RunaWFE. Графический редактор бизнес-процессов. Руководство разработчика.

Версия 3.0

Введение

За основу графического редактора бизнес процессов RunaWFE взят графический редактор бизнес процессов JBOSS JBPM, который был модифицирован соответственно требованиям компании RUNA. RunaWFE технологически построен на основе GEF (Graphical Editing Framework) являющейся частью модульной платформы Eclipse. Платформа Eclipse реализует модель сервисов OSGi (OSGi Framework) на платформе Java.

OSGi Framework предоставляет унифицированную среду для работы приложений (называемых bundles), связывающую:

- среду выполнения приложения (Execution Environment);

- модули, дополняющие политики загрузки классов Java private классами для модуля и контролируемым связыванием модулей;

- управление жизненным циклом модулей приложения, позволяющее динамически устанавливать, запускать, останавливать, обновлять и удалять модули;

- сервисы регистрации, обеспечивающие динамическое совместное использование объектов приложениями.

Платформа Eclipse представляет собой набор подсистем, реализованных небольшим исполняемым приложением ядра и набором модулей (плагинов), расширяющих функциональность платформы. В контексте данного документа термины «модуль» и «плагин» равнозначны и взаимозаменяемы. Использование обоих терминов обусловлено главным образом стилистическими соображениями.

Ядро платформы Eclipse в процессе выполнения динамически обнаруживает, конфигурирует и запускает плагины платформы. Eclipse поддерживает динамическое подключение плагинов, описываемых дескрипторами плагинов (файлах MANIFEST.MF и plugin.xml). Для расширения функциональности, плагины платформы в дескрипторах плагинов определяют точки расширения (extension points). Точка расширения представляет собой xml описание интерфейса расширяемого компонента плагина. Расширяющие плагины используют точки расширения для добавления функциональности. Платформа Eclipse не разграничивает плагины, созданные пользователями и плагины самой платформы.

Платформа Eclipse реализована на Java, что обеспечивает переносимость разработанных приложений для работы на разных платформах под различными операционными системами.

GEF предоставляет основу для создания графических редакторов. GEF реализована как набор плагинов расширяющих плагины платформы Eclipse. GEF связывает элементы модели приложения с их графическими представлениями, реализованными с помощью графических компонент библиотеки Draw2d. Контроллеры GEF поддерживают визуальное представление элементов модели в MVC (model-view-controller) архитектуре. Для каждого элемента представления, соответствующий этому представлению контроллер интерпретирует события интерфейса пользователя и трансформирует их в команды обработки соответствующего элемента модели.

Обобщенная архитектура GEF показана на Рис. 1. Назначение компонентов GEF представлено в Табл. 1.

Табл. 1 Компоненты архитектуры GEF.

Компонент	Назначение компоненты
Модель (Model)	Модель представляет собой сохраняемые данные. Модель должна предусматривать механизм уведомления о внесенных изменениях.
Представление (View)	Представление это визуальное отображение модели. Оно состоит из фигур отображающих элементы модели. Модель может быть представлена как графически, так и виде иерархической (древовидной) структуры.
Контроллер (Controller)	Контроллеры связывают элементы модели и соответствующие им элементы представления. В соответствии с представлением контроллеры могут быть графическими или иерархическими. Они ответственны за редактирование элементов модели через представление, а также отображение изменений элементов модели в представлении. Контроллеры используют политики редактирования – элементы, выполняющие большинство задач редактирования.
Действие (Action)	Действия это элементы, обрабатывающие ввод данных. Действия конвертируют события интерфейса пользователя в запросы, которые используют программный интерфейс контроллеров.
Запрос (Request)	Запросы это элементы инкапсулирующие события интерфейса пользователя. Запросы позволяют абстрагироваться от источника события.
Команда (Command)	Команды инкапсулируют данные об изменениях модели. Команды возвращаются контроллерами в ответ на запросы. Команды также содержат информацию о возможности взаимодействия.
Событие (Event)	События это изменения в интерфейсе пользователя, приводящие к изменениям представления или модели.

