Business Process Management with Spagic 3

Author Luca Rossato
Gianfranco Boccalon

1 Document Goal

The goal of this document is to provide you with an introduction on using Spagic Workflow features in Spagic platform by designing, deploying and monitoring some sample workflow processes.

2 Versions History

Version/Release n° :	1.0	**Date**	July 30, 2010
Description	First release (English version)		

3 Related documents

This document comes after **Spagic 3 Getting Started**. It doesn't replace their contents, it expands the explanations to other functionalities of the Spagic platform in order to create and execute workflow processes using Spagic platform.

4 Introduction

4.1 Spagic Studio Overview

Spagic Studio IDE is composed as a set of tools and graphical editors, based on the eclipse platform. In this document we focus attention on these features of Spagic Studio:

- Graphical BPMN Editor (based on standard eclipse stp BPMN editor).
- Set of extensions to add to BPMN files, technology and Spagic detailed information on BPMN files.
- Drag And Drop technology specific services/service bindings from a runtime view to BPMN Flow elements.
- Fill technologies detailed properties, for each BPMN task after you've annotated this with a particular service.
- Generate and Deploy processes with both human and/or automatic tasks.

4.2 Modeling processes with BPMN items used in Spagic 3

While modeling processes you can use a predefined set of BPMN items already contained in Spagic 3such as: events of start and end, task and looping task, gateway parallel or exclusive with conditions and much more. In Spagic 3 there is also the possibility to associate attributes to a process.

In Spagic3 you can model processes where there are different actors that contributes to the process execution by drawing the process using different swimlanes inside the main pool lane and defining all the actors that will execute each human task of the process.

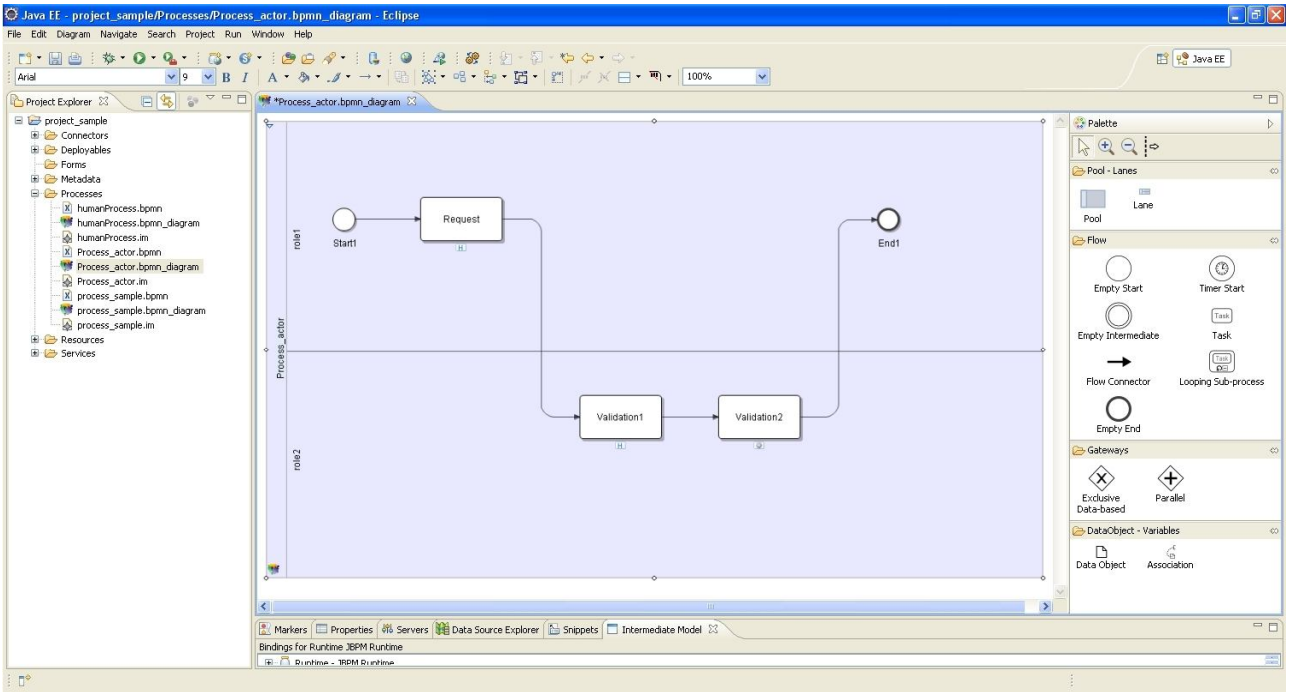


Image 1 - Process with human Tasks and two different actors

The swimlane title refers to the user role that can execute tasks inside it, in this way you can give the responsibility of a task execution to an actor belonging to a defined role. In the *Image 1* there are two different roles declared (role1 and role2) that will execute respectively the Request and the Validation tasks. The explanation on how to obtain the mapping between process users and roles we refer to the paragraph 4.3.4.

4.3 Process elements

Any process is composed by different types of elements, described below you can find the main elements we will use for our examples. Annotations are not necessary to define or run process elements, but, in some cases, can aid tasks filtering for the execution.

4.3.1 Start

Start elements only define the first entry point of a workflow process.

The pictures above represent the start element in its two forms: without annotation and with annotation. There is no substantial difference between the two forms.

4.3.2 End

End elements defines end points of a workflow process.

4.3.3 Task

Tasks must have a unique name in the process and can be annotated or not. Annotated tasks can hold attributes that can be used by the workflow API to filter tasks in filtered lists.
The pictures below represent a task without annotation, a task with "Human Task" annotation and a task with "Automatic Task" annotation:

4.3.4 Transitions

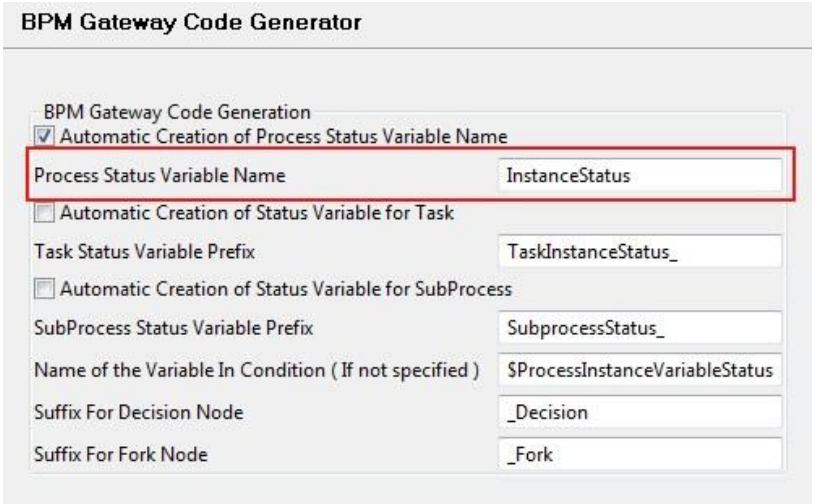
All steps must be linked with the element Transition.

The transition usually is not annotated, with the exception explained in the next paragraph.

4.3.4.1 Shortcut for Exclusive gateways

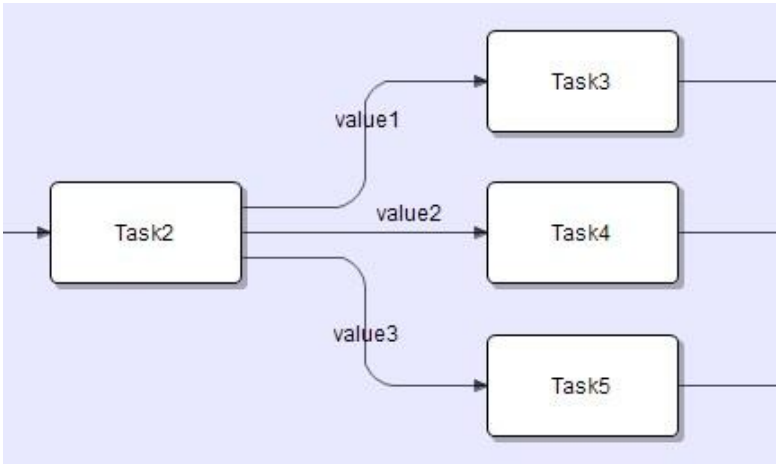
If a label is added to a transition, it's automatically interpreted as a conditional transition. The transition is taken only if the <*InstanceStatus*> variable is equal to the value of the label.

