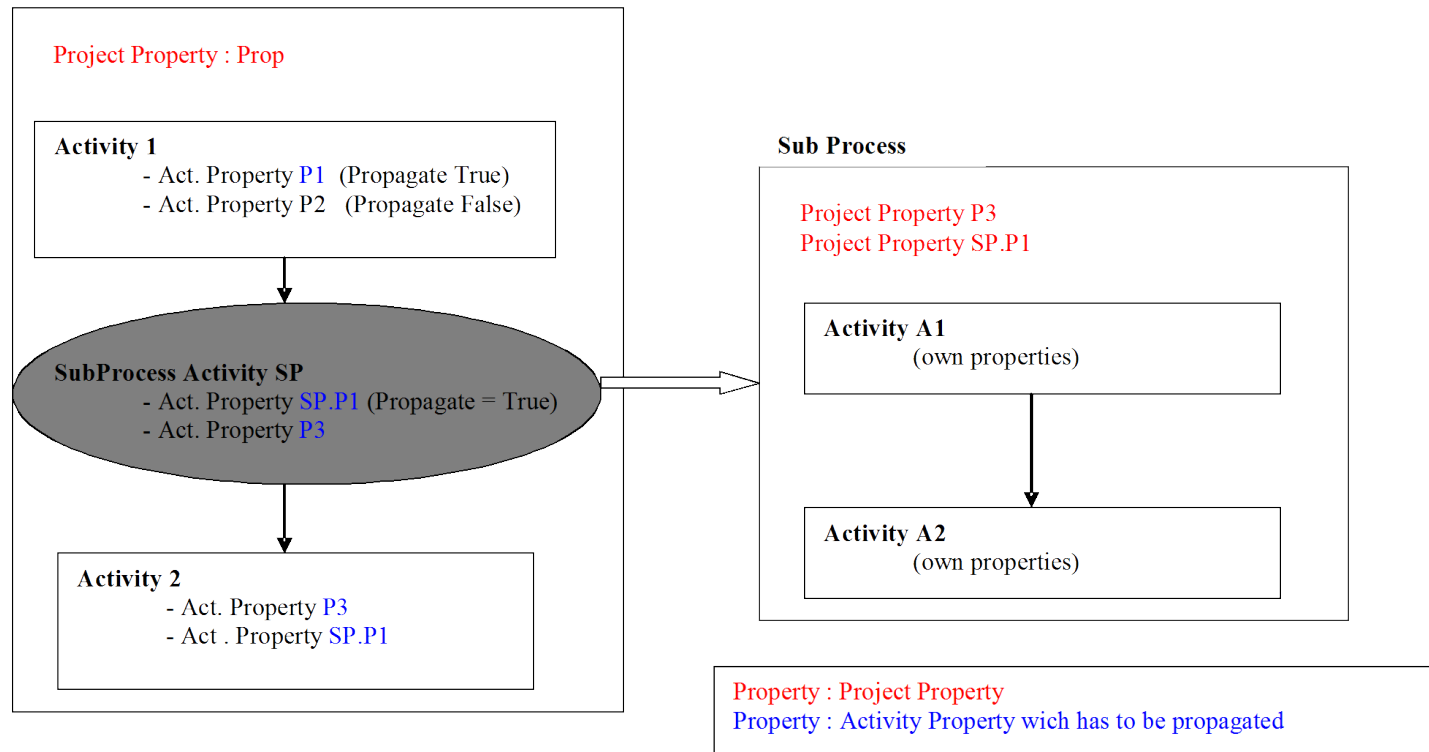
- Independently existing business process which takes part in another more sophisticated process
- SubProcess activities are started and terminated automatically by the Workflow engine



