WF-system. Business Processes Structure.

Table of Contents

Introduction

The workflow language for Runa WFE is jPdl. jPdl language is workflow language of JBOSS JBPM. This document describes how to use jPdl in Runa WFE. Also this document provides a few working examples of business processes, which show how to develop business processes for Runa WFE.

Runa WFE business process description language restrictions

Reference documentation for jPdl language can be found at http://www.jboss.com/products/jbpm/docs/jPdl.

Present document provides details of using jPdl with Runa WFE.

Process archive structure

Runa WFE business process is defined using process archive. This archive is jar archive (usually with .par extension) containing set of XML documents, form files and Java class files. Optionally the .par file may contain image of business process graph (graph.gif) and detailed process description (description).

The file-archive structure:

- processdefinition.xml
- forms.xml
- graph.gif
- description
- Folder: forms
  - *.form
- Folder: classes
  - *.class

processdefinition.xml file located in the root of process archive contain process graph and swimlane definitions.

forms.xml file in the root of archive defines process variables and list of process forms.

Optional graph.gif file is graphical representation of process graph displayed in properties of deployed process definition.

Optional description file contains detailed process description in html format, displayed in list of deployed process definitions.

All form files defined in forms.xml must be present in process archive. It is highly recommended to put all form files into “forms” directory in the root of archive.

If business process contains auxiliary Java classes they must be packaged in “classes” directory in the root of archive. These classes will be loaded by system core during process deployment.

Description of processdefinition.xml

Swimlanes

Runa WFE uses swimlane initializes to determine executors who will execute a process activity. Every activity has swimlane associated with it. Swimlane determines who can execute this activity. This decision is made by mean of swimlane assignment handler. Assignment handler is Java class that returns id of user who can execute activity.

If user queries for list of available activities (via tasklist) the mechanism called orgfunction is used to decide whether user can execute specific task or not. Orgfunction is a Java class that implements logic of organization function. This class generates list of executors that can execute activity. If user is in this list, activity is shown as available in tasklist. Orgfunction class must implement ru.runa.af.organizationfunction.OrganizationFunction interface.

After competition of activity corresponding swimlane is initialized. Swimlane initialization is initialization of special process variable (with same name as name of swimlane) with user id.

Runa WFE uses the following algorithm determine whether activity can be completed by user or not:

- If the swimlane is not initialized, the activity is displayed in tasklists for all users from orgfunction output. When the first user executes the activity, swimlane is initialized for this user id.
- If the swimlane is initialized with user id, the activity is displayed in tasklist for this user only.

There is another way to initialize swimlane. User id can be assigned to process variable with name of swimlane. In this case orgfunction is not needed and swimlane is initialized after variable setting.

To define swimlane it is required to define following elements:

- Swimlane assignment handler, which returns user id who can execute activity. Runa WFE provides implementation of assignment handler (ru.runa.wf.jbpm.delegation.assignment.AssignmentHandler) returning user id for initialized swimlanes and empty string for not initialized swimlanes.
- Orgfunction, which returns list of user ids who can execute activity. Orgfunction must be defined using jPdl assignment delegation configuration with following format: <Initializer Java class>(<parameter>, <parameter>, …)

Runa WFE distribution contains following demo implementations of orgfunction that can be used as reference:

- ru.runa.af.organizationfunction.ExecutorByNameFunction. Parameter is name of executor. The initializator returns user or group of users.
- ru.runa.af.organizationfunction.DemoChiefFunction. Parameter is user id. The initializator returns the chief of this user (demo specific class)

Parameters can be either string constants or process variable value. Variable values must be enclosed in special braces -- ${} (e.g. ${variable name}). In this case during execution orgfunction class receives corresponding variable value.

Swimlane substitution

In addition to orgfunction swimlane assignment there is a mechanism to complete tasks as another user. This mechanism is called substitution and user that performs another user task is called substitute . Substitution allows to define which users can execute task as substitute. To define substitution it is required to define two additional elements (in addition to assignment

handler and orgfunction) in the swimlane initializer:

- Substitution criteria – specifies if swimlane substitution is required.
- Substitution function – an orgfunction which returns list of substitute ids if substitution is required.

