

RUNA WFE. Структура файла-архива бизнес-процессов.

Версия 2.1

© 2004-2008, ЗАО “Руна”. RUNA WFE является системой с открытым кодом и распространяется в соответствии с LGPL лицензией (<http://www.gnu.org/licenses/lgpl.html>).

Оглавление

Введение.....	2
Бизнес-процессы в системе Runa WFE.....	2
Что такое бизнес-процесс. Неформальное описание.....	2
Формальное определение бизнес-процесса.....	3
Перспектива Управления Потокoм.....	3
Перспектива Данных.....	4
Перспектива Ресурсов.....	4
Перспектива Операций.....	4
Структура файла-архива бизнес-процесса.....	5
Версии бизнес-процессов.....	5
Описание элементов языка определения бизнес-процессов, используемого в системе Runa WFE.....	5
Описание файла processdefinition.xml и используемых в нем тегов.....	5
Описание файла processdefinition.xml.....	5
Использование ролей (элемента swimlane) в системе Runa WFE.....	6
Описание структуры файла.....	7
Описание тега process-definition.....	8
Описание элемента swimlane.....	9
Описание элемента start-state.....	9
Описание элемента state.....	9
Описание вложенного элемента assignment.....	10
Вложенный элемент action.....	10
Вложенный элемент transition.....	10
Описание элемента milestone.....	10
Описание элемента process-state.....	10
Описание элемента decision.....	10
Описание элемента fork.....	11
Описание элемента join.....	11
Описание элемента end-state.....	12
Описание элемента transition.....	12
Описание элемента action.....	12
Описание элемента delegation.....	12
Описание файла forms.xml.....	13
Описание файла определения форм (файлы без расширения).....	13
Примеры построения файлов-определений бизнес-процессов.....	13
HelloWorld процесс.....	14
Процесс сверхурочные.....	15
Перспектива управления потоком.....	16

Перспектива данных.....	16
Перспектива Ресурсов.....	17
Перспектива операций.....	17
Файл-архив бизнес-процесса.....	17
Файл processdefinition.xml	17
Файл forms.xml.....	19
Файл variables.xml.....	20
Файл graph.gif.....	20
Файлы форм.....	20
Файлы валидаций (проверок) для переменных форм.....	21
Структура архива.....	22

Введение

RUNA WFE - открытая, масштабируемая, ориентированная на конечного пользователя система управления бизнес-процессами для средних и крупных предприятий, основанная на популярном workflow ядре JBOSS- jBPM.

Бизнес-процессы в системе Runa WFE

Язык определения бизнес-процессов в системе Runa WFE основан на языке jPDL, (проект JBOSS jBPM). В языке определения бизнес-процессов системы Runa WFE есть несколько расширений jPDL. Некоторые возможности jPDL (например, подгрузка классов через файл-архив бизнес-процесса) пока системой RUNA WFE не поддерживаются. Описание языка jPDL можно посмотреть по следующему URL:

<http://www.jbpm.org/jpdl.html>.

В настоящем документе описывается язык определения бизнес-процессов системы Runa WFE.

Команда export редактора бизнес-процессов помещает описание бизнес-процесса в jar-архив с расширением .rag.

Замечание. Файл должен быть jar-файлом без компрессии. Файл можно создать, например, командой «jar cvf0 <имя архива>.jar .».

Описание бизнес-процесса, помещенное в файл-архив, представляет собой набор XML-файлов и файлов, описывающих использующиеся в бизнес-процессе формы. Также архив может содержать графические файлы – графическое изображение графа бизнес-процесса и иконку бизнес-процесса.

При помощи интерфейса системы Runa WFE можно загрузить разработанный бизнес-процесс в систему.

После того, как бизнес-процесс загружен в систему, он появляется в списке бизнес-процессов, на него можно давать права и запускать на выполнение.

Что такое бизнес-процесс. Неформальное описание.

В workflow системе деятельность предприятия представляется в виде множества бизнес-процессов. **Бизнес-процесс – это упорядоченный по времени набор заданий, выполняемых как людьми, так и информационными системами предприятия, направленный на достижение заранее известной бизнес цели за известное время.**

Бизнес-процесс можно представить в виде математического графа – набора

вершин, называемых узлами-Действиями и Маршрутными узлами, соединенных между собой возможными переходами. По этим переходам перемещаются точки управления. При переходе точки управления в конкретный узел-Действие соответствующему исполнителю направляется задание.

Задание может быть назначено сотруднику предприятия, группе сотрудников или боту (Бот – это специальное приложение в информационной системе предприятия)

Если исполнитель не является ботом, то соответствующее задание появится в списке заданий исполнителя. При клике на задание будет отображена соответствующая заданию форма. После реального выполнения задания сотрудник должен заполнить поля формы, предназначенные для ввода данных кликнуть на командной кнопке «Выполнить»: После того, как задание выполнено, точка управления переместится в следующий узел-Действие (узлы-Действия) бизнес-процесса. Если задание соответствует группе пользователей, то это задание появится в списках заданий всех членов данной группы. Однако выполнить задание сможет только один пользователь – тот, который сделает это первым.