The name of the <*InstanceStatus*> variable is defined in the Spagic preferences:

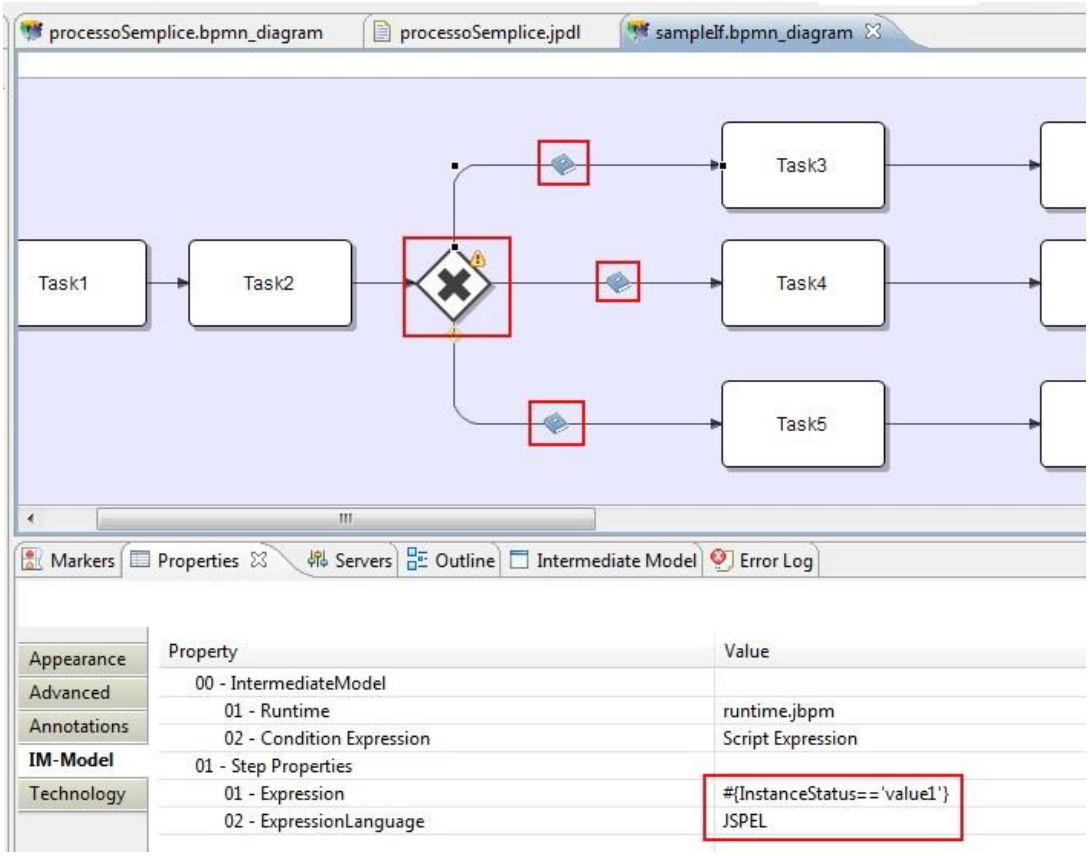


This is a shortcut to implement quickly the exclusive gateway, without writing too many expressions.

For example, this process fragment:



is fully equivalent to this fragment:



where the transitions have been explicitly annotated with the expression containing the condition to evaluate, and an exclusive data based gateway has been inserted.

4.3.5 Lanes

Lanes are intended to assign human tasks execution to some users that have the role and permissions to run those actions.

In the picture below there is an example where the process is split in two zones, the part up where there are tasks assigned to the users of the role1 and the part down where there are role2's tasks.

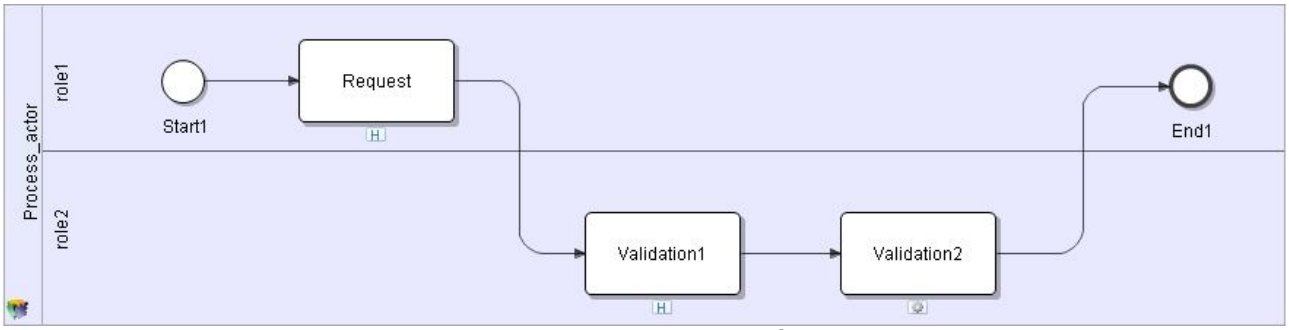


Image 2 - Lanes example.

In order to define users and roles you have to modify userprofile.xml file under the target directory you defined. (As explained in Spagic 3 Getting Started document).

The following is a very simple implementation of the authorization mechanism. More roles and user can be defined in order to increase hierarchy and operation security. The concept is to associate a user (normally a string provided by the application caller) with a role (normally a string reported as name of the lane).

```
<!-- Example authorization file -->
<AUTHORIZATIONS>
	<ENTITIES>
		<USERS>
			<USER userID="su" password="su" />
			<USER userID="actor1" password="actor1" />
			<USER userID="actor2" password="actor2" />
		</USERS>
		<ROLES>
			<ROLE roleName="su" description="Super User" />
			<ROLE roleName="actor1" description="Actor User" />
			<ROLE roleName="actor2" description="Actor User" />
		</ROLES>
	</ENTITIES>
	<RELATIONS>
		<BEHAVIOURS>
			<BEHAVIOUR userID="su" roleName="su"/>
			<BEHAVIOUR userID="actor1" roleName="actor1"/>
			<BEHAVIOUR userID="actor2" roleName="actor2"/>
</BEHAVIOURS>
	</RELATIONS>
</AUTHORIZATIONS>
```

4.3.6 How to implement you profile service inside Spagic

In Spagic, you can use every profile system that retrieves the roles written in the BPMN process diagram from username. You just have to write an component inside an OSGI bundle that implements the interface IProfileService that is in the plugin org.spagic.workflow.api. Here we explain how to build an implementation in this component.

1. Create an OSGI plugin running under equinox.
2. In the MANIFEST.MF file, you have to add **org.spagic.workflow.api** in the **import-package**.
3. Create a component, and add in its xml description file that provides the service IProfileService.

For example, the xml description file can be written in this way

```
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0"
	name="CustomProfileService">
   <implementation class="CustomProfileService"/>
   <service>
      <provide interface="org.spagic.workflow.api.IProfileService"/>
   </service>

</scr:component>
```

4. In the implementation class you can put your custom way to retrieve roles form user implementing the method getUserRoles. Here there is a sample of the implementation class of the component CustomProfileService:

```
import org.spagic.workflow.api.IProfileService;
import org.osgi.service.component.ComponentContext;

public class CustomProfileService implements IProfileService {

    protected void activate(ComponentContext componentContext){

    }

    protected void deactivate(ComponentContext componentContext){
    }


    public List<String> getUserRoles(String actorId) {

    //  implement here the logic of retrieving roles from users
    }

    public List<String> getUsers() {

    }
```

5. Export the bundle as Deployable Plugins and Fragments and put the plugin in SERVICE_MANAGER_FOLDER/plugins . Remove the plugin org.spagic3.profile.simple from the same folder.
6. In SERVICE_MANAGER_FOLDER\configuration\org.eclipse.equinox.simpleconfigurator\bundles.info and delete

```
org.eclipse.equinox.simpleconfigurator,1.0.100.v20090520-
1905,plugins/org.eclipse.equinox.simpleconfigurator_1.0.100.v20090520-
1905.jar,4,true
```

and insert

```
NAME_OF_PLUGIN,VERSION_OF_PLUGIN
,plugins/NAME_OF_THE_PLUGINS_WITH_ALSO_JAR_EXTENSION,4,true
```

Instead of the previous one.