Bonita basic, workflow data

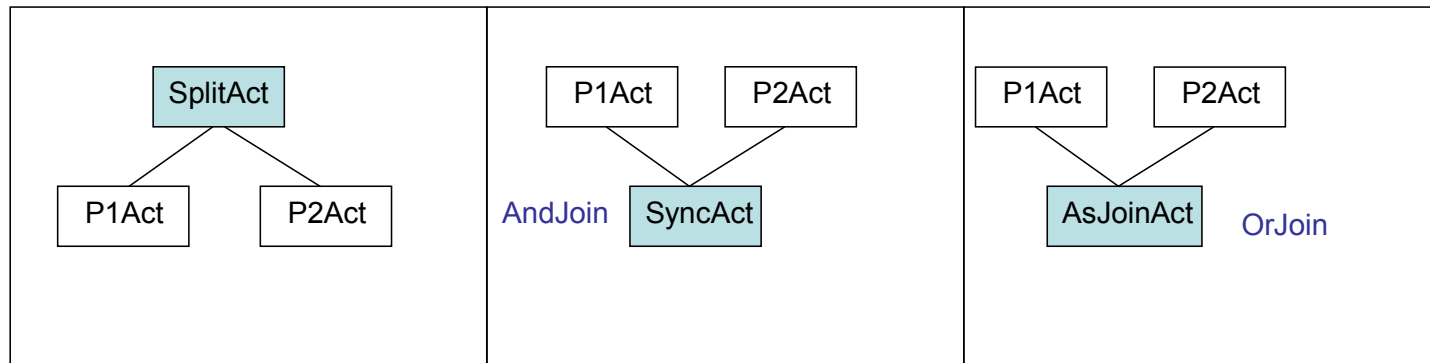
- A **property** is a workflow unit of data, commonly known as workflow relevant data.
 - Project and activity level definition
 - Properties propagation during the execution at activity level

Process



Bonita basic, transitions

- **A transition** is a dependency expressing an order constraint between two activities. In BONITA, transitions are also termed **Edges**.
- The transition pattern is determined according to the type of the activity



Bonita basics, hooks

Hooks, the workflow magic !

- Used to connect to both internal and external information systems
- Two types:
 - Interactive Hooks: beanshell language
 - Java Hooks: Java files
- They are associated to the following events
 - Before/After start
 - Before/After terminate
 - On Cancel, On Ready, On DeadLine

Bonita basics, mappers and performers

■ Role Mapper

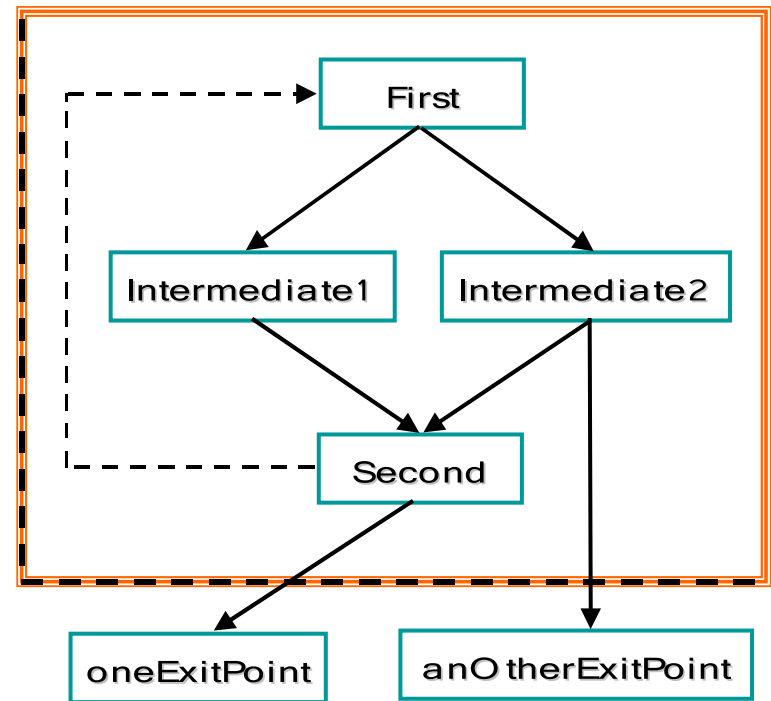
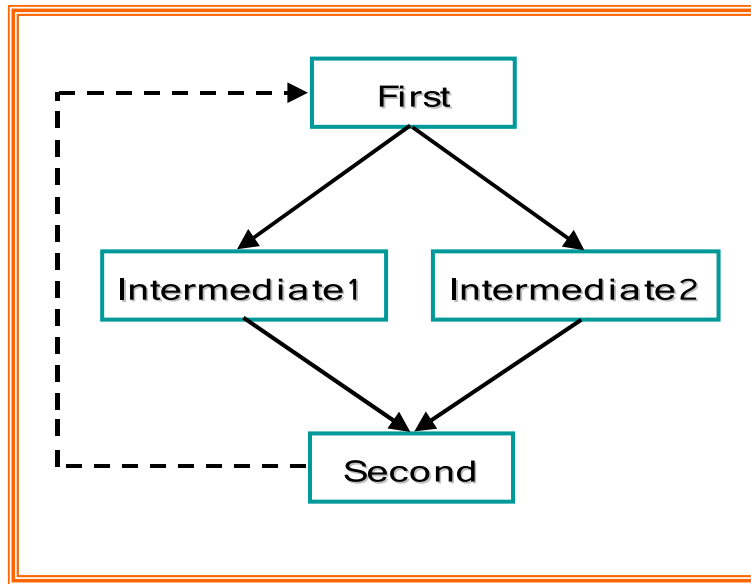
- Entity which fill automatically workflow roles in a process instance
- “Instance creation time”
- Directly plugged with the Information System
- Mapper types: Custom, LDAP and Property