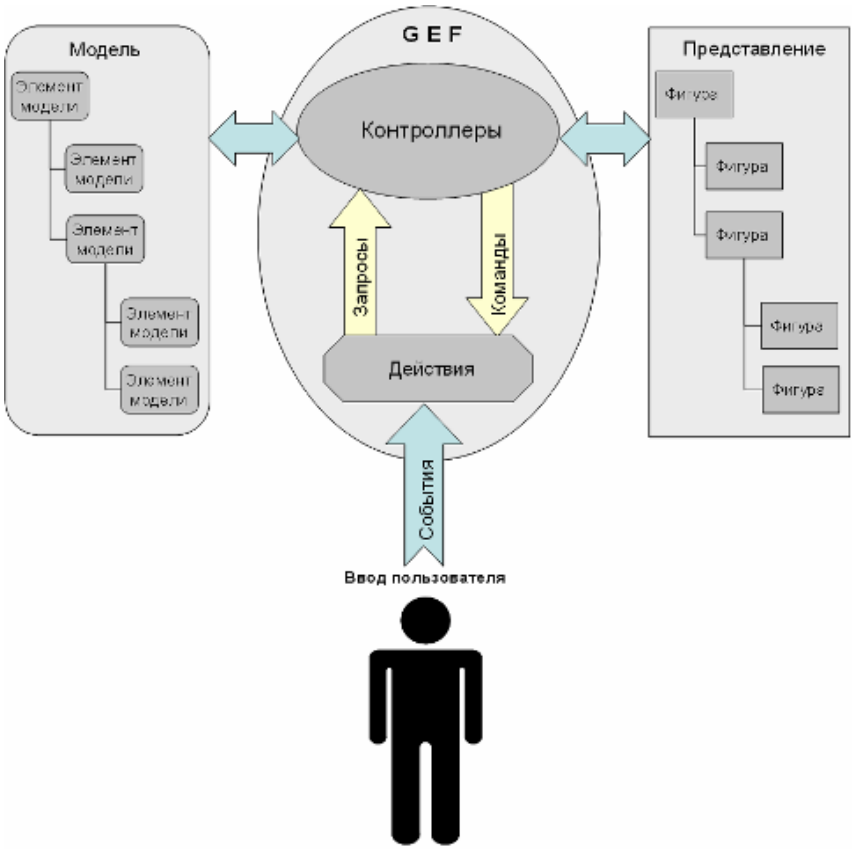


Рис. 1 Обобщенная архитектура GEF.

Модули графического редактора

Графический редактор построен на основе JBOSS JBPM, основной модуль которого jbpm.core загружает и выгружает определения бизнес процессов, создает экземпляры бизнес процессов и потоки их выполнения, а также останавливает выполнение этих потоков. Другие модули графического редактора, при реализации своей функциональности используют сервисы ядра модуля jbpm.core.

Перечень и назначение модулей графического редактора приведены в Табл. 2. Взаимосвязи модулей показаны на Рис. 2.

Табл. 2 Модули графического редактора.

Наименование модуля	Назначение модуля
org.jbpm.core	Содержит библиотеки ядра системы JBOSS JBPM, а также интерфейсы для работы с ядром.
org.jbpm.ui	Модуль содержит пакеты графического редактора JBOSS JBPM, включающие GEF, элементы модели, графические представления. Пакеты JBOSS JBPM используются в графическом редакторе бизнес процессов RunaWFE.
ru.runa.jbpm.ui	Модуль приложения графического редактора бизнес процессов RunaWFE. Построен на основе модуля org.jbpm.ui.
tk.eclipse.plugin.htmleditor	Модуль HTML редактора.
tk.eclipse.plugin.wysiwyg	Модуль визуального (WYSIWYG) редактора. Расширяет функционал tk.eclipse.plugin.htmleditor.