Finally delete all files and folders contained in SERVICE_MANAGER_FOLDER \configuration\org.eclipse.osgi and restart your service manager. Spagic will use now your own profile services in the execution of the processes.

4.3.7 Processes design and execution

A workflow process can contain automatic tasks, human ones or both mixed together.

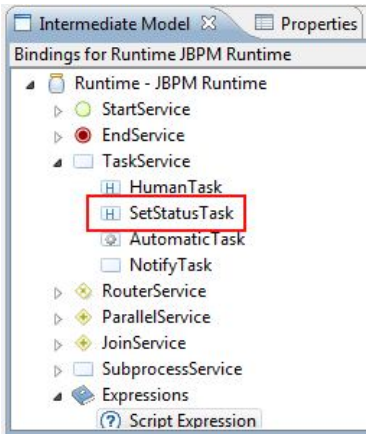
4.4 Processes with automatic task

In order to know how to design a process with automatic task see the chapter 6.1 (Creation of a new process) in “Spagic 3 Getting Started” document.

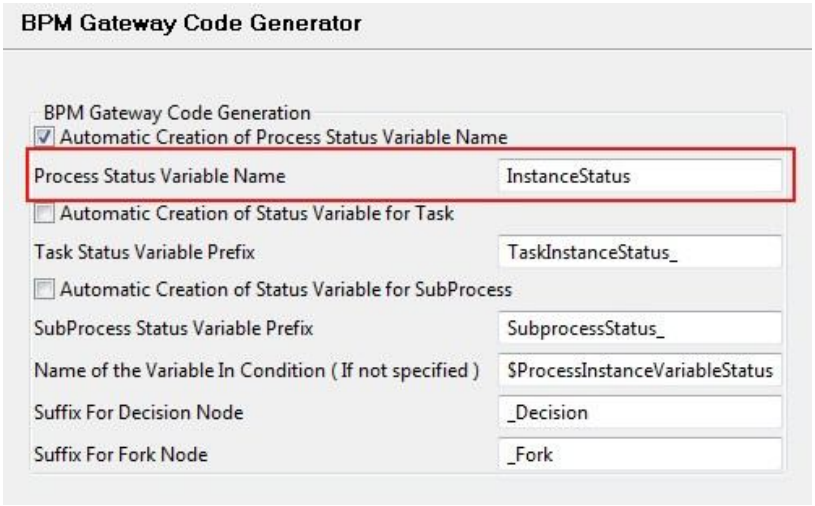
4.4.1 Special Automatic Tasks

In the following paragraph we describe some special kind of automatic tasks that can be used to manage some processes.

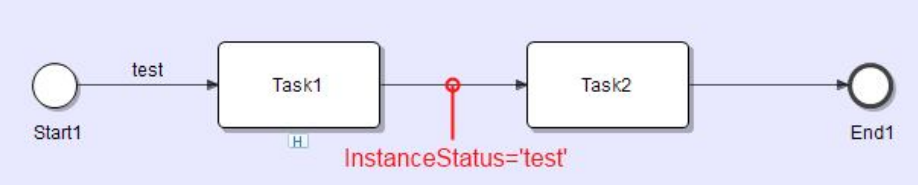
4.4.1.1 SetStatusTask



This task retrieves the label of the first incoming transition and set it as value of the variable <*InstanceStatus*>.
To use this task, the option “Automatic Creation of Process Status Variable” must be enabled.



For example, after starting the following sample process, Task1 (a task of type SetStatusTask) will be executed automatically, and the result will be the variable <*InstanceStatus*> initialized with the value “test”.



4.5 Processes with human task

In order to create a workflow process with human task inside Spagic Studio 3 you have two different ways, the human task can have an associated form or not.

If you don't attach any service to a task of a process, the task will be stated as manual so when you deploy and execute the process it will stops at the first human task; then when this first task will be solved, the process will continue until the next manual task or until the end (if it has no more of them). In order to resume a process blocked in a human task without an associated form the actor can use the provided API as explained in the Spagic Workflow APIs chapter.

In Spagic Studio you can also associate a form definition to a human task. A form definition contains the fields and fields data type that the task needs in order to execute. In order to create a form for a task you first have to configure the fields of this form. To do this, right click on *New* → *Other*. In the new Wizard, under the Spagic folder choose *New Metadata* and click on Next.

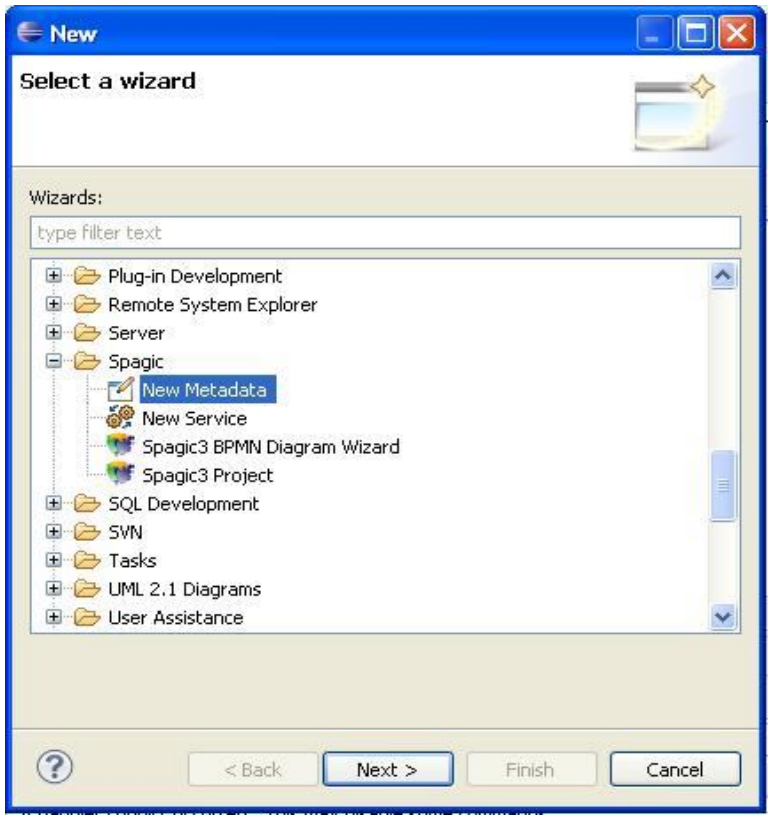


Image 3 – New Metadata.

In the wizard set the placement of the connector under the directory PROJECT_DIR/Metadata. If is not already present, click on the Browse button and set it manually, in the folder selection window. Finally set the name (e.g. metadata_test). of the main "container" of the field that are in the form and click Finish. Then the metadata editor appears to build the form we want to create. In the left side there is the Metadata definition in which you can see the composition of the form while in the right side there is the Metadata definition details in which you can define the features of the fields.
In the example below on the left we have the main "container" of form's fields and on the right you can specify its ID.