Substitution criteria is Java class implementing ru.runa.af.organizationfunction.SubstitutionCriteria interface. Substitution criteria accepts the set of parameters similar to orgfunction parameters.
Substitution function is a regular orgfunction i.e. Java class implementing ru.runa.af.organizationfunction.OrganizationFunction interface.
In addition to regular set of parameters (text constants and variable values) both substitution criteria and substitution function accept special parameter which is denoted with '?' symbol. This parameter represents substituted user and calculated at runtime for each swimlane. This parameter can be passed to substitution criteria or substitution function if substituted user is required for criteria or function implementation.

In swimlane initializer orgfunction, substitution criteria, substitution function must be defined in following order: <orgfunction>[;<substitution function>:<substitution criteria>]. Substitution criteria and substitution function can be omitted if no substitution required for swimlane.

Runa WFE distribution contains demo implementation of substitution criteria that can be used as reference:

- ru.runa.af.organizationfunction.AlwaysTrueSubstitutionCriteria, has no parameters and always require substitution for any assigned user.

Description of forms.xml

JBOSS JBPM 2.0 does not define the forms.xml structure. Runa WFE requires forms.xml. The forms.xml consists of single tag <forms>. This tag contains a set of <form>-tags. Every <form>-tag corresponds to activity with a graphical form, or to activity in which variables are initialized.

Tag <form> has three mandatory attributes:

- state – activity name
- file – name of the file, which contains the form.
- type – form type (the only supported type for now is “html”)

If variable is assigned in the activity, the activity must contain variable tag with the following attributes:

- name – variable name (mandatory)
- format – the name of parsing class (optional, default format use variable value as is)
- optional – the variable optionality (by default variable is mandatory)

The complete version of XML Schema for forms.xml can be found in the resource directory of Runa WFE distribution.

Description of *.form files

Every .form file contains the activity form description. The reference form parsing mechanism uses HTML with additional tags <customtag>. These tags are used to display process variable value in the form.

The < customtag> tag have the following attributes:

- var – process variable name
- delegation – name of Java class, responsible for the variable value rendering (must

implements ru.runa.wf.web.html.VarTag interface)

Examples of process archives development.

HelloWorld process.

The process scenario:

- At start HelloWorld form appears.
- When the button “complete” is pressed the process ends.

This process has two nodes:

- Start-state
- Stop-state

The content of processdefinition.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE process-definition PUBLIC  "-//jBpm/jBpm Mapping DTD 2.0//EN"     "http://jbpm.org/dtd/
processdefinition-2.0.dtd">
<!-- process-definition tag beginning  -->
<process-definition name="Hello World">
    <!--  Swimlane definition -->
  <swimlane name="requester" />

    <!-- Process start point -->
  <start-state name="Hello World state" swimlane="requester">
    <!-- Transition to the next state -->
       <transition to="done"/>
  </start-state>
    <!--  The state, where process ends -->
  <end-state name="done" />

<!-- process-definition tag end -->
</process-definition>
```

The content of forms.xml file:

```
<?xml version="1.0"?>
<forms xmlns="http://runa.ru/xml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://runa.ru/xml forms.xsd">