■ Performer Assignment

- Entity which modifies the standard assignment rules for activities
- “Instance execution time”
- Performer Types: Callback, Property

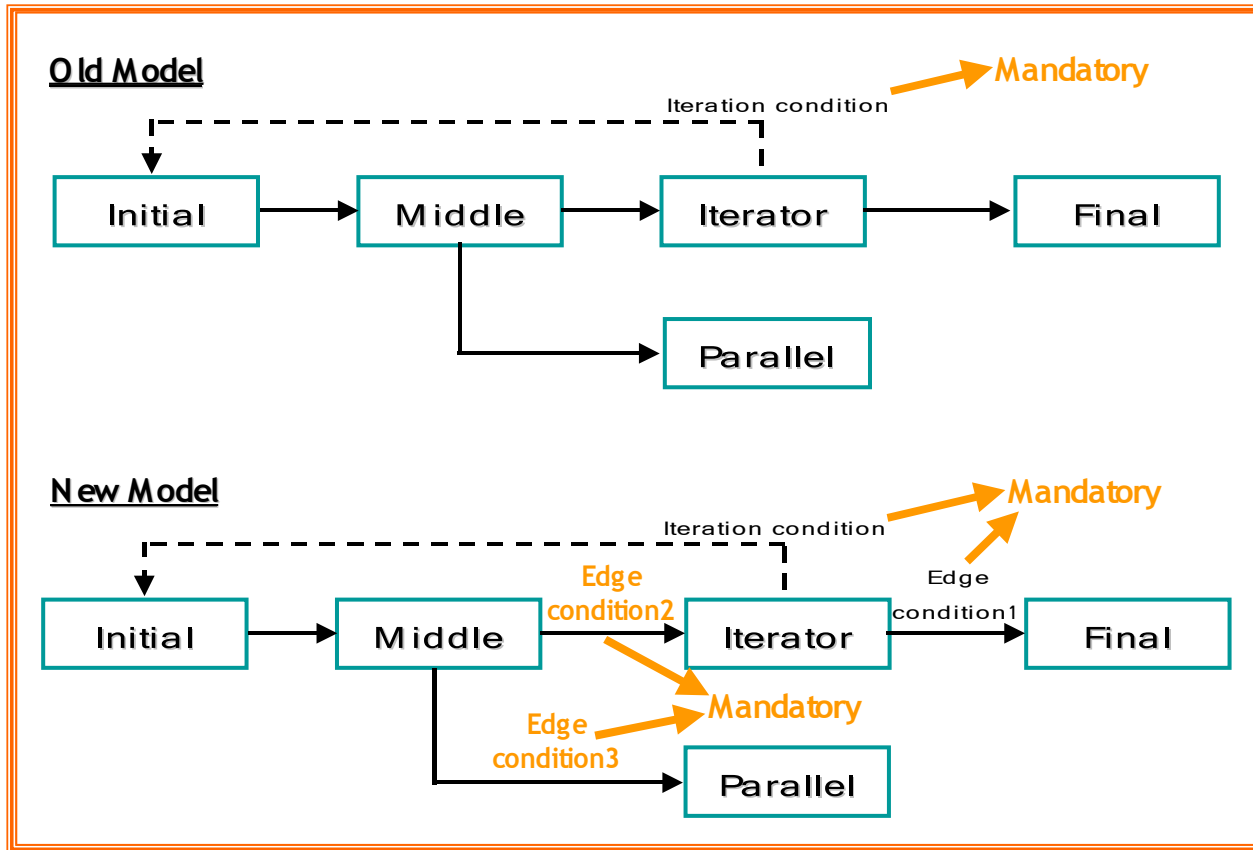
Bonita basics, iterations (1/2)