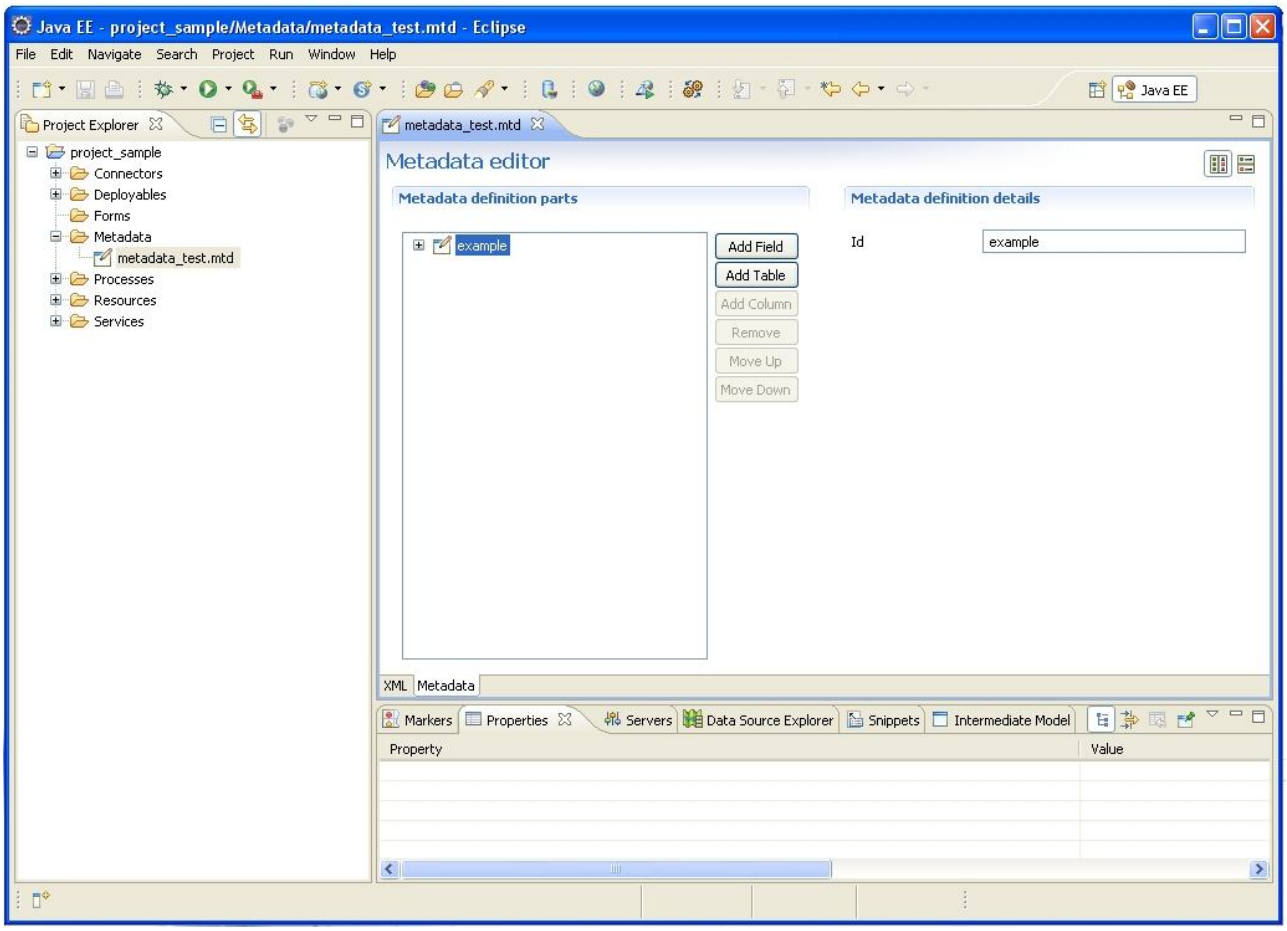


Image 4 – Form Editor.

By clicking the *Add Field* button, you can add a new field to the form and in the details on the right side you can define the id and the type of the field.
Clicking the *Add Table* button you add a new table to the form in which you can add new column, specifying the id and the type of the column. You can also decide if a field is mandatory or not for the execution of the task the form belongs to.
Once created the fields and tables of the form, you can change the order in which the fields will appear in the form using the buttons "Move Up" and "Move Down".

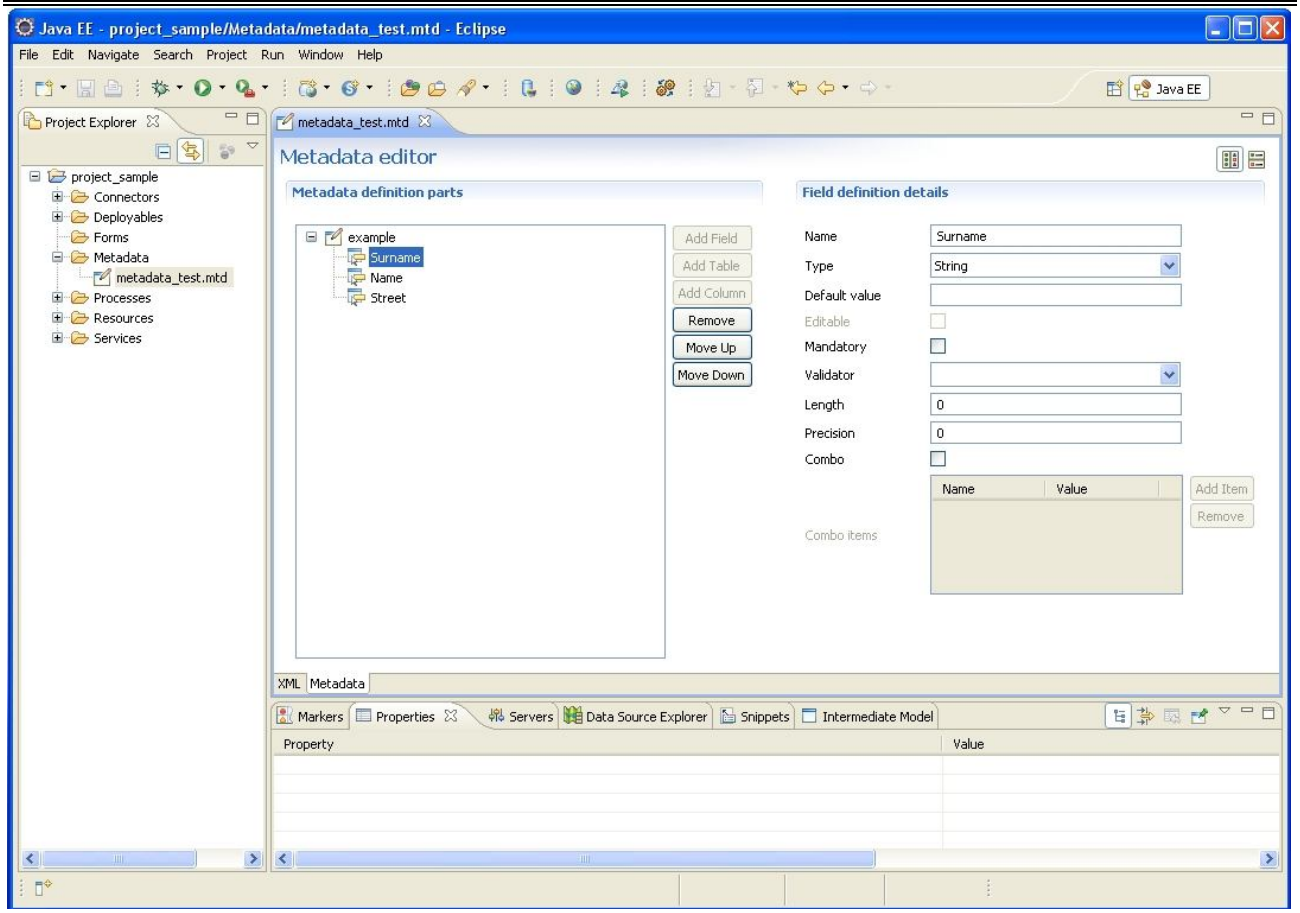


Image 5 – Fields order in Form Editor.

Before to attach the form to the task, we have to configure the process and set the binding of the task to human task.

To do this, we have to select the main swimlane that contains the process and go in the view *Properties*, click on *Technology* and in the combo box select *JBPM Runtime* and click on *Bind to Runtime*.