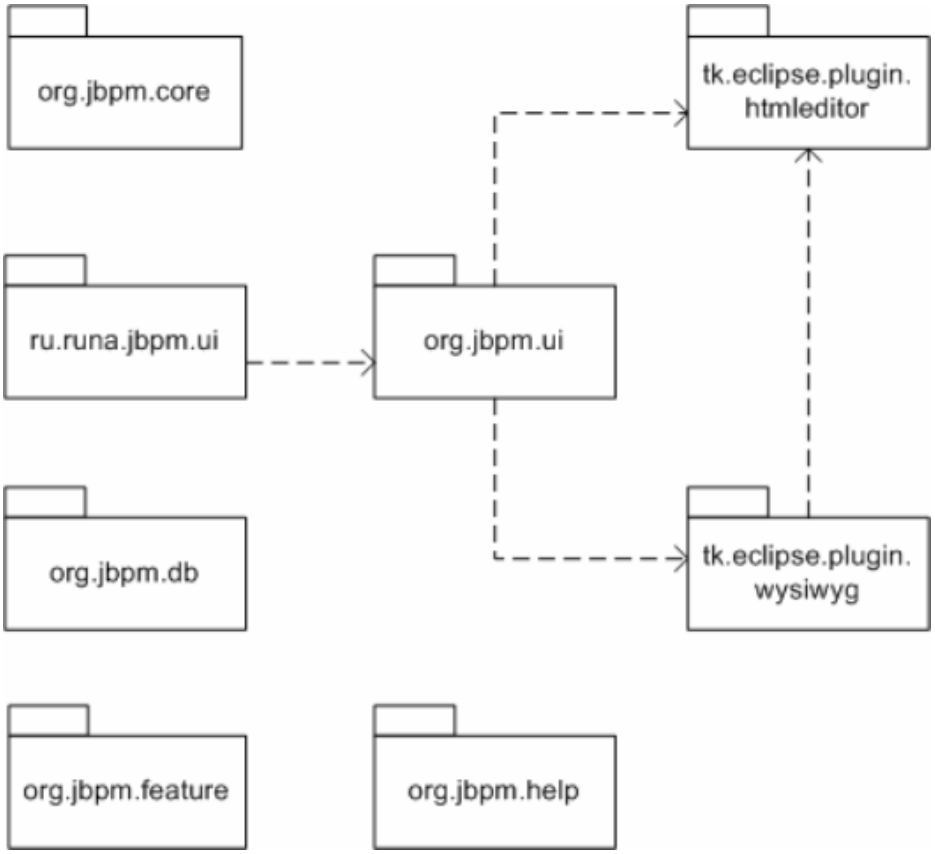


Рис. 2 Зависимости модулей RunaWFE.

Сохранение в XML и чтение модели процесса из XML происходит в одном классе-сериализаторе (для каждой версии jPDL). Обновление версии jPDL для процесса может быть осуществлено итерациями - от предыдущей версии к текущей. Версия используемого jPDL процесса определяется по структуре processdefinition.xml. В случае необходимости добавления новой версии jPDL нужно понимать структуру кода. Отправной точкой служит extension point org.jbpm.ui.elements (описанный в elements.exsd). Каждое из расширений

org.jbpm.ui.elements (сейчас 2.0 и 3.2) содержит список элементов модели + провайдер содержимого этой версии jPDL, который состоит из сериализатора и класса-конвертера из предыдущей версии jPDL.

НАСТРОЙКА СРЕДЫ РАЗРАБОТКИ

Настройка eclipse

Для работы с редактором используется Eclipse 3.4 в редакции для Java EE (с пакетом GEF) .

Нужно импортировать проекты как plug-in project.

Локализация

Нужно установить в eclipse плагины локализации (должны соответствовать по версии), а также (отдельно) локализацию для GEF.

Для eclipse: http://download.eclipse.org/technology/babel/babel_language_packs/ [1].

Для GEF http://www.eclipse.org/gef/translations/translation.html [2] (Group 2)

Скачайте проект gpd из SVN-репозитория на sourceforge https:/ / runawfe. svn. sourceforge. net/ svnroot/ runawfe/ RunaWFE-3. x/ trunk/ gpd [3] и создайте в Eclipse при помощи опции import отдельный проект для каждого плагина проекта gpd (кроме ru.runa.specific) Всего должно получиться 6 проектов.