- Both structured and arbitrary cycles are supported by Bonita



Bonita basics, iterations (2/2)

- Bonita v2 adds full support for arbitrary cycles through conditions



Bonita basics, versioning

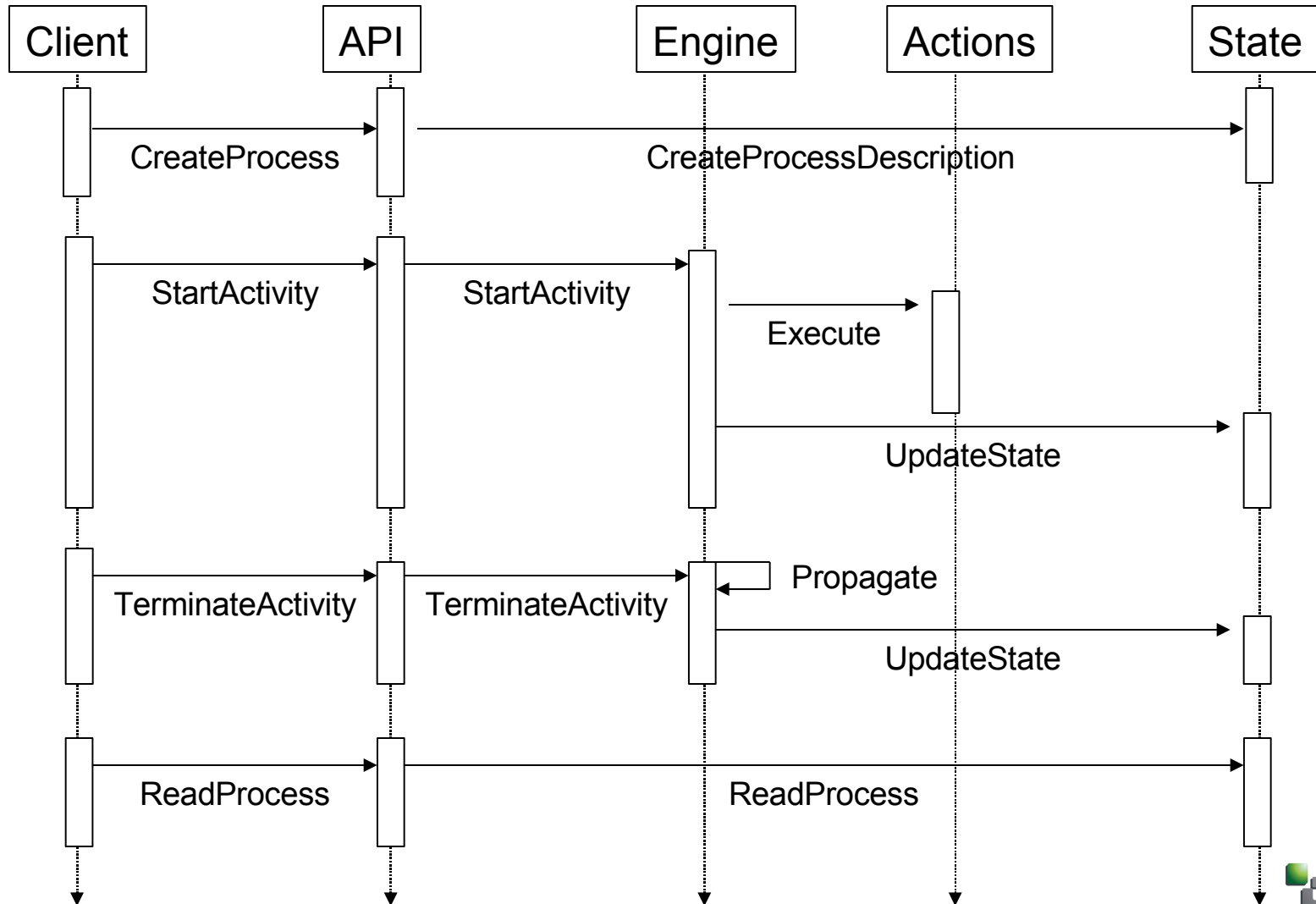
Workflow model versioning

- Bonita v2 introduces versioning capabilities
 - Multiple versions of a same workflow model are allowed
 - Progressive and intuitive migration from one workflow version to another
 - Undeploy state was added to the workflow model life cycle
 - Current users can still working with one version
 - New users will start working with the latest version