Формальное определение бизнес-процесса.

Бизнес-процесс формально определяется при помощи задания следующих Перспектив (точек зрения или слоев/уровней рассмотрения):

- ✓ Перспектива Управления Потокком (control-flow perspective)
- ✓ Перспектива Данных (data perspective)
- ✓ Перспектива Ресурсов (resource perspective)
- ✓ Перспектива Операций (operational perspective)

Рассмотрим подробно все уровни определения бизнес-процесса.

Перспектива Управления Потокком

Перспективу Управления Потокком можно определить как математическое понятие - направленный граф: множество узлов, соединенных между собой дугами (возможными Переходами). Узлы бизнес-процесса могут быть двух типов: Узлы, соответствующие Шагам процесса (назовем их Действиями) и Маршрутные узлы. По Переходам перемещается Точка Управления (указатель на активный узел процесса), руководствуясь правилами в Маршрутных Узлах.

В узле, соответствующем Шагу процесса («на каждом Шаге процесса») WF-система дает задание Исполнителю (пользователю или группе пользователей) и ждет ответа (сообщения, что работа выполнена). После ответа Исполнителя Точка Управления движется по Переходу к следующему узлу процесса. К узлу, соответствующему Шагу процесса (Действию) может примыкать один или более входящих и только один исходящий переход.

Маршрутный узел соответствует разветвлению-слиянию Точек Управления. В этих узлах WF-система, на основании содержащихся в Маршрутных Узлах правил, выбирает следующий узел (узлы), в который будет передано управление. Соответственно с этими узлами связано обязательно более одного входящего или исходящего Перехода.

В выполняющемся бизнес-процессе одновременно может быть несколько Точек Управления. В соответствии с бизнес логикой процесса Точка Управления может разделиться на несколько Точек Управления в Маршрутном Узле процесса, также Точки Управления могут ждать друг друга в другом Маршрутном Узле и сливаться в одну Точку.

Перспектива Данных

Перспектива Данных бизнес-процесса соответствует набору переменных бизнес-процесса. Переменные бизнес-процесса могут являться входящими и исходящими параметрами при взаимодействии WF-системы с информационными системами предприятия.

При помощи переменных бизнес-процесса происходит обмен информацией между узлами-Действиями процесса и, как следствие, между внешними информационными системами, т.е. бизнес-процесс может переносить информацию в корпоративной информационной среде между разнородными информационными системами.

Переменные бизнес-процесса также используются в узлах-разветвлениях при выборе конкретного внутреннего перемещения точки управления по какому-либо из возможных Переходов. Выбор Перехода происходит на основании правил бизнес логики процесса описанной в Перспективе Управления Потокom.

Перспектива Ресурсов

Перспективе Ресурсов бизнес-процесса соответствует список Исполнителей, которые могут выполнить Шаги бизнес-процесса. При этом под Исполнителями мы понимаем как сотрудников предприятия, так и информационные системы или специализированные устройства. Это Перспектива плотно связана с организационной моделью и моделью информационных систем предприятия.

Перспектива Операций

В рамках перспективы операций каждому узлу-Действию бизнес-процесса ставятся в соответствие конкретные интерфейсы взаимодействия с Исполнителями.

В случае Исполнителя - сотрудника предприятия это будет соответствующим образом закодированная графическая форма. В случае Исполнителя-бота это может быть список спецификаций команд боту (включая типы параметров и возвращаемых значений).

Структура файла-архива бизнес-процесса

Файл с расширением .par

- processdefinition.xml
- forms.xml
- variables.xml
- graph.gif
- start.png
- Файлы форм
- Файлы валидации форм

Граф бизнес-процесса и исполнители (перспектива потока управления и перспектива ресурсов) описываются в файле processdefinition.xml. Файл forms.xml, содержит список соответствующих узлам-Действиям форм. Файл graph.gif, содержит графическое изображение графа бизнес-процесса. Описания используемых в бизнес-процессе форм находятся в файлах, имена которых определены в файле forms.xml. В файле variables.xml описаны переменные бизнес-процесса и их типы. В файлах валидации форм описаны проверки значения переменных, выполняющиеся при выполнении соответствующего

задания.

Версии бизнес-процессов

Версионный механизм основан на следующих принципах:

- При загрузке новой версии определения бизнес-процесса в систему данные, соответствующие новой версии, запоминаются в базе данных. Этой версии присваивается следующий по порядку номер версии. Данные, соответствующие предыдущим версиям при этом не удаляются.
- Система рассматривает определения бизнес-процессов как разные версии одного процесса, если у них совпадает имя процесса.
- Экземпляр бизнес-процесса все время соответствует определению бизнес-процесса, обладавшему наибольшим номером версии на момент запуска этого экземпляра бизнес-процесса. Если во время выполнения этого экземпляра в систему будут загружены следующие версии определения бизнес-процесса, они уже не окажут влияния на его выполнение.

Описание элементов языка определения бизнес-процессов, используемого в системе Runa WFE **Описание файла *processdefinition.xml* и используемых в нем тегов**

Описание файла *processdefinition.xml*

В системе Runa WFE предусмотрены такие понятия, как группы пользователей и функции над оргструктурой. Для того, чтобы иметь возможность с ними работать, надо определять элементы swimlane специальным образом. Это подробно описано в следующем разделе.