После установки (и перезапуска eclipse) в проекте org.jpbm.ui откройте gpd.product > Configuration. Удалите все плагины, затем добавьте 6 плагинов проекта gpd и добавьте все зависимости. Теперь можно собирать.

Сборка приложения

Графический редактор бизнес процессов предназначен для работы как отдельное RCP (Rich Client Platform) приложение (продукт). Сборка RCP приложения графического редактора бизнес процессов должна выполняться после внесения изменений в модули редактора.

Экспорт RCP приложения выполняется на основе файла продукта **gpd.product**. Редактор файла продукта содержит вкладки: «Обзор», «Конфигурация», «Бренд».

На вкладке «Обзор» редактора файла продукта задаются:

- идентификатор продукта;

- приложение которое следует запускать при запуске продукта;

- имя отображаемое в заголовке приложения.

В разделе «Тестирование», ссылка «Синхронизировать» используется для обновления дескриптора plugin.xml главного модуля продукта в соответствии с внесенными изменениями в модули продукта в среде разработки. Ссылки «Запуск продукта» и «Запуск в режиме отладки» позволяют протестировать работу RCP приложения без его экспорта.

В разделе «Экспорт», ссылка «Мастер экспорта продукта Eclipse» служит для задания параметров экспорта и экспорта RCP приложения на основе заданной конфигурации экспорта на вкладке «Конфигурация».

На вкладке «Конфигурация» в разделе «Модули и фрагменты» задаются модули входящие в состав RCP приложения. При этом, после задания главного модуля приложения, набор необходимых модулей может быть определен автоматически. Особенности запуска RCP приложения задаются в разделах «Файл конфигурации» и «Аргументы запуска».

Для сборки RCP приложения графического редактора бизнес процессов необходимо выполнить следующие действия:

1. Открыть файл продукта org.jbpm.ui.gpd.product. На вкладке «Обзор» в разделе «Определение продукта» должно быть установлено:

- ИД продукта: org.jbpm.ui.RUNA;

- Приожение: ru.runa.jbpm.ui.bp editor;

- Product Name: RunaWFE GPD;

- Конфигурация продукта основана на plug-in.

1. В разделе «Тестирование» выбрать ссылку «Синхронизировать» для синхронизации внесенных изменений с главным модулем продукта.
2. Для добавления вновь созданных модулей (если таковые есть) в набор, на вкладке «Конфигурация» выбрать ссылку «Добавить» и в открывшемся списке выбрать необходимые модули.

Среда Eclipse предоставляет возможность заново сформировать список необходимых модулей RCP приложения. Для этого следует:

- удалить все модули выбрав ссылку «Удалить все»;

- добавить в пустой список главный модуль RCP приложения;

- выбрать ссылку «Добавить обязательные модули».

1. На вкладке «Обзор» в разделе «Экспорт» выбрать ссылку «Мастер экспорта продукта Eclipse».
2. В окне Мастера экспорта:

- указать полный путь целевого каталога экспорта;

- в опциях компилятора указать совместимость с целевой Java машиной;

- нажать кнопку готово.

RCP приложение будет экспортировано по заданному пути.

Для экспорта RCP приложения на целевые платформы Linux, Macosx, Solaris, в среде разработки Eclipse, должен быть установлен пакет плагинов Eclipse-RCP-delta-pack. Плагин Eclipse-RCP-delta-pack можно установить воспользовавшись ссылками главного меню Help/Software Updates/Manage Configuration для автоматического обновления установленной платформы Eclipse и выбрав плагин в перечне плагинов.

После установки Eclipse-RCP-delta-pack в редакторе файла продукта добавляется вкладка для запуска RCP приложения на выбранной платформе.

РАСШИРЕНИЕ функциональности редактора

GPD предоставляет возможность расширения функциональности базирующейся на eclipse extensions points.

(

http://wiki.eclipse.org/FAQ_What_are_extensions_and_extension_points [4]

http://www.vogella.de/articles/EclipseExtensionPoint/article.html [5]

)

Для начала работы требуется создать новый plug-in проект и добавить в его зависимости необходимые плагины (как минимум те, в которых определены extension points, это org.jbpm.ui, tk.eclipse.plugin.wysiwyg).