Agenda, Bonita Workflow

- Bonita introduction and workflow basics
 - Community and Objectweb relationship
 - Project history
 - Architecture
- Bonita Highlights
 - J2EE “Quality of Service”
 - XPDL standard support
- Bonita basics
 - Activities, Workflow data, transitions, sub processes, users, roles...
 - Hooks, performers, mappers and initiators
 - Iterations
- **Workflow development**
 - **Bonita API's**
 - **Bonita stand alone client**
 - **Use case: The Approval Workflow sample**
 - **Developing hooks, mappers, performers and initiators**
- Portal integration
 - eXo platform integration: workflow portlets
 - ECM features

BONITA, main interaction diagram



Bonita API's

■ The ProjectSession API

- Functions allowing to define a workflow process.
 - Creation of activities, transitions, roles, actions, workflow data...
 - Process instantiation
- Project interface will automatically retrieve the identity of the calling user in the J2EE security context.

■ The UserSession API

- The User interface provides access to process execution control functions.
 - Retrieve useful information about the workflow processes/instances (todo list, running list...)
 - Execute a particular workflow action (start, terminate, cancel)
- User interface will automatically retrieve the identity of the calling user in the J2EE security context.

Bonita, fairly simple to use !

Workflow Definition API

```
public class ProcessDefinition {  
    static public void main(String[] args) throws  
        HeroException{  
  
        // Create workflow project  
        ProjectSessionHome hm =ProjectSessionUtil.getHome();  
        ProjectSession rm = hm.create();  
        rm.initModel("e-citizen");  
  
        // Add new activity to the workflow process  
        rm.addNode("Subs Order",Constants.Nd.AND_JOIN_NODE);  
        rm.setNodeTraditional("Subs order");  
    }  
}
```

Bonita, fairly simple to use !

Workflow XPDL API

- The XPDL support in Bonita has been reached thanks to the Bonita XPDL module
- This module will parse the XPDL file and call the ProjectSession Bean API.
- To import a XPDL file, go to the BONITA_HOME directory and type:

```
ant import-xpdl -Duser="USER_NAME" -Dpasswd="PASS" -Dxpdl="XPDL_PATH"
```

Bonita, fairly simple to use !

Workflow Execution API

```
public class ProcessExecution {  
    static public void main(String[] args) throws  
        HeroException{  
  
        String instName = rm.instantiateProject("e-citizen");  
  
        // Get Todo List  
        UserSessionHome uHome=UserSessionUtil.getHome();  
        UserSession uSession = uHome.create();  
        Collection nodes = uSession.getToDoList(instName);  
    }  
}
```

Bonita API's, sample application

- Developing a workflow stand alone application
 - “Bonita Simple Project Ever” sample
 - Java client application
 - Using Jakarta Ant to compile and run the sample
 - Java web applications can be developed in the same way
 - Jar dependencies:
 - bonita-client.jar: workflow API's
 - client.jar: JOnAS server client container
 - Configuration settings:
 - carol.properties file: rmi protocol, port and host address in which the workflow server is running
 - auth.conf: security configuration file
 - security and naming properties:
 - java.security.auth.login.config
 - java.naming.factory.initial

Case study

- Approval workflow sample
 - Bonita, “The Trilogy”, part one article
 - Identifying involved workflow entities
 - Workflow definition
 - Java file
 - XPDL file
 - Workflow execution
 - Workflow instances creation
 - Retrieving workflow data through the “Pagination API”
 - Performance
 - Test scenario
 - Test results



Workflow Development

Developing hooks, mappers, performer assignments and initiators

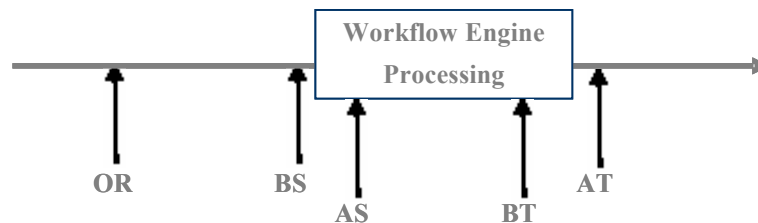
Developing Hooks

- External java classes performing user defined operations
- Hooks can be triggered at different moments of a workflow process execution
- Hooks can reach the Workflow API's as well as external system thru Java, JCA connectors, Web Services calls
- Hooks types:
 - Process level: onInstantiate
 - Activity level: onReady, afterStart, beforeStart, afterTerminate, beforeTerminate, onDeadline, onCancel