Использование ролей (элемента *swimlane*) в системе Runa WFE

Swimlane (роль-Дорожка) –представляет собой специальный тип переменных бизнес-процесса. Используется для определения Пользователей, которые могут выполнить определенное Действие. Роль-Дорожка ставится в соответствие узлу-Действию. В Runa WFE до начала исполнения Действия роли-Дорожке должен быть поставлен в соответствие инициализатор, который возвращает пользователя, либо группу пользователей. Инициализация роли-Дорожки состоит в том, что ей (как переменной) присваивается ID пользователя или группы пользователей. Если роль-Дорожка еще не проинициализирована, то ее инициализация происходит в момент прихода управления в данный узел-Действие. Если роль-Дорожка проинициализирована группой пользователей, то после того, как Действие выполнено, производится доинициализация роли-Дорожки: в роль-Дорожку вместо ID группы помещается ID того пользователя, который выполнил данное Действие.

Опишем понятие инициализации более подробно:

С понятием роль-Дорожка в системе Runa WFE связан алгоритм назначения заданий:

В момент, когда управление попадает в данный узел_Действие, происходит следующее:

- ✓ Если роль-Дорожка проинициализирована группой пользователей, то задания получают все пользователи, входящие в группу, однако только тот пользователь, который первым выполнит это задание, заново проинициализирует роль-Дорожку своим ID.
- ✓ Если роль-Дорожка уже проинициализирована, то задание будет направлено

только тому пользователю, ID которого она проинициализирована.

Если роль-Дорожка не определяется явно как переменная (формой, ботом или в start-state), то ее определение в файле processdefinition.xml обязательно должно содержать тег delegation, в котором должен быть указан класс, наследующий интерфейс org.jbpm.delegation.AssignmentHandler (как правило, это класс ru.runa.wf.jBPM.delegation.assignment.AssignmentHandler) и строку инициализации роли-Дорожки.

Строка инициализации роли-Дорожки должна представлять собой следующее:
<Класс специального вида>(<параметр>, <параметр>, ...)

В настоящее время нами разработаны следующие классы, которые можно употреблять в данном выражении:

- ru.runa.af.organizationfunction.ExecutorByNameFunction – в параметр надо передать имя пользователя или группы, инициализатор будет возвращать ID этого пользователя или ID группы
- ru.runa.af.organizationfunction.DemoChiefFunction – в параметр надо передать имя пользователя, инициализатор будет возвращать руководителя этого пользователя (класс разработан для Демо-примеров)

Каждый параметр представляет собой либо строку, либо \$(<имя переменной бизнес процесса>). В случае переменной бизнес-процесса классу передается значение этой переменной, в случае обычной строки – классу передается эта строка.

Замечание. В самом jPdl строка инициализации роли-Дорожки не специфицирована.

Описание структуры файла

Файл processdefinition.xml представляет собой XML-документ, состоит из «начала файла» и тега process-definition.

Начало файла – это следующий текст:

```
<?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">
```

Тег process-definition описан ниже в данном разделе.

Пример файла 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="simple example">
  <description>This is sample workflow process.</description>
  <!-- SWIMLANES -->
  <swimlane name="requester" />
  <swimlane name="boss">
    <delegation
```

```

class="ru.runa.wf.jBPM.delegation.assignment.AssignmentHandler">ru.runa.af.organizationfun
ction.ExecutorByNameFunction(Administrator)</delegation>
</swimlane>
<!-- START-STATE -->
<start-state name="request " swimlane="requester">
    <transition to="proceed"/>
</start-state>
<!-- NODES -->
<state name="proceed">
    <assignment swimlane="boss" assignment="required" />
    <transition to="done" />
</state>

<!-- END-STATE -->
<end-state name="done" />
</process-definition>

```

Описание тега process-definition

Первым элементом тега является необязательный элемент description. Далее следует блок описания ролей-дорожек (набор тегов swimlane). Далее следует блок описаний типов (набор тегов type). Далее следует начальный узел-Действие в графе бизнес-процесса (тег start-state). Этот тег должен обязательно содержать переход (тег transition) в следующий узел бизнес-процесса. Также этот тег содержит параметр swimlane. Параметру присваивается имя роли-Дорожки, которая будет инициализирована пользователем, запустившим экземпляр бизнес-процесса.

Далее идет список тегов, соответствующих обычным узлам бизнес-процесса. Список типов обычных узлов бизнес-процесса:

- state (узел-Действие)
- decision (исключающий выбор)
- fork (параллельное расщепление)
- join (синхронизация)
- process-state (подпроцесс)

Каждый узел-Действие должен содержать тег assignment с параметром swimlane – значение этого параметра будет определять Исполнителя, которому будет направлено задание после прихода управления в данный узел-Действие бизнес-процесса. Каждый узел должен содержать одну или несколько ссылок на «следующие» узлы бизнес-процесса.

Следующим вложенным тегом тега бизнес-процесса должен быть тег, соответствующий точке окончания бизнес-процесса (тег end-state). После прихода управления в этот узел бизнес-процесс завершается.

DTD-спецификация, описывающая тег process-definition, выглядит следующим образом:

```

<!ELEMENT process-definition ( description?,
    swimlane*,
    type*,
    start-state,
    ( state |
    milestone |
    process-state |

```

```
decision |
fork |
join
)*,
end-state,
action* ) >
```

```
<!ATTLIST process-definition name CDATA #REQUIRED >
```

Замечание. В отличие от jPDL, язык описания бизнес-процессов системы Runa WFE игнорирует тег `type`. Также язык игнорирует тег `action`, вложенный в тег `process-definition`.

Описание элемента `swimlane`

Swimlane (роль-Дорожка) –представляет собой специальный тип переменных бизнес-процесса. Используется для определения Пользователей, которые могут выполнить определенное Действие. Роль-Дорожка ставится в соответствие узлу-Действию.

DTD-спецификация, описывающая тег `swimlane`, выглядит следующим образом:

```
<!ELEMENT swimlane ( description?, delegation? ) >
<!ATTLIST swimlane name CDATA #REQUIRED >
```

DTD-спецификация, описывающая тег `delegation`, выглядит следующим образом:

```
<!ELEMENT delegation ( #PCDATA ) >
<!ATTLIST delegation class CDATA #REQUIRED >
```

Описание элемента `start-state`

Элемент `start-state` соответствует точке старта процесса. В описании бизнес процесса этот элемент должен присутствовать в единственном экземпляре. В отличие от обычных узлов графа бизнес-процесса, `start-state` содержит параметр `swimlane`. Значением параметра является имя роли-Дорожки, которая будет инициализирована ID пользователя, который запустил бизнес-процесс.

К `start-state` может быть присоединена графическая форма. В этом случае она сразу будет показана после выполнения команды «запустить процесс». Эта форма используется для ввода начальных данных бизнес-процесса. До нажатия на кнопку «выполнить» этой формы бизнес-процесс еще реально не будет запущен, то есть, если вообще не заполнить входящую форму (например, закрыть web-браузер), то бизнес процесс не стартует.

DTD-спецификация, описывающая тег `start-state`, выглядит следующим образом:

```
<!ELEMENT start-state ( description?, transition+ ) >
<!ATTLIST start-state name CDATA #REQUIRED
swimlane CDATA #IMPLIED >
```

Описание элемента `state`

Элемент `state` соответствует узлу-Действию (`activity` в терминах диаграммы деятельности языка UML).

DTD-спецификация, описывающая тег `state`, выглядит следующим образом:

```
<!ELEMENT state ( description?, assignment?, action*, transition+ ) >
<!ATTLIST state name CDATA #REQUIRED >
```


Описание вложенного элемента `assignment`

Элемент используется для определения Пользователей, которые могут выполнить определенное Действие. При помощи параметра `swimlane` определенная Роль-Дорожка ставится в соответствие данному узлу-Действию. Если к моменту выполнения Действия эта роль-Дорожка еще не была проинициализирована, то она инициализируется ID того пользователя, который выполнил данное действие.

DTD-спецификация, описывающая тег `assignment`, выглядит следующим образом:

```
<ELEMENT assignment EMPTY >
<!ATTLIST assignment swimlane CDATA #IMPLIED
                    assignment (optional|required) #IMPLIED
                    authentication (optional|required|verify) #IMPLIED >
```

Вложенный элемент `action`

К элементу `state` могут быть «присоединены» `actions` (реализованы java-классами специального вида). Соответствующий код этих классов будет выполнен в случае, если произойдут некоторые `event`'ы, такие, как

- Приход точки управления в узел-Действие
- Уход точки управления из узла-Действия
- и т.д.

Подробнее элемент `action` описан ниже в данном документе.

Вложенный элемент `transition`

Элемент `transition` указывает на следующий узел, в который перейдет точка управления.

Подробнее элемент `transition` описан ниже в данном документе.

Описание элемента `milestone`

Элемент `milestone` соответствует паттерну `milestone`. Текущая версия нашего языка этот элемент не поддерживает.

Описание элемента `process-state`

Элемент `process-state` запускает подпроцесс. При этом базовый процесс ждет в этом узле окончания подпроцесса.

Текущая версия нашего языка этот элемент не поддерживает.

Описание элемента `decision`

Элемент `decision` соответствует маршрутному узлу «Исключающий выбор». «Выбирает» на основании текущих значений переменных бизнес-процесса один из нескольких возможных исходящих переходов. Правила, по которым осуществляется выбор перехода, задаются при помощи тега `delegation`. Разработанный нами класс `ru.runa.wf.jbpm.delegation.decision.BSFDecisionHandler` осуществляет выбор перехода на основании переданного ему в теле тега `BeanShell` скрипта (см. <http://www.beanshell.org/intro.html>). В теле конфигурации доступны все переменные бизнес-процесса (Желательно при их использовании явно приводить их тип). Скрипт должен возвращать значение типа `String`, совпадающее с именем одного из исходящих переходов.

Пример использования тега:

```

<decision name="check business trip type">
  <delegation class="ru.runa.wf.jbpm.delegation.decision.BSFDecisionHandler">
    <![CDATA[
      if( (String) businessTripType.equals("local"))
        return "local";
      else
        return "toAnotherRegion";
    ]]>
  </delegation>
  <transition name="local" to="done"/>
  <transition name="toAnotherRegion" to="Make an order"/>
</decision>

```

DTD-спецификация, описывающая тег decision, выглядит следующим образом:

```

<!ELEMENT decision ( description?, delegation, action*, transition+ ) >
<!ATTLIST decision name CDATA #REQUIRED>

```

Описание элемента fork

Элемент fork соответствует маршрутному узлу «Параллельное расщепление». Для пришедшей в узел точки управления генерируются точки управления для всех исходящих переходов. Все сгенерированные точки управления далее выполняются параллельно. Требуется «закрывающего» элемента join, «собирающего» все порожденные fork'ом точки управления, вместе с которым образует так называемый параллельный блок: Область в бизнес-процессе, у которой есть одна точка входа и одна точка выхода (вход-выход точек управления через «боковые стороны» параллельного блока запрещен).

DTD-спецификация, описывающая тег fork, выглядит следующим образом:

```

<!ELEMENT fork ( description?, delegation?, action*, transition+ ) >
<!ATTLIST fork name CDATA #REQUIRED
  corresponding-join CDATA #IMPLIED>

```

Замечание. Поведение элемента fork можно переопределить через delegation. Для этого существует специальный интерфейс ForkHandler.

Описание элемента join

Элемент join соответствует маршрутному узлу «Синхронизация». Имеет только один исходящий переход. Этот узел «собирает» все порожденные соответствующим fork'ом точки управления. После того, как все точки управления пришли в join, из него выходит единственная точка управления, которая перемещается в следующий узел.

DTD-спецификация, описывающая тег join, выглядит следующим образом:

```

<!ELEMENT join ( description?, delegation?, action*, transition ) >
<!ATTLIST join name CDATA #REQUIRED
  corresponding-fork CDATA #IMPLIED>

```

Замечание. Поведение элемента fork можно переопределить через delegation. Для этого существует специальный интерфейс JoinHandler.

Описание элемента end-state

Элемент end-state соответствует точке окончания процесса. В описании бизнес процесса этот элемент должен присутствовать в единственном экземпляре. В момент прихода управления в этот узел бизнес-процесс полностью завершается.

DTD-спецификация, описывающая тег end-state, выглядит следующим образом:

```

<!ELEMENT end-state EMPTY >
<!ATTLIST end-state name CDATA #REQUIRED>

```

Описание элемента transition

Элемент transition определяет переход между узлами бизнес-процесса.

DTD-спецификация, описывающая тег transition, выглядит следующим образом:

```
<!ELEMENT transition ( action* )>
<!ATTLIST transition name CDATA #IMPLIED
                    to CDATA #REQUIRED>
```

Описание элемента action

Элемент action определяет java код, который будет выполнен ядром WF-системы в случае возникновения тех или иных событий (events) во время выполнения бизнес-процесса.

DTD-спецификация, описывающая тег action, выглядит следующим образом:

```
<!ELEMENT action ( delegation ) >
<!ATTLIST action event-type (process-start|process-end|
                             state-enter|state-leave|state-after-assignment|
                             milestone-enter|milestone-leave|
                             decision-enter|decision-leave|
                             fork-enter|fork-every-leave|
                             join-every-enter|join-leave|
                             transition) #IMPLIED>
```

Поведение элемента action можно определить через delegation. Для этого существует специальный интерфейс ActionHandler.

Описание элемента delegation

Delegation – специальный механизм, при помощи которого разработчик бизнес-процесса может включать в бизнес-процесс свои собственные Java классы. Для загрузки этих классов в ядро в системе предусмотрен специальный class loader.

В зависимости от того, внутри какого тега использован delegation, прилагаемый Java-класс должен реализовывать определенный интерфейс. Например, в случае тега action, это интерфейс ActionHandler, в случае тега decision, это интерфейс DecisionHandler и т.д. Также delegation-класс всегда реализует интерфейс Configurable.

Delegation задается при помощи следующих составляющих:

1. Имя используемого класса – атрибут class (обязательно)
2. Конфигурация для delegation – #PCDATA в теле тега (не обязательно)

DTD-спецификация, описывающая тег delegation, выглядит следующим образом:

```
<!ELEMENT delegation ( #PCDATA ) >
<!ATTLIST delegation class CDATA #REQUIRED>
```

Описание файла forms.xml

В JBOSS JBPM 2.0 структура файла forms.xml определена неформально и этот файл не является обязательным. В Runa WFE структура файла forms.xml определена строго и файл является обязательным. Файл состоит из единственного тега forms. Внутри тега forms находится набор тегов form. Каждый тег соответствует узлу, которому соответствует графическая форма, или в котором присваиваются значения переменным бизнес-процесса.

Файл forms.xml состоит из единственного тега forms. Внутри тега forms находится набор тегов form. Каждый тег соответствует узлу, которому соответствует графическая форма, или в котором присваиваются значения переменным бизнес-процесса.

У тега form есть три обязательных атрибута

- state – название узла бизнес-процесса
- file – имя файла, соответствующего графической форме, которая будет показана в проигрывателе форм для задания из данного узла. Имя каждого файла должно содержать префикс «forms/»
- type – тип формы (в настоящее время можно использовать только «html»)

Полностью XML-схема, определяющая forms.xml находится в папке resource в дистрибутиве системы.

Описание файла определения форм (файлы без расширения)

Каждый файл содержит описание формы на языке HTML, расширенном при помощи дополнительного тега customtag.

Тег customtag содержит следующие атрибуты

- var – имя переменной бизнес-процесса
- delegation – имя класса, который будет использован при работе с переменной через графическую форму (класс должен реализовывать интерфейс VarTag)

Примеры построения файлов-определений бизнес-процессов.

HelloWorld процесс.

Построим простейший бизнес-процесс. Процесс будет заключаться в следующем: После запуска бизнес-процесса на экране появится форма HelloWorld, после нажатия кнопки “Выполнить” в этой форме, процесс завершится.

Этот процесс будет состоять из трех узлов-Действий:

- Начальный узел-Действие, совпадающий с точкой начала бизнес-процесса
- Точка окончания бизнес-процесса

Файл processdefinition.xml будет выглядеть следующим образом:

```
<?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">

<!-- Начало тега process-definition -->
<process-definition name="Hello World">

    <!-- Определение роли-Дорожки -->
    <swimlane name="requester" />

    <!-- Точка начала бизнес-процесса и ввода начальных данных -->
    <start-state name="Hello World state" swimlane="requester">
        <!-- Переход в следующий узел -->
        <transition to="done"/>
    </start-state>

    <!-- Точка завершения бизнес-процесса -->
    <end-state name="done" />

<!-- Завершение тега process-definition -->
```

```
</process-definition>
```

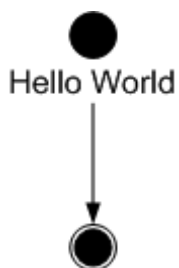
Файл forms.xml будет выглядеть следующим образом:

```
<?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">

  <!-- Тег связывает узел-Действие с графической формой -->
  <form state="Hello World state" file="forms/HelloWorld.form" type="html">
  </form>