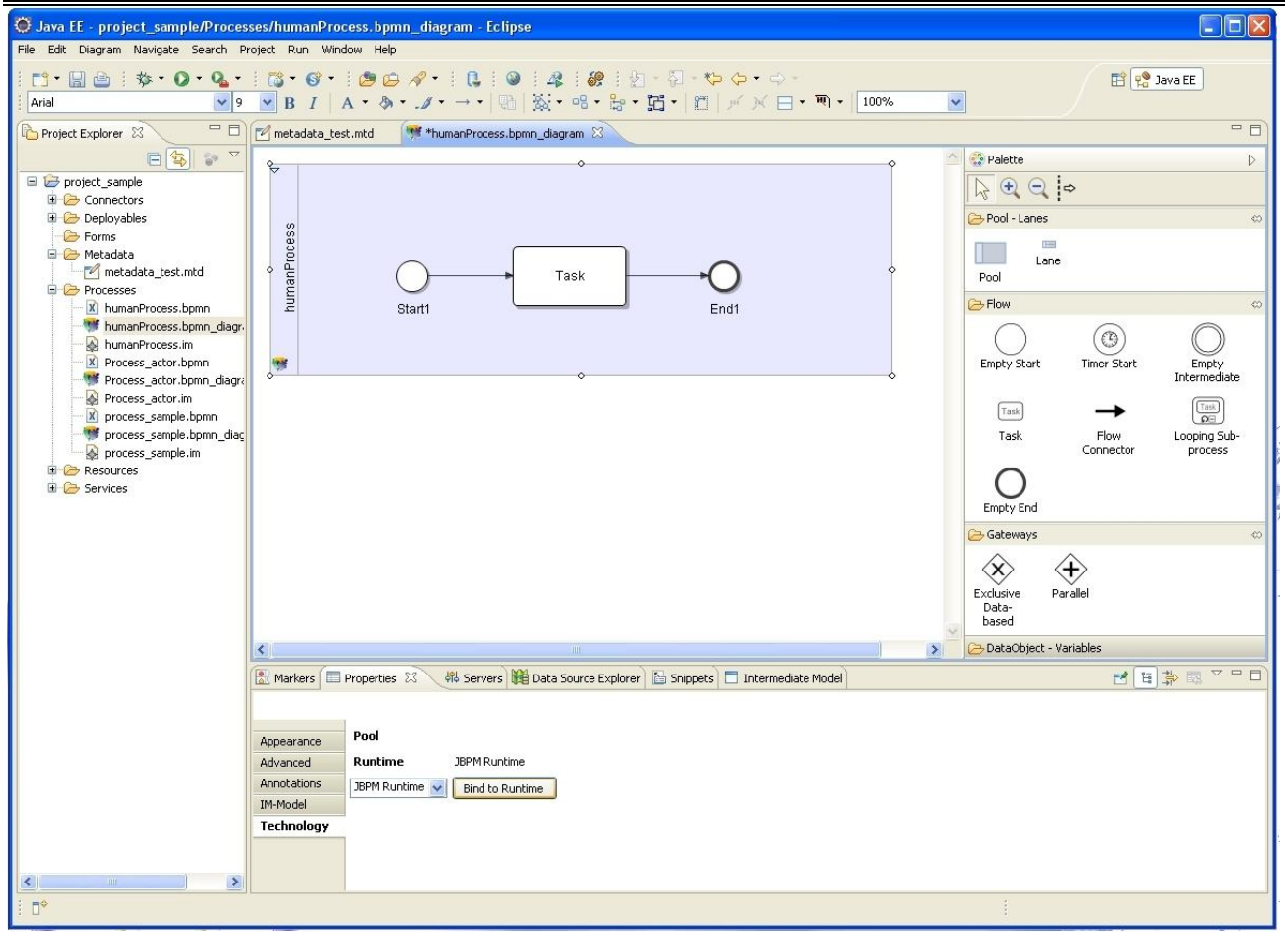


Image 6 – Process Technology.

Then switch to *Intermediate Model* tab. If it's not already open go to *Windows → Show View → Other → Intermediate Model*. Inside the list *Runtime – JBPM Runtime* select the *TaskService* submenu and under it select *HumanTask* service.

In order to tell the task to operate as a Human Task simply drag and drop the HumanTask icon over the task image inside the process and click on "*Attach Service[TaskService] with Binding (HumanTask) on Task".*

Click on Task and set the Property "*Form Type*" to *Dynamic*. Now in order to attach the form to the manual task you have to drag and drop the metadata configuration to the manual task and select "*Fill the property(Form Schema) with this file (metadata_test.mtd) on Property*". In this way the form will be automatically linked to the manual task.

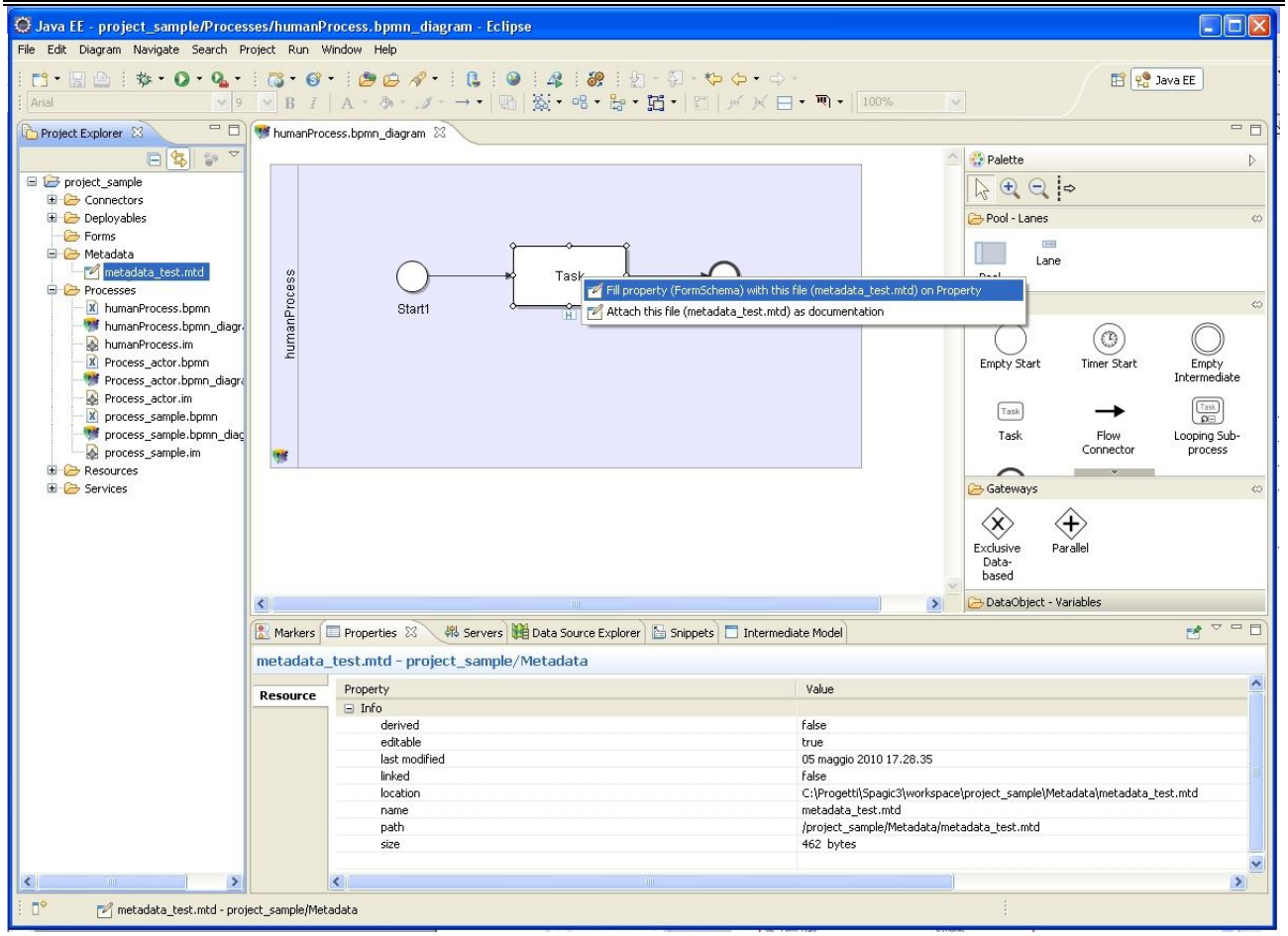


Image 7 – Assign Form to Human Task.

Then, if you click on the human task you have joined the form you'll find under the tab Properties, the path of the metadata file of the form in form schema. After this you can add an input and an output connectors as explained in "Spagic 3 Getting Started" document.

4.5.1 Setting an expiration time to a Human Task

You can also add for each human task a deadline in which the task has to be completed: furthermore you can also associate a service or a connector that will be executed when the time is expired without completing the human task, in this way we can avoid the process to be freezed if a human task is blocked. After having annotated the task as human, as previously described, we can set under the tab properties the expiration time of the task. We can set a date or a relative period of time.