Также можно всю работу делать в существующем проекте (например org.jbpm.ui), но это не рекомендуемый подход.

Как создавать новый проект

(

http://www.ibm.com/developerworks/library/os-ecplug [6]

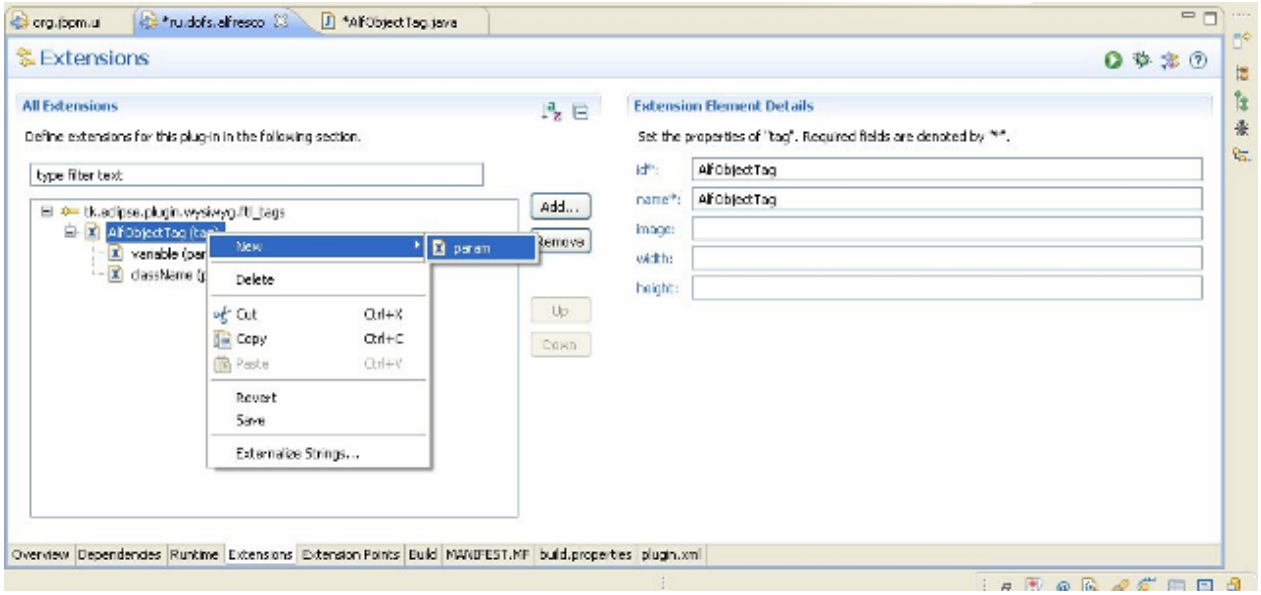
http://www.eclipse.org/articles/Article-Your%20First%20Plug-in/YourFirstPlugin.html [7]

)

Создание (и редактирование) расширений производится стандартным способом.

Рассмотрим на примере добавления нового FTL тега.

Откройте файл plugin.xml, перейдите на вкладку Extensions, добавить и выберите tk.eclipse.plugin.wysiwyg.ftl_tags (если его нет в списке – удостоверьтесь что плагин, объявляющий данную точку расширения находится в зависимостях разрабатываемого плагина)



Действия добавить, удалить элемент дерева (кроме корня – ссылки на точку расширения) производятся в контекстном меню по правой кнопке на выбранном элементе (на рисунке мы добавляем параметр к тегу).

Структура дерева определяется схемой точки расширения, которой можно воспользоваться при возникновении трудностей.

После проделанной работы новый плагин необходимо включить в сборку GPD, это можно сделать 2-мя способами:

- в файле org.jbpm.ui/gpd.product на вкладке Configuration добавить плагин и собрать свежий GPD

- собрать плагин независимо (например из plugin.xml, вкладка Overview -> Export wizard) и собранный jar подложить в {GPD}/plugins. После этого рекомендуется запустить GPD с опцией –clean.