</forms>
```

В файл graph.gif запишем следующее изображение:



Файл HelloWorld.form может быть, например, следующим:

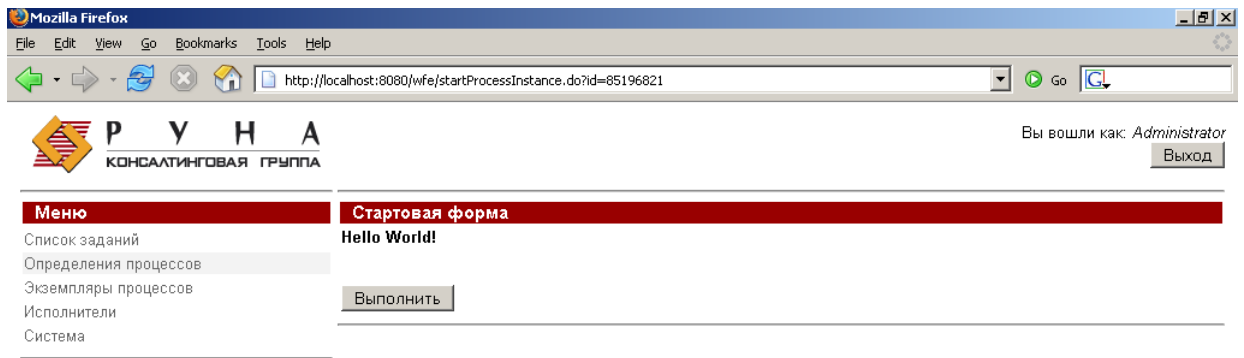
```
<b>Hello World!</b> <br> <br> <br>
```

Структура архива HelloWorld.par будет следующей:

- Файл HelloWorld.par
- processdefinition.xml
 - forms.xml
 - graph.gif
 - HelloWorld

Процесс можно загружать в систему.

При выполнении форма должна выглядеть так:



Процесс сверхурочные.

Краткое описание процесса:

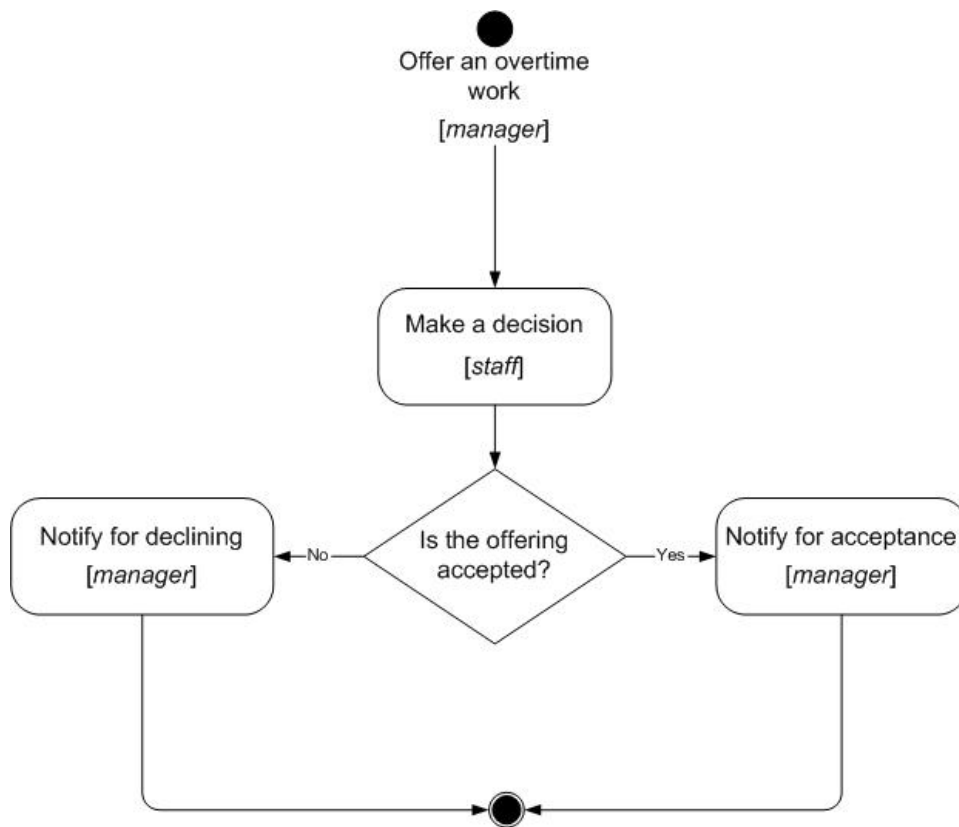
Руководитель предлагает сотруднику выйти на работу сверхурочно – сотрудник соглашается или отказывается. Руководитель получает уведомление соответственно о согласии или об отказе.

Предположим, что все руководители являются членами группы manager, а все сотрудники – членами группы staff.

Проектирование бизнес-процесса

Перспектива управления потоком

Перспектива управления потоком будет соответствовать следующему графу:



Перспектива данных

Введем следующие переменные бизнес-процесса:

Переменная	Описание переменной	Тип	Узел, в котором переменная инициализируется
staff	ID работника	ID	Offer an overtime work
since	Дата-время с...	Дата-время	Offer an overtime work
till	Дата-время по ...	Дата-время	Offer an overtime work
reason	Причина	Строка	Offer an overtime work
comment	Комментарий	Текст	Offer an overtime work
staff person decision	Решение работника	Логический	Make a decision
staff person comment	Комментарий работника	Текст	Make a decision

Перспектива Ресурсов

Введем следующие роли-Дорожки:

- manager - руководитель
- staff person - работник

Инициализация ролей-Дорожек:

Роль-Дорожка	Как инициализируется
manager	Тот, кто запустил бизнес-процесс. Предполагается, что права на запуск данного бизнес-процесса есть только у членов группы manager (руководителей)
staff	Члены группы staff. Предполагается, что в эту группу входят все работники предприятия

Таблица соответствия – в каких узлах какие роли-Дорожки используются:

Узел-Действие	Роль-Дорожка
Offer an overtime work	manager
Make a decision	staff
Notify for declining	manager
Notify for acceptance	manager

Перспектива операций

Обмен данными в этом бизнес-процессе происходит только через графические формы. Соответствие переменных и форм выписано в таблице переменных бизнес-процесса. Тип HTML элемента определяется типом переменной бизнес-процесса во всех случаях, кроме переменной staff. Ее значение определяется Choice'ом, содержащим всех членов группы staff.

Файл-архив бизнес-процесса

Файл 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>Сверхурочные</description>
  <!-- Определение ролей-Дорожек -->
  <!-- Декларируется роль-Дорожка manager, инициализирована будет
  в стартовой точке -->
  <swimlane name="manager" />

  <!-- Декларируется роль-Дорожка staff, инициализирована будет
  в стартовой точке -->
  <swimlane name="staff"/>

  <!-- Стартовая точка -->
  <!-- Роль-Дорожка manager инициализируется пользователем,
```



```

запустившим процесс -->
<!-- Роль-Дорожка staff будет инициализирована при помощи графической формы -->
<start-state name="Offer an overtime work" swimlane="manager">
  <transition to="Make a decision"/>
</start-state>

<!-- Узлы бизнес-процесса -->

<!-- Узел-Действие -->
<state name="Make a decision">
  <description>В этом узле-Действии работник принимает или отклоняет
    предложение сверхурочных работ</description>
  <!-- Задается исполнитель узла-Действия (роль-Дорожка staff) -->
  <assignment swimlane="staff" assignment="required" />
  <transition to="Is the offering accepted"/>
</state>

<!-- Маршрутный узел – исключаяющий выбор -->
<decision name="Is the offering accepted">
<!-- Использование механизма delegation. Выбор нужного перехода задан
    при помощи BeanShell скрипта -->
  <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>

<!-- Узел-Действие -->
<state name="Notify for acceptance">
  <description>Задание – ознакомиться с согласием на
    сверхурочные работы</description>
  <assignment swimlane="manager" assignment="required" />
  <transition to="done" />
</state>

<state name="Notify for declining">
  <description>Задание – ознакомиться с несогласием на
    сверхурочные работы</description>
<assignment swimlane="manager" assignment="required" />
  <transition to="done" />
</state>

<!--Точка завершения процесса -->
<end-state name="done" />

</process-definition>