Developing Hooks

- Manual vs automatic activities
- Hooks execution life cycle

Workflow hook name	Short name	Automatic activity	Manual activity
onReady	OR		x
beforeStart	BS		x
afterStart	AS		x
beforeTerminate	BT	x	x
afterTerminate	AT	x	x
onDeadline	OD		x
onCancel	OC		x



Developing Hooks

■ Writing a Hook

- Java class which implement either NodeHookI or ProjectHookI interfaces

```
- package hero.hook;
import hero.interfaces.*;
public class myHookOnReady implements hero.hook.NodeHookI {
    public String getMetadata() {return Constants.Nd.ONREADY;}
    public void onReady(Object b,BnNodeLocal n) throws
        HeroHookException {
        System.out.println ("Hello world");
    }
    - //Other Events...
}
```

■ Compiling and deploying a Hook

- Ant task which performs those operations:
 - *ant deployHook -DhookClass=<name of you java source file>*

Developing Mappers and Initiators mappers

- Those Java classes are both aimed to designate persons
 - Mapper Java class is aimed to designate the persons corresponding to a specific user defined role
 - Resolved at process instantiation time
 - Users-Role mapping
 - Initiator mapper Java class is aimed to designate the persons authorized to start a process
 - Filter the instantiation of a process

Developing Mappers

■ Writing a Mapper

- Java class which implement the RoleMapperI interface

```
- package hero.mapper;  
  import hero.util.HeroException;  
  import java.util.*;  
  public class CustomGroupMembers implements hero.mapper.RoleMapperI {  
      public Collection searchMembers(Object b,BnRoleLocal n, String  
      userName) throws HeroException {  
          Collection users = new ArrayList();  
          users.add("bsoa");  
          return users;  
      }  
  }
```

■ Compiling and deploying a Mapper

- Ant task which performs those operations:
 - *ant deployMapper -DmapperClass=<name of you java source file>*

Developing Initiator Mappers

■ Writing a Initiator Mapper

- Java class which implement the InitiatorMapperI interface

```
- package hero.initiatorMapper;
import hero.util.HeroException;
import hero.interfaces.BnProjectLocal;
import java.util.*;
public class CustomGroupMembers implements
    hero.initiatorMapper.InitiatorMapperI {
    public Collection searchMembers(Object b,BnProjectLocal n, String
        userName) throws HeroException {
        ArrayList users = new ArrayList();
        users.add("bsoa");
        return users;
    }
}
```

■ Compiling and deploying a Initiator Mapper

- Ant task which performs those operations:
 - *ant deployInitiatorMapper -DinitiatorMapperClass=<name of you java source file>*



Developing Performer Assignments

- Entity which modifies the standard assignment rules for activities
 - Resolved at activity execution time
 - Commonly used together with Mappers
- Performer assignments plays a key role in the delegation process
 - Together with the Mapper entity

Developing Performer Assignments

■ Writing a Performer Assignment

- Java class which implement the NodePerformerAssignI interface

```
- package hero.performerAssign;
import hero.interfaces.*;
public class PropertySelectActors implements
    hero.performerAssign.NodePerformerAssignI {
    public void selectActors(Object b,BnNodeLocal n, String userName) throws
        HeroException {

        try {
            n.setActivityPerformer("bsoa");
        } catch (Exception e) {e.printStackTrace(); throw new WorkflowException
            (e.getMessage());}
    }
}
```

■ Compiling and deploying a Performer assignment

- Ant task which performs those operations:
 - *ant deployPerformer -DperformerClass=<name of you java source file>*