Типы форм

В данный момент поддерживаются 3 типа форм:

- FTL (HTML + Freemarker (www.freemarker.org [8]))

- VarTag (HTML + VarTags (simple template engine))

- InfoPath (формы создаются с помощью MS Infopath)

Этот список можно расширить собственным типом формы.

Схема точки расширения: org.jbpm.ui.formtype

- element[@name] – название типа формы

- element[@contributor] – класс, расширяющий org.jbpm.ui.forms. FormType

- element[@type] – расширение файлов формы

Описание методов org.jbpm.ui.forms. FormType

- IEditorPart openForm(IFile formFile, FormNode formNode)

Открывает форму для редактирования

- String getFormFileName(IFile definitionFile, FormNode formNode)

Возвращает начальное название файла

- Map<String, Integer> getFormVariableNames(IFile formFile, FormNode formNode)

Возвращает список переменных из формы в виде Название – Тип использования {чтение, запись, не удалось определить}

- void validate(IFile formFile, FormNode formNode)

Осуществляет проверку формы

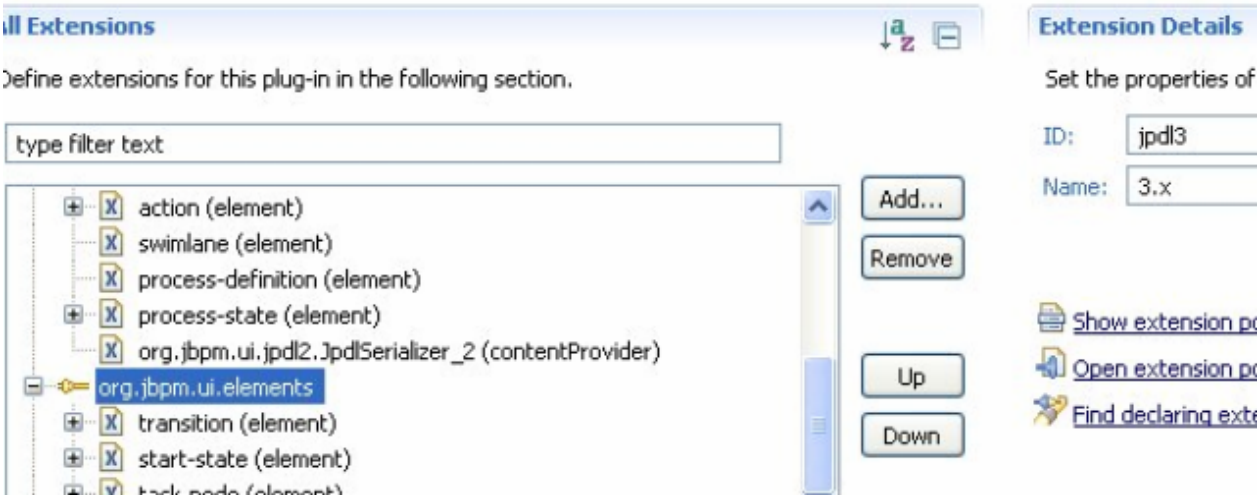
Версии jPDL

Редактор способен поддерживать одновременно несколько версий jPDL (сейчас 2.x и 3.x). Добавление новой версии jPDL сложно, но возможно.

Схема точки расширения отсутствует, используется немного другой подход. При добавлении новой версии нужно добавить расширение org.jbpm.ui.elements, в свойствах которого указать

- @ID – уникальное название

- @Name – версия jPDL



Возможности расширения (или изменения) модели в разных версиях jPDL1) Создание контекстных элементов меню - фильтр по EditParts2) Создание новых свойств (в properties view)

Элементы jPDL

Это элементы, доступные из палитры редактора.