  <!--  This tag links state with form file -->
  <form state="Hello World state"  file="forms/HelloWorld.form" type="html">
      <!--  No variables are initialized -->
  </form>
</forms>
```

The graph.gif file may contain the following picture:

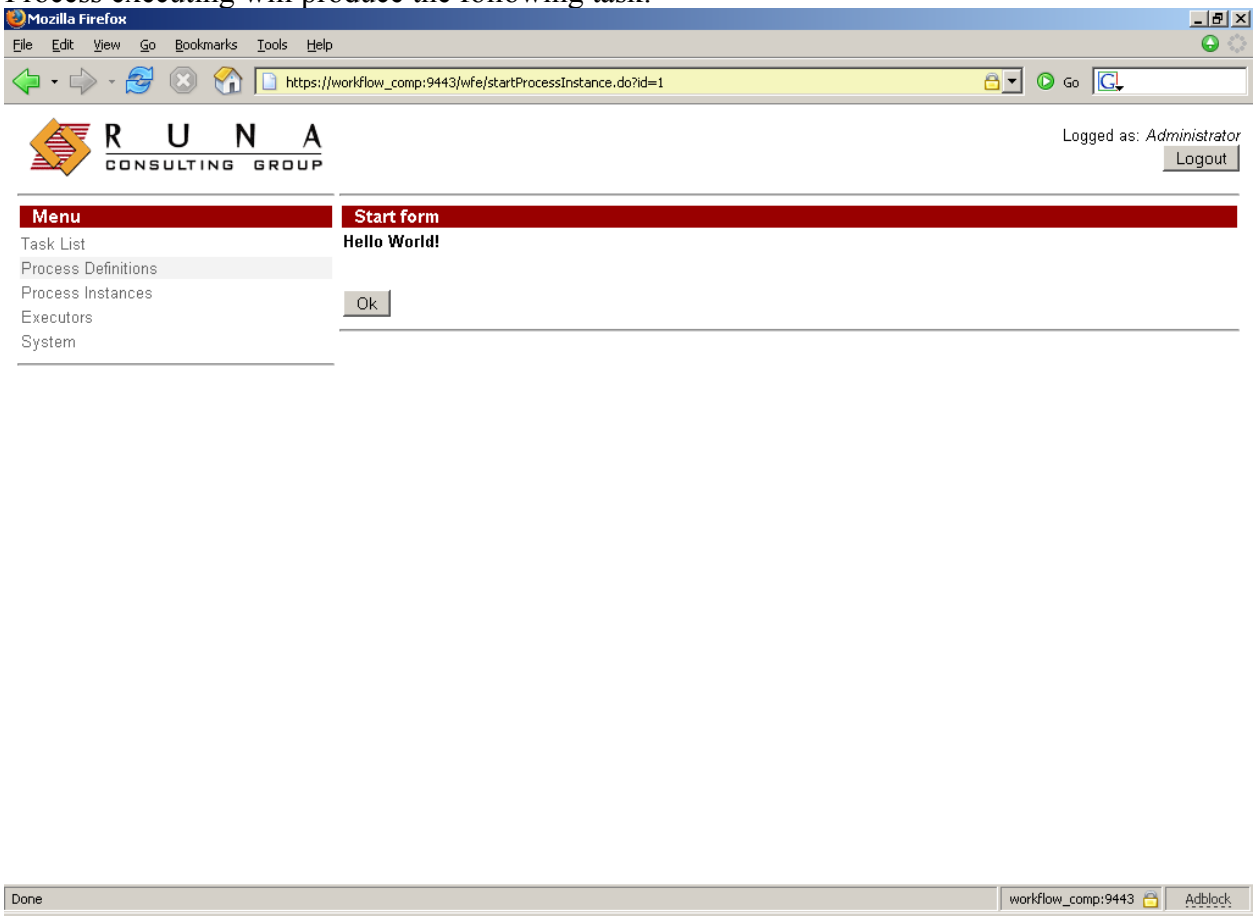
The HelloWorld.form file content:

```
<b>Hello World!</b> <br> <br> <br>
```

The structure of HelloWorld.par is:

File HelloWorld.par
- processdefinition.xml
- forms.xml
- graph.gif
- Folder forms
    - HelloWorld.form

Now process archive HelloWorld.par can be deployed into Runa WFE.
Process executing will produce the following task:



OverTime process.

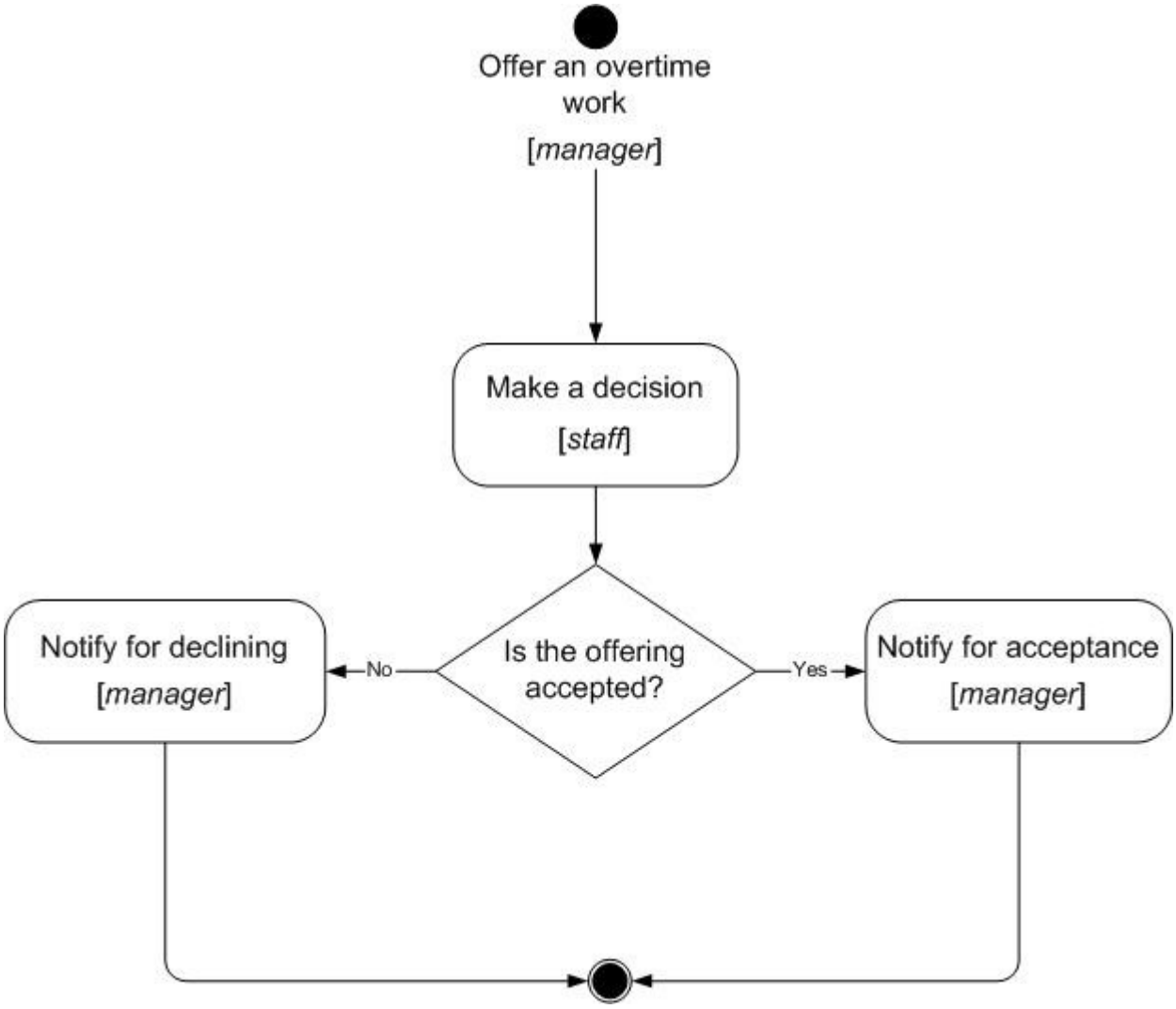
Short description:

Manager offers an over time work to employee. Employee accepts or rejects the offer. Then manager receives the corresponding notification.

All managers are members of group “manager” and all employees are members of group “staff”.

Process designing

Step 1. Process graph designing



Step 2. Variables setting up.

Establish the following variables for the OverTime process:

Variable	**Description**	**Type**	**Activity, where initialization take place**
staff	Employee's ID	ID	Offer an overtime work
since	Date-time since…	Date-time	Offer an overtime work
till	Date-time till…	Date-time	Offer an overtime work
reason	OverTime reason	String	Offer an overtime work
comment	comment	Text	Offer an overtime work
staff person decision	employee decision	Boolean	Make a decision
staff person comment	employee comment	Text	Make a decision

Step 3. Task executors setting up

Establish following swimlanes:

- staff person - employee
- manager – manager (chief of “staff person”)

Swimlanes initialization:

Swimlane	Who initialize swimlane
manager	Person, who starts process. (It is supposed, that the only members of group “manager” have rights to start this process.)
staff	Members of group “staff”.

Swimlane – activity mapping:

Activity	Swimlane
Offer an overtime work	Manager
Make a decision	Staff
Notify for declining	Manager
Notify for acceptance	Manager

Step 4. Graphical forms designing.

Swimlane – forms mapping:

Activity	Form containing file
Offer an overtime work	OfferAnOvertimeWork.form
Make a decision	MakeaDecision.form
Notify for declining	NotifyForDeclining.form
Notify for acceptance	NotifyForAcceptance.form

The type of every HTML element in a form is determined by the type of process variable in all cases except variable staff. The staff variable corresponds to choice, containing all members of staff group names.

Process archive development

File processdefinition.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE process-definition PUBLIC
    "-//jBpm/jBpm Mapping DTD 2.0//EN"
    "http://jbpm.org/dtd/processdefinition-2.0.dtd">

<process-definition name="over time work demo">
  <description> Over time work </description>
  <!-- Swimlanes definitions -->
  <! -- swimlane “manager”  definition, (initialization takes place in the start point) -->
  <swimlane name="manager" />

  <! -- swimlane “staff”  definition, (initialization takes place in the start point) -->
  <swimlane name="staff"/>
```

```
  <!--  Start point -->
  <!-- Swimlane manager is initialized by user starting the process -->
  <!-- Swimlane staff is initialized with the help of graphical form -->
  <start-state name="Offer an overtime work" swimlane="manager">
     <transition to="Make a decision"/>
  </start-state>

  <!-- Nodes -->

  <!-- Activity -->
  <state name="Make a decision">
     <description>Here employee accept or decline over time offer </description>
     <!-- Associated with activity swimlane is staff -->
     <assignment swimlane="staff" assignment="required" />
     <transition to="Is the offering accepted"/>
  </state>

  <!-- Exclusive choice -->
  <decision name="Is the offering accepted">
  <!-- The delegation mechanism is used. The transition is chosed with the
                                                                                                    help of
BeanShell script -->
    <delegation class="ru.runa.wf.jbpm.delegation.decision.BSFDecisionHandler">
               <![CDATA[
               if(Boolean.valueOf(staffPersonDecision).booleanValue())
                       return "accept";
               else
                       return "decline";
               ]]>
    </delegation>
               <transition name="accept"     to="Notify for acceptance"/>
               <transition name="decline"  to="Notify for declining"/>
  </decision>

  <!-- Activity -->
  <state name="Notify for acceptance">
    <description>the task is - to get acquainted with acceptance of
over time work</description>
        <assignment swimlane="manager" assignment="required" />
        <transition to="done" />
  </state>

  <state name="Notify for declining">
<description>the task is - to get acquainted with rejection of
over time work</description>
<assignment swimlane="manager" assignment="required" />
        <transition to="done" />
  </state>

  <!-- The end point of the process -->
  <end-state name="done" />

</process-definition>
```