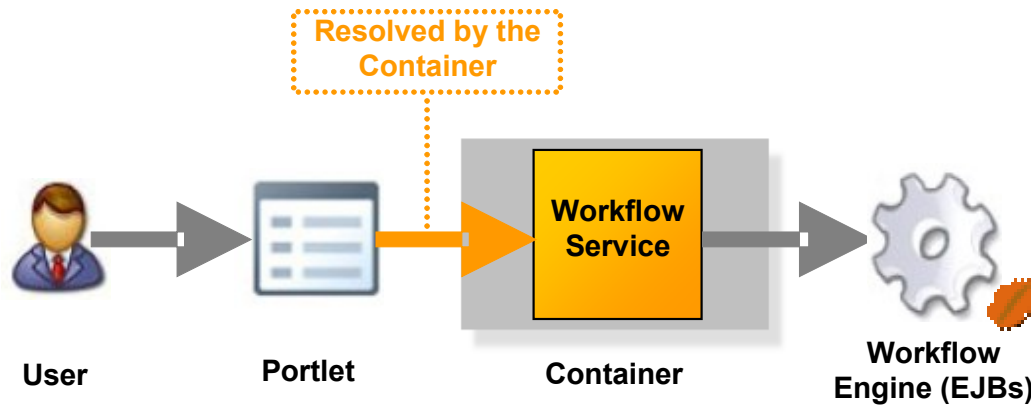
Agenda, Bonita Workflow

- Bonita introduction and workflow basics
 - Community and Objectweb relationship
 - Project history
 - Architecture
- Bonita Highlights
 - J2EE “Quality of Service”
 - XPDL standard support
- Bonita basics
 - Activities, Workflow data, transitions, sub processes, users, roles...
 - Hooks, performers, mappers and initiators
 - Iterations
- Workflow development
 - Bonita API's
 - Bonita stand alone client
 - Use case: The Approval Workflow sample
 - Developing hooks, mappers, performers and initiators
- Portal integration
 - eXo platform integration: workflow portlets
 - ECM features

Bonita, portal integration



- Integration with eXo platform
 - Taking the best from both portal and workflow worlds
 - Presentation, personalization, and integration features from eXo
 - Automation, modeling and collaboration of Bonita
- End user will interact with the Bonita through portlets



Execution portlet

Workflow portlets: eXo portal

- Workflow deployment, management, and monitoring through JSR 168 Portlets

AdminWorkflow			
Processes Monitor Timers Monitor			
Process id	Process Name	Process version	Action
1	content-backup	1	View Delete
2	payraise	1	View Delete
3	content-validation	1	View Delete
4	holiday process	1	View Delete

Administration portlet

BP Definition Controller Tasks Controller			
Process id	Name	Version	Action
2	payraise	1	Start
4	holiday process	1	Start

ToDoList portlet

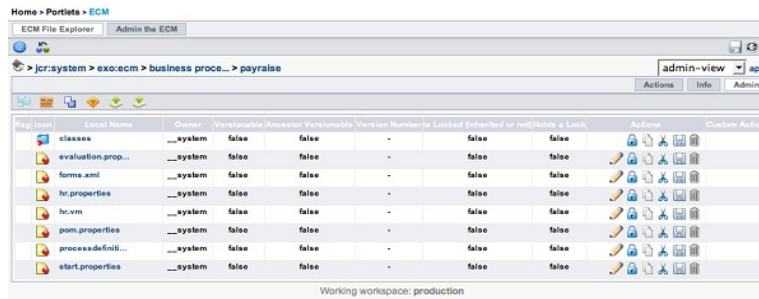
BP Definition Controller Tasks Controller	
Request a pay raise	
Amount :	<input type="text"/>
Priority :	<input type="text" value="not important"/>
Reward :	<input type="checkbox"/>
Reason :	<div></div>
Let's play cancel	

Execution portlet

Content management: eXo ECM

- Document management processes based on the JCR and the eXo ECM component

JCR browser portlet

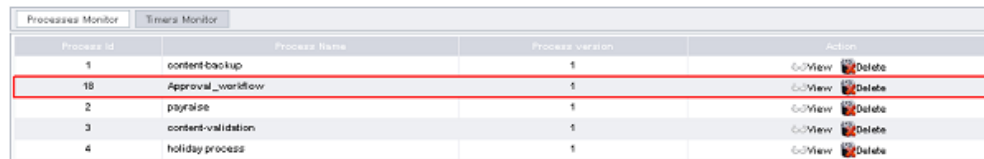


Home > Portlets > ECM
ECM File Explorer Admin the ECM
> jcr:system > exo:ecm > business proce... > payrollse admin-view apply
Actions Info Admin

Item icon	Local Name	Owner	Versionable/Assessable	Versionable	Version Number/Is	Locked	Subscribed or not/Hide a Lock	Actions	Custom Actions
classes	__system	false	false	-	false	false			
evaluation.prop...	__system	false	false	-	false	false			
forms.amd	__system	false	false	-	false	false			
hr.properties	__system	false	false	-	false	false			
hr.vrm	__system	false	false	-	false	false			
pom.properties	__system	false	false	-	false	false			
process.definit...	__system	false	false	-	false	false			
start.properties	__system	false	false	-	false	false			