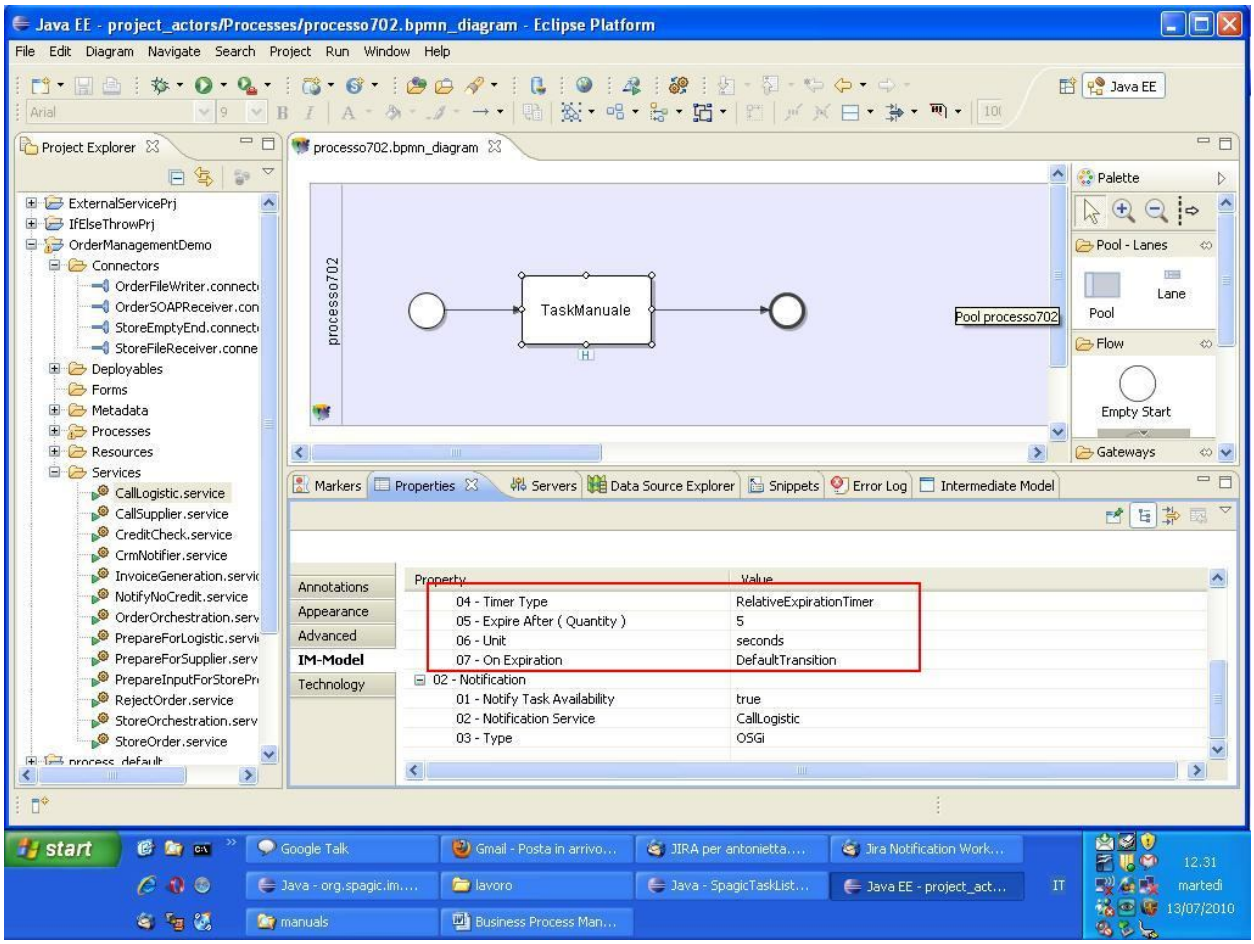


Image 8 - Expiration Property of Human Task

If you want to start a service or a connector if the task expires without being completed, you simply have to drag and drop the service/connector to the human task and click on click on Activation Task Availability to the Human Task

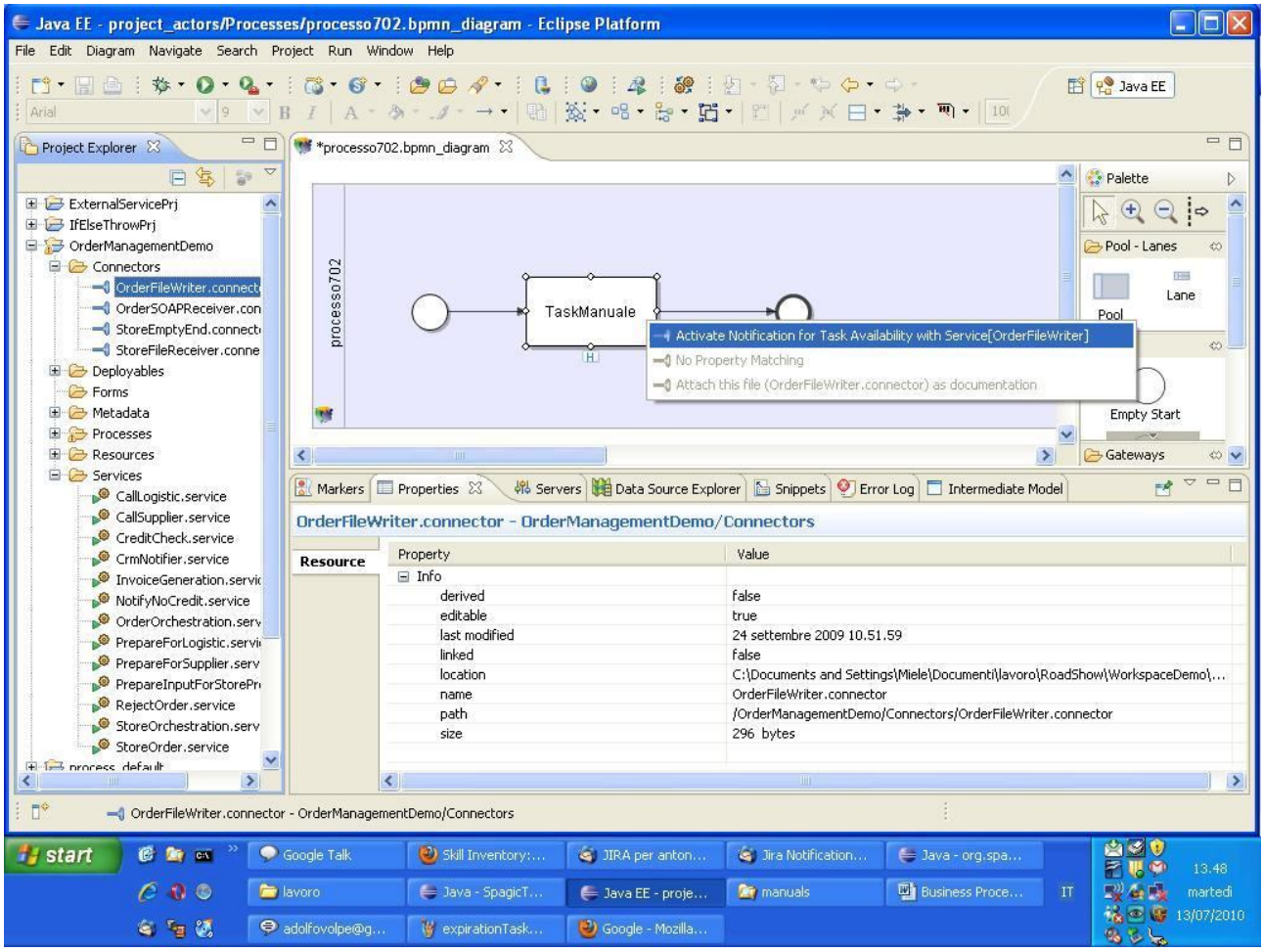


Image 9 - Activate notification with service

4.6 Processes with automatic and manual tasks

To model processes with both automatic and manual task you have to draw the bpmn diagram of the process, and then for the automatic task you have to attach the services you have previously configured to the task in the way we have described before. For the manual tasks that compose the process, if these tasks don't need a form to be executed, simply leave them without any item or property, the deployment service will recognize them as manual tasks, and during the execution the process will stops when it arrives at them. The process could be resumed invoking the specific APIs.

If you want to attach a form to the human tasks you have to follow the procedure described in the previous chapter, configuring the technology, specifying the Task as a JBPM Human Task and specifying its Form Type before attaching a form. In this way we can also model, deploy and execute hybrid processes, with both human and automatic tasks.

4.7 Process execution

Before we can execute the process we must deploy it as explained in “Spagic 3 Getting Started” document.
Once deployed the process, we can start it in two ways, by invoking the dedicated API method, or by a connector.

For automatic processes the execution will continue straight to the end executing all the services provided by the various steps.

For the processes with manual tasks without form the process will block its execution and wait on each manual task. The execution could be resumed using the specific APIs or with the Spagic TaskList web application.