File forms.xml

```
<?xml version="1.0"?>
<forms xmlns="http://runa.ru/xml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://runa.ru/xml forms.xsd">

  <!-- Here activity is linked with form, initialized in this activity variables are established,
the classes for variables parsing is setting up -->
  <form state="Offer an overtime work"  file="forms/OfferAnOvertimeWork.form"
type="html">
      <variable name="staff" />
      <variable name="since" format="ru.runa.wf.web.forms.format.DateTimeFormat" />
      <variable name="till" format="ru.runa.wf.web.forms.format.DateTimeFormat" />
      <variable name="reason" />
      <variable name="comment" />
  </form>

  <!-- Here activity is linked with form, initialized in this activity variables are established,
                                                                                the classes for
variables parsing is setting up -->
  <form state="Make a decision"  file="forms/MakeaDecision.form" type="html" >
       <variable name="staffPersonDecision"
                                                                format="ru.runa.wf.web.forms.form
```

```
at.BooleanFormat"/>
        <variable name="staff person comment" />
  </form>

  <!-- Here activity is linked with form. Variables in this form are not assigning -->
  <form state="Notify for declining" file="forms/NotifyForDeclining.form" type="html" >
  </form>

  <!-- Here activity is linked with form. Variables in this form are not assigning -->
  <form state="Notify for acceptance" file="forms/NotifyForAcceptance.form" type="html" >
  </form>

</forms>
```

File graph.gif

graph.gif is the picture from the section “Step 1. Process graph designing”

Files *.form

File OfferAnOvertimeWork.form:

```
<table cellspacing="0">
  <tr>
    <td valign="top">

      <table cellspacing="0" bgcolor="#eeeeee" style="border-style:solid;
                                                                    border-width:1px;border-
color:black;">

       <tr>
          <th colspan="2">
            <h3>Offer an overtime work</h3>
            <hr>
          </td>
        </tr>

       <tr title="staff">
           <td align="right">
           Employee:
           </td>
           <td>
 <!-- Special tag, it corresponds to choice, containing the list of group which name is defined
with the help of “var=…” construction . The tag  returns ID of chosen group member. Tag uses
delegation mechanizm -->
       <customtag var="staff" delegation =
"ru.runa.wf.web.html.vartag.GroupMembersComboboxVarTag" />
           </td>
       </tr>
       <tr title="since">
          <td align="right">
            DateTime since (dd.mm.yyyy hh:mm):
          </td>
          <td>
            <!--Special tag for working with dates-->
            <customtag var="since" delegation="ru.runa.wf.web.html.vartag.DateTimeInputVarTag" />
          </td>
        </tr>

       <tr title="till">
          <td align="right">
            DateTime till (dd.mm.yyyy hh:mm):
          </td>
          <td>
            <customtag var="till" delegation="ru.runa.wf.web.html.vartag.DateTimeInputVarTag" />
          </td>
        </tr>

        <tr title="reason">
          <td align="right">
            Reason :
          </td>
          <td>
            <INPUT TYPE="text" NAME="reason">
          </td>
```

```
        </tr>

        <tr title="comment">
          <td align="right">
            Comment :
          </td>
          <td>
            <textarea name="comment"> </textarea>
          </td>
        </tr>
</table>
```