Схема точки расширения: org.jbpm.ui.elements

- element[@name] – уникальное (в пределах данной версии jPDL) название элемента

- element[@model] -класс модели (наследник org.jbpm.ui.common.model.GraphElement)

- element[@graphicalEditPart] – отвечает за отображение на графе

- element[@treeEditPart] – отвечает за отображение в дереве схемы

- element[@figure] – отвечает за отрисовку в графе

- element/entry – необязательный элемент, появляющийся в палитре (при отсутствии элемент не доступен из палитры)

- element/entry[@id] – уникальное название элемента в палитре, по этому полю при отображении палитры производится сортировка внутри группы

- element/entry[@category] – название группы

- element/entry[@label] – текст в палитре

- element/entry[@type] – тип элемента (блок или переход)

- element/entry[@icon] – рисунок в палитре

- contentProvider[@serializerClass] – обеспечивает сохранение и загрузку в данной версии jPDL

- contentProvider[@converterClass] – обеспечивает обновление процесса с прошлой версии jPDL

ActionHandlers

Обработчики, служащие для исполнения кода во время исполнения экземпляра процесса. Могут быть установлены во многих местах процесса.

Схема точки расширения: org.jbpm.ui.delegablePropertyDescriprors

- delegable[@type] – тип {actionHandler|decisionHandler|assignmentHandler}

- delegable[@name] – удобное название в редакторе (только для отображения)

- delegable[@className] – полное название класса

- delegable[@cellEditorProvider] – класс, расширяющий org.jbpm.ui.custom.DelegableProvider. Позволяет создать UI для редактирования конфигурации.

Также см. Appendix A

DecisionHandlers

Используются для элементов Decision в jPDL. Интерфейс определяет метод, принимающий решение по какому переходу следовать на основании переменных экземпляра процесса.

Схема точки расширения: org.jbpm.ui.delegablePropertyDescriprors

То же самое, как и для ActionHandler.

Замечание: delegable[@cellEditorProvider] может реализовать org.jbpm.ui.custom.IDecisionProvider, привнося новую функциональность.

Также см. Appendix A

Организационные функции

Классы, реализующие интерфейс ru.runa.af.organizationfunction. OrganizationFunction называются орг. Функциями. Интерфейс объявляет единственный метод

long[] getExecutorIds(Object[] parameters), который возвращает список кодов исполнителей в WF, параметры определяются реализацией.

Они используются в инициализаторе роли.

Схема точки расширения: org.jbpm.ui.orgfunctions

- orgfunction[@name] – название, используется для отображения в редакторе

- orgfunction[@className] – полное название класса орг. функции

- orgfunction/parameter – определение параметров орг. функции

- orgfunction/parameter [@name] –название, используется для отображения в редакторе

- orgfunction/parameter [@type] – тип переменных, которые могут быть использованы в качестве динамического значения параметра

- orgfunction/parameter [@value] – начальное значение параметра

- orgfunction/parameter [@multiple] – если значение установлено в true, то таких параметров может быть несколько подряд.

Замечание: orgfunction/parameter [@multiple] может быть установлено в true только для последнего параметра.

Конструкторы ролей

Откройте GPD, Роли, кнопка изменить. Появится диалоговое окно с редактированием инициализатора роли. Оно имеет древовидную структуру со степенью вложенности 2 (корневая – переход по табам, вторая – раскрывающиеся блоки в содержимом таба). Элементы GUI, открывающиеся в блоках могут быть добавлены в этот диалог.

См. схему точки расширения: org.jbpm.ui.swimlaneelements

Валидаторы

Используются для проверки переменных. Существует 2 разновидности: глобальные и назначаемые на переменную.

Замечание: в данный момент нельзя добавить новый глобальный валидатор (имеется только валидатор, основанный на BeanShell, можно писать любой java-код).

Схема точки расширения: org.jbpm.ui.validators

- validator[@name] – название валидатора, в WF указывается в файле validators.xml

- validator[@displayName] – отображаемое имя

- validator[@description] - отображаемое описание

- validator[@applicable] – типы переменных, к которым может быть применен данный валидатор. Указывается через запятую, например (long,double). Пустое значение означает все типы.

- validator/param – параметры валидатора

- validator/param[@name] – название параметра. В классе валидатора по этому имени доступно значение параметра.