Working workspace: production

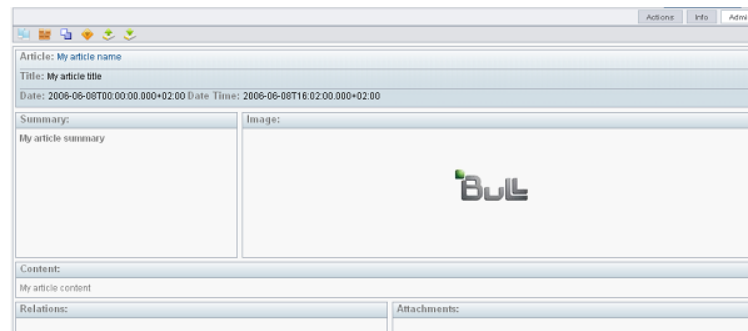
Workflow portlet



Processes Monitor Timers Monitor


Process Id	Process Name	Process version	Action
1	content-backup	1	Go View Delete
18	Approval_workflow	1	Go View Delete
2	payrolse	1	Go View Delete
3	content-validation	1	Go View Delete
4	holiday process	1	Go View Delete

Document sample



Article: My article name
Title: My article title
Date: 2008-06-08T00:00:00.000+02:00 Date Time: 2008-06-08T16:02:00.000+02:00

Summary: My article summary

Image: 

Content: My article content

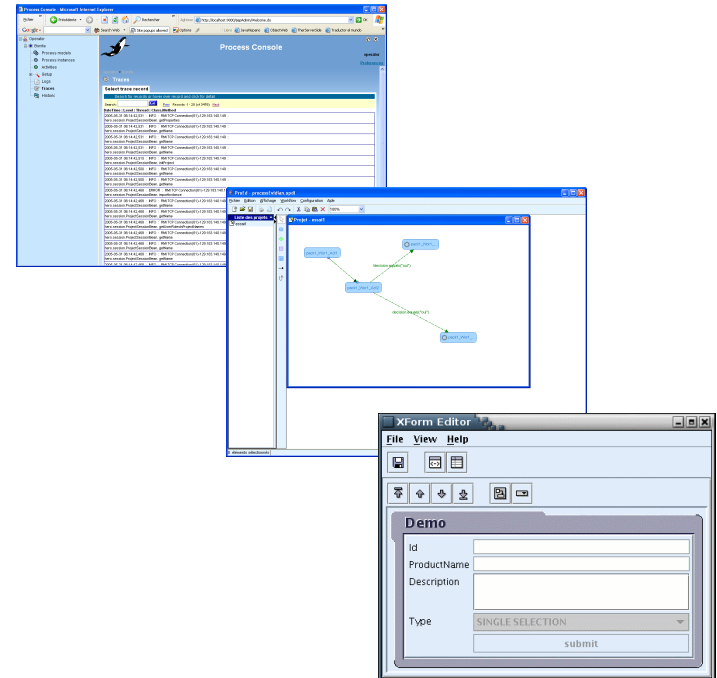
Relations: Attachments:

Agenda (Bonita Workflow Platform)

- **Bonita Workflow Platform features**
- **Bonita Workflow Console: the common workflow environment**
 - **Administrator's profile**
 - User Management and configuration
 - Customizing the Workflow Console
 - **Conceptor's profile**
 - Defining a workflow process (ProEd)
 - Steps to design a workflow process
 - Developing Hooks, mappers, performer assignments and initiator mappers
 - Customizing web forms with the xform Editor
 - **User's profile**
 - User's Worklist: start list, running list, todo list and done list
 - **Operator's profile**
 - Operator's worklist
 - Global settings setup
 - User delegation thru performer assignments
 - Visualizing logs, traces and history files
- **Appendices**
 - Recovering Bonita Workflow Platform data
 - Bonita Workflow Platform documentation
 - Tuning
 - Integration with Orchestra
 - Current development

Bonita Workflow Platform Features

- Integrated environment for workflow users



- **Generic workflow console**

- Administrate Information System
- Survey and manage workflow execution
- Define and deploy workflow processes
- Handle user Interaction

- **Common entry point for users**

- Configurable user management
- Users profiles

- **Dedicated tools**

- Graphical workflow Editor
- Web forms Editor



Thanks for your attention !

Workflow Project Manager
miguel.valdes-faura@bull.net