Content of files

- MakeaDecision.form
- NotifyForAcceptance.form
- NotifyForDeclining.form

is similar to the content of OfferAnOvertimeWork.form

Process archive structure

File overTimeDemo.par

- processdefinition.xml
- forms.xml
- graph.gif
- Folder forms
  - OfferAnOvertimeWork.form
  - MakeaDecision.form
  - NotifyForAcceptance.form
  - NotifyForDeclining.form

After archiving it is possible to deploy process into Runa WFE.

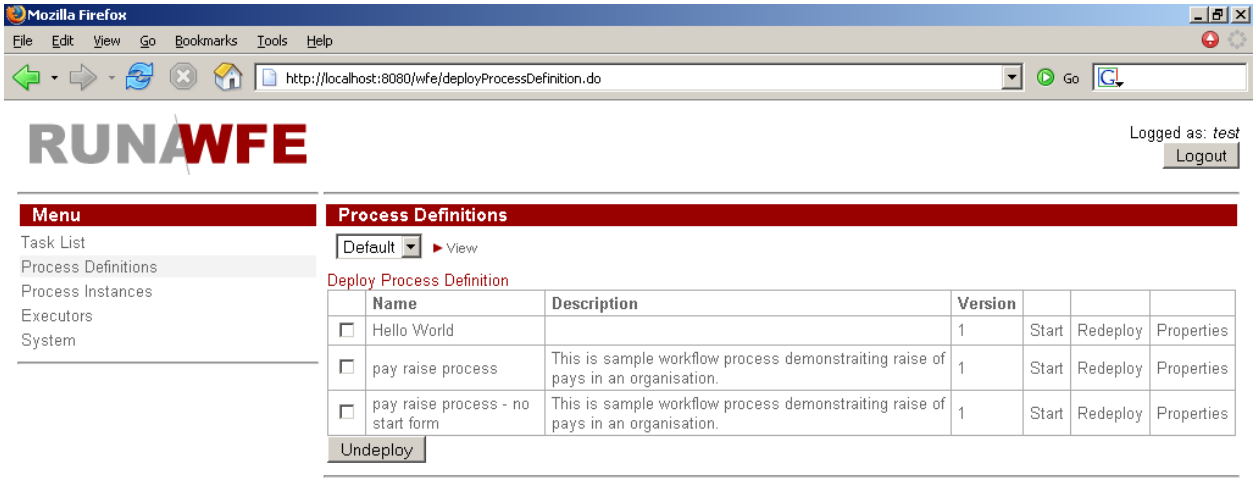
You can find two more complex processes in the Runa WFE distributive:

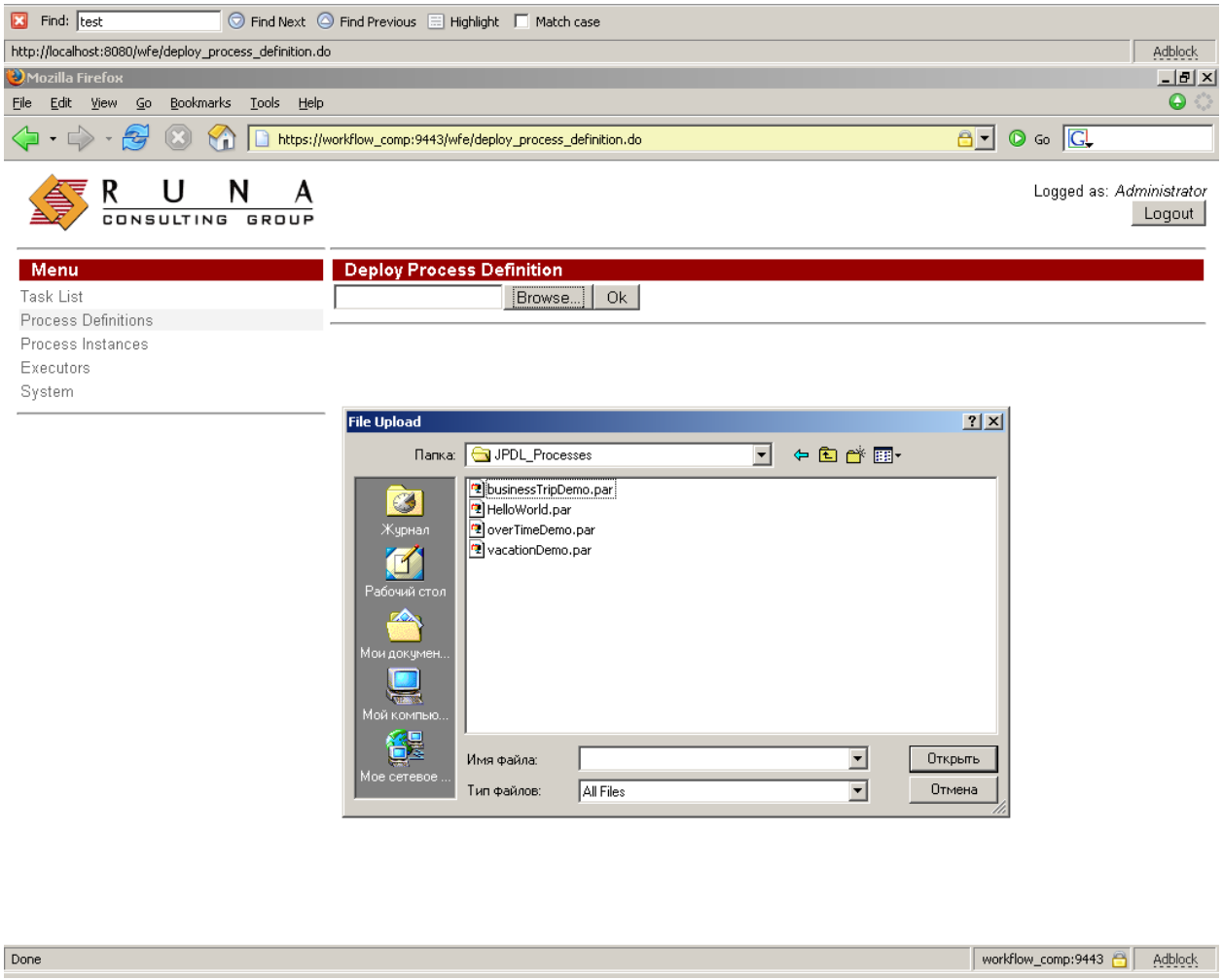
- VacationDemo.par – vacation
- BusinessTripDemo – business trip

Runa WFE. Deployment.

Go to process menu.
Press deploy process definition[1].
Press browse button to select appropriate process definition archive.
Press Ok button.
After deployment process appears in the process list.

1*Note:In order to deploy a process you must have Process Definition permission on System (can be granted via system menu).*


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:8080/wfe/deployProcessDefinition.do
Go
RUNAWFE
Logged as: test
Logout
Menu
Task List
Process Definitions
Process Instances
Executors
System
Process Definitions
Default
View
Deploy Process Definition
Name
Description
Version
Hello World
1
Start
Redeploy
Properties
pay raise process
This is sample workflow process demonstraiting raise of pays in an organisation.
pay raise process - no start form
Undeploy


Find: test
Find Next
Find Previous
Highlight
Match case
http://localhost:8080/wfe/deploy_process_definition.do
Adblock
Mozilla Firefox
File Edit View Go Bookmarks Tools Help
https://workflow_comp:9443/wfe/deploy_process_definition.do
Go
RUNA CONSULTING GROUP
Logged as: Administrator
Logout
Menu
Task List
Process Definitions
Process Instances
Executors
System
Deploy Process Definition
Browse...
Ok
File Upload
Папка:
JPDL_Processes
businessTripDemo.par
HelloWorld.par
overTimeDemo.par
vacationDemo.par
Журнал
Рабочий стол
Мои докумен...
Мой компью...
Мое сетевое ...
Имя файла:
Тип файлов:
All Files
Открыть
Отмена
Done
workflow_comp:9443
Adblock