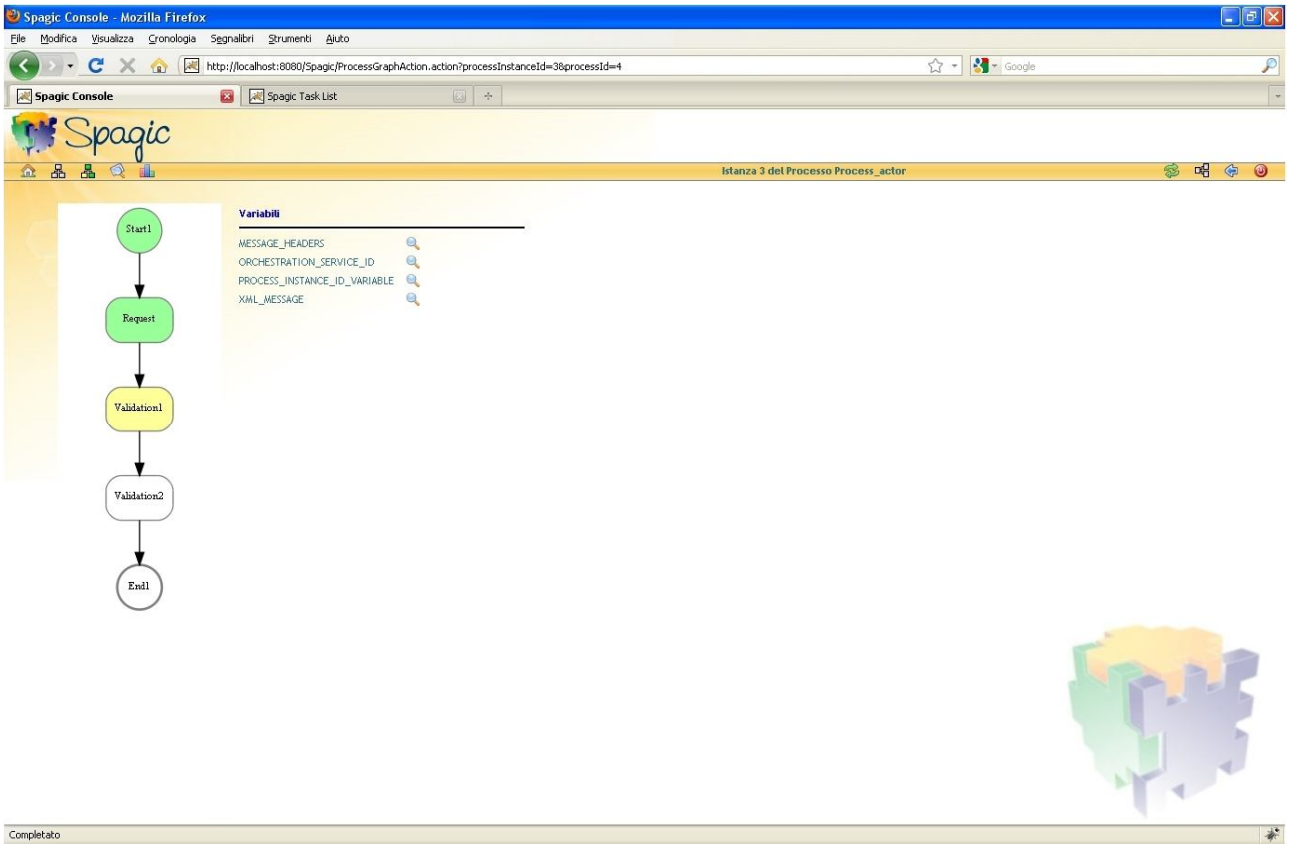


Image 10 – Process Waiting for Human Task.

For the manual task that has a form associated, the procedure is almost the same, in order to continued, the process needs that the mandatory fields of the form are filled and validated (if these field has a validator configured).

5 Spagic Workflow APIs

Spagic provides a series of APIs that allows users to interact with the human task in order to start a process, resume processes stopped for a human task execution, complete the forms associated to a human task and terminate a process execution.
These APIs are composed by three main elements: *ProcessEngine, QueryAPI* and *ControlAPI.*
For method names and interfaces refers to the provided Javadoc and APITester.java class example.

5.1 ProcessEngine

The process engine implementation defines which workflow engine will be used: local or remote engine that encapsulate jBPM.
The process engine is a container of the control and query APIs.

5.2 QueryAPI

QueryAPI is used to set or retrieve process elements like processes, tasks and statuses.

5.3 ControlAPI

ControlAPI is the core API to run processes, execute tasks and obtain all the process variables.

6 Spagic TaskList

In Spagic there is an useful application to interact with processes with human tasks. It consist in a tasklist where there are displayed all the tasks that has been generated and are waiting to be filled and declared solved, according to the role of the user that using the tasklist.

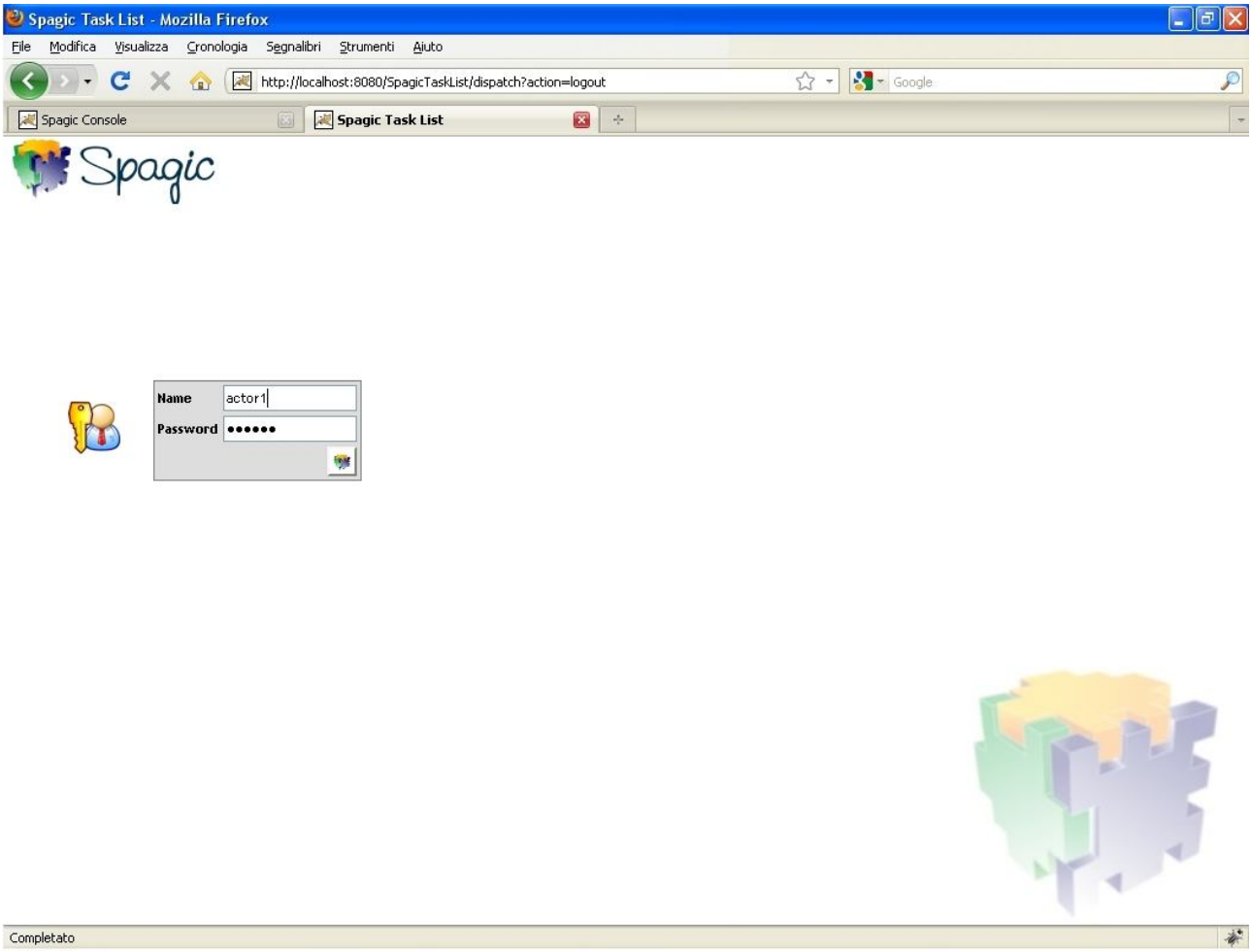


Image 11 – Login into Spagic Task List.