```

Файл forms.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<forms xmlns="http://runa.ru/xml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://runa.ru/xml forms.xsd">
<form file="Offer an overtime work" jsValidation="false" state="Offer an overtime work"
type="html" validationFile="Offer an overtime work.validation.xml"/>
<form file="Make a decision" jsValidation="false" state="Make a decision" type="html"
validationFile="Make a decision.validation.xml"/>
<form file="Notify of acceptance" jsValidation="false" state="Notify of acceptance"
type="html" validationFile="Notify of acceptance.validation.xml"/>
<form file="Notify of rejection" jsValidation="false" state="Notify of rejection" type="html"
validationFile="Notify of rejection.validation.xml"/>
</forms>
```

Файл variables.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<variables xmlns="http://runa.ru/xml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://runa.ru/xml variables.xsd">
<variable format="org.jbpm.web.formgen.format.DefaultFormat" name="staff"/>
<variable format="ru.runa.wf.web.forms.format.DateTimeFormat" name="since"/>
<variable format="ru.runa.wf.web.forms.format.DateTimeFormat" name="till"/>
<variable format="org.jbpm.web.formgen.format.DefaultFormat" name="reason"/>
<variable format="org.jbpm.web.formgen.format.DefaultFormat" name="comment"/>
<variable format="org.jbpm.web.formgen.format.DefaultFormat" name="staff person
comment"/>
<variable format="ru.runa.wf.web.forms.format.BooleanFormat" name="staffPersonDecision"/>
</variables>
```

Файл graph.gif

Содержание файла соответствует рисунку, приведенному в разделе «Перспектива управления полтоком»

Файлы форм

Файл OfferAnOvertimeWork:

```

<link href="form.css" type="text/css" rel="stylesheet" />
<div class="form-container">
<p class="legend"><strong>Offer an overtime work</strong></p>
<div><label for="staff">Employee <em>*</em></label>

```

<!-- Специальный тег, расширяющий HTML, выдает на экран choice, содержащий список членов группы, название которой передается в переменной var=... . Возвращает ID выбранного члена группы. В теге используется механизм delegation -->

```

<customtag var="staff"
delegation="ru.runa.wf.web.html.vartag.GroupMembersComboboxVarTag" /></div>

```

```

<div><label for="since">Since <em>*</em></label><em><font size="-1">

```

<!-- Специальный тег, расширяющий HTML, служит для работы с датами-->

```

<customtag var="since" delegation="ru.runa.wf.web.html.vartag.DateTimeInputVarTag" />
(dd.mm.yyyy)</font></em></div>

```

```

<div><label for="till">Till <em>*</em></label><em><font size="-1"><customtag var="till"
delegation="ru.runa.wf.web.html.vartag.DateTimeInputVarTag" />
(dd.mm.yyyy)</font></em></div>

```

```

<div><label for="reason">Reason <em>*</em></label><input id="reason" name="reason"
/></div>

```

```

<div><label for="comment">Comments</label><textarea id="comment" wrap="hard"
name="comment"></textarea></div>

```

```

</div>

```

Аналогично данному файлу строятся файлы:

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

Файлы валидаций (проверок) для переменных форм

Файл Offer an overtime work.validation.xml

```

<validators>
<field name="since">
<field-validator type="required">
    <message>Field is required</message>
</field-validator>

```

```

</field>
<field name="reason">
<field-validator type="stringlength">
    <message>Length cannot be more than 100 symbols</message>
    <param name="maxLength">100</param>
</field-validator>
<field-validator type="required">
    <message>Value is required</message>
</field-validator>
</field>
<field name="staff">
<field-validator type="required">
    <message>Field is required</message>
</field-validator>
</field>
<field name="comment">
<field-validator type="stringlength">
    <message>Length cannot be more than 255 symbols</message>
    <param name="maxLength">255</param>
</field-validator>
</field>
<field name="till">
<field-validator type="required">
    <message>Field is required</message>
</field-validator>
</field>
<validator type="expression">
    <message>Till should be later that since</message>
    <param name="expression">till.getTime() > since.getTime()</param>
</validator>
</validators>

```

Аналогично строятся файлы валидации для остальных форм.

Структура архива

Файл overTimeDemo.par

- processdefinition.xml

- forms.xml
- graph.gif
- Файлы форм:
 - OfferAnOvertimeWork
 - MakeaDecision
 - NotifyForAcceptance
 - NotifyForDeclining

Далее процесс можно загружать в систему.

Замечание. Дистрибутив системы также содержит другие демо-процессы:

- VacationDemo.par – отпуск
- BusinessTripDemo – командировка
- TimerDemo – пример использования таймера