To interact with a task you can click the *Complete* icon under the Actions field, if the task is without the associated form the Complete button will simply resume the process execution while if the task has an associated form the Complete button makes the form of the task to appear. In this case its possible to fill and complete the task, fill it partially and save the fields modified or reject completely the task. When a task is completely filled on all its mandatory fields then it's possible to resume the process execution by clicking on the form *Complete* button.

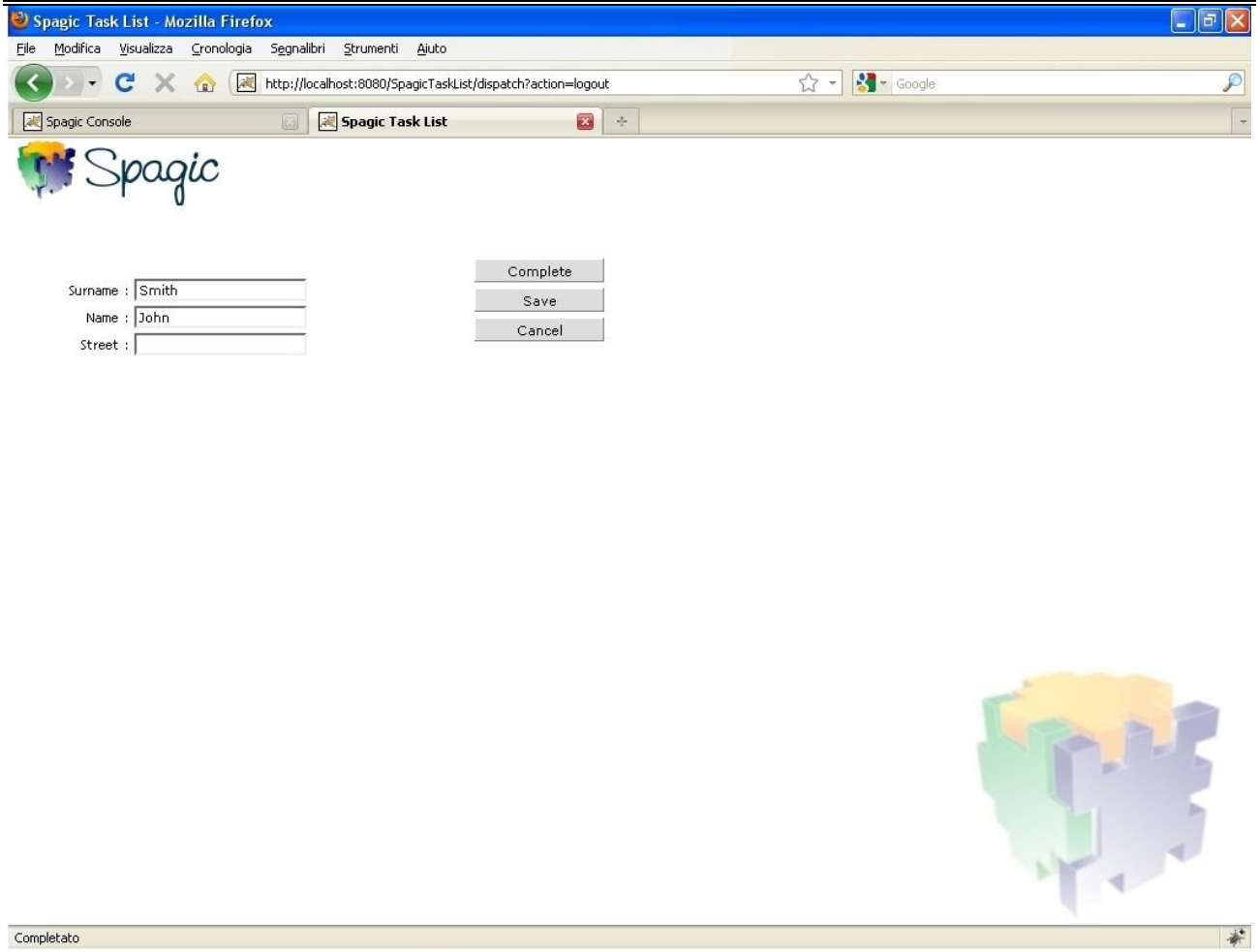


Image 12 – Task List Form filling.

In the Task List there is also available a series of filters in order to select tasks status, date of creation, date of expiration and much more.

As we have previously said there is the possibility of different users to interact with the process execution for the tasks that deals with their role; each user needs to login into the Task List in order to fill the form or simply resume the process execution.

At any time the Task List will show only one waiting task for each process instance or more tasks if there are parallel flows or several process instances in execution. Once executed the current task, the Task List application will automatically refresh.

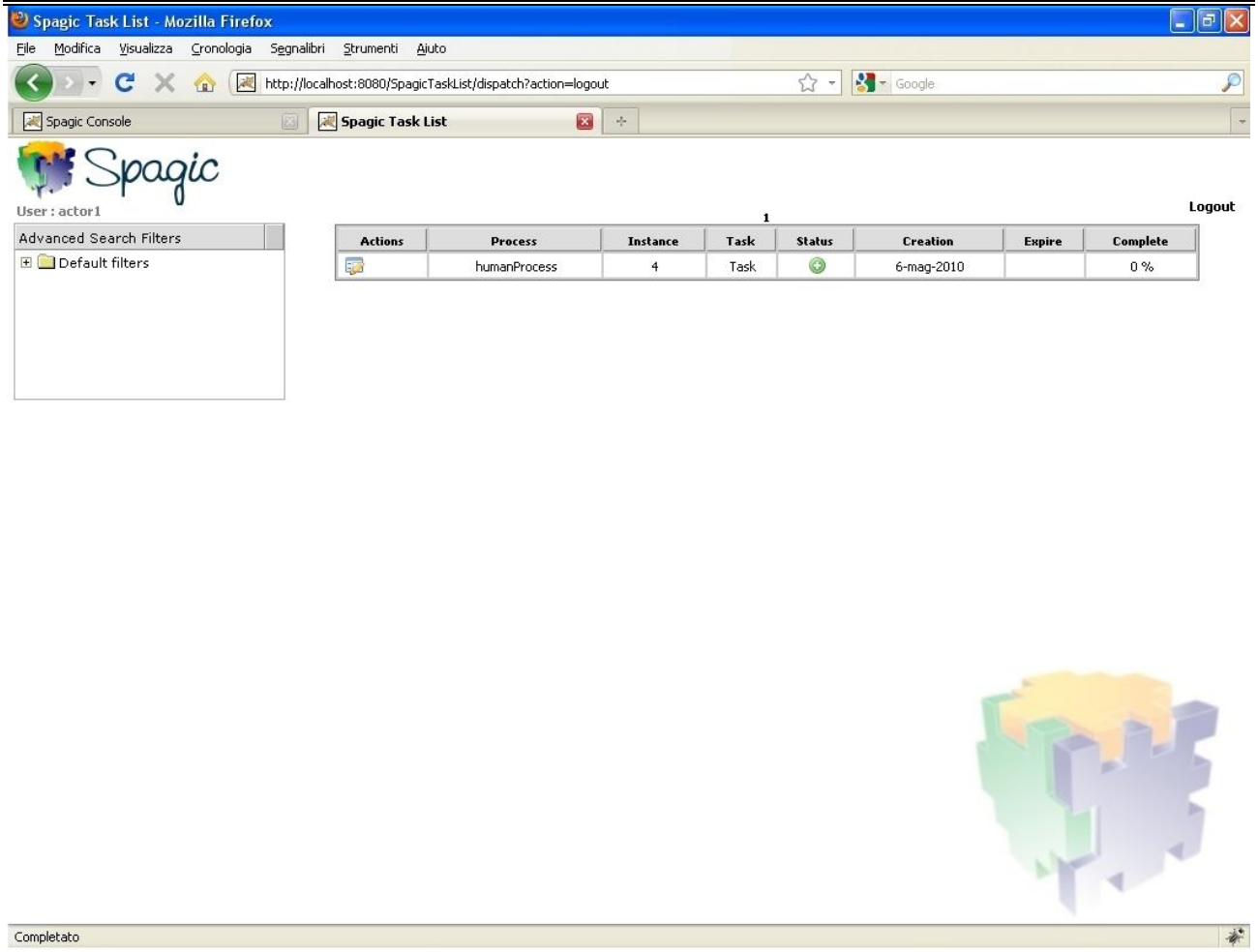


Image 13 – Task List.