- validator/param[@displayName] – отображаемое имя

- validator/param[@type] – тип значения параметра

Форматы переменных

Используются в 2-х назначениях:

- определяют тип переменной в WFE

- определяют правила (java код) разбора данных из web формы в WFE

Замечание: WFE позволяет переменной с назначенным форматом X в редакторе присвоить значение типа Y (например с помощью ActionHandler, RMI (если проверка переменных для задачи отсутствует)).

В данный момент нет точки расширения в редакторе для определения нового типа формата.

См. Appendix A

FTL: теги

В терминах freemarker – это методы, которые в WFE исполняются и на выходе дают некий HTML.

Схема точки расширения: tk.eclipse.plugin.wysiwyg.ftl_tags

- tag[@id] – название тега, в WF должно быть объявлено в файле freemarker-tags.xml

- tag[@name] – отображаемое имя тега

- tag[@image] – путь к рисунку тега, отображаемому в графическом редакторе

- tag[@width], tag[@height] – размеры рисунка

- tag/param[@name] - отображаемое имя параметра

- tag/param[@type] – тип отображения параметра (text – поле ввода значения, combo – выбор из списка)

- tag/param[@variableAccess] –задействована ли переменная в данном параметре и как (запись, чтение). Используется для поиска переменных формы.

- tag/param[@values] – список переменных определенного типа в combo, any означает все переменные.

- tag/param/paramValue – статические значения combo параметра. Действуют при пустом значении tag/param[@values] и tag/param[@type] = combo.

- tag/param/paramValue[@name] – отображаемое имя

- tag/param/paramValue[@value] – значение выбранной опции

FTL: отображение переменных

В терминах freemarker –это $test?format

Схема точки расширения: tk.eclipse.plugin.wysiwyg.ftl_formats

- type[@name] – название типа переменной. Список всех типов: string|double|long|date|time|boolean

- type/format – форматирование переменной с помощью freemarker

- type/format[@name] – отображаемое имя

- type/format[@value] – название, следуемое после ? при генерирование FTL кода

Var: теги (deprecated)

См. схему tk.eclipse.plugin.wysiwyg.var_tags

1. == Добавление нового пункта меню ==

Меню графического редактора бизнес процессов RunaWFE содержится в дескрипторе plugin.xml плагина ru.runa.jbpm.ui. Плагин ru.runa.jbpm.ui определяет пункты меню графического редактора путем добавления функциональности в точку расширения org.eclipse.ui.actionSets плагина org.eclipse.ui.

Добавлять и редактировать пункты меню графического редактора удобно на вкладке «Расширения» (Extensions) редактора манифеста модуля среды разработки Eclipse. Заметим, что сами по себе элементы меню не имеют прикладной функциональности. Они скорее являются средством структурирования функциональных элементов actions (действия), которые можно рассматривать как конечные пункты меню («листья» в иерархии меню).

Для добавления пункта меню:

1. Открыть файл plugin.xml плагина ru.runa.jbpm.ui в редакторе манифеста модуля.
2. Создать элемент меню. Для этого в контекстном меню элемента «Главное меню» (main menu) точки расширения org.eclipse.ui.actionSets последовательно выбрать пункты «Создать» (New), затем «menu». Редактор манифеста модуля создаст новый элемент расширения menu, а в правой части окна редактора

отобразятся свойства этого элемента:

- id – уникальный идентификатор элемента;

- label – метка отображаемая на элементе;

- path – путь локализации пункта меню в структуре меню.

1. Добавить элемент сепаратор в созданный пункт меню. Для того, чтобы созданный пункт меню мог использоваться для расширения другими плагинами, сепаратор должен иметь имя «additions».

Для добавления действий:

1. В контекстном меню элемента «Главное меню» (main menu) точки расширения org.eclipse.ui.actionSets последовательно выбрать пункты «Создать» (New), затем «action». Редактор манифеста модуля создаст новый элемент расширения action, а в правой части окна редактора отобразятся свойства этого элемента:

- id – уникальный идентификатор элемента;

- label – метка отображаемая на элементе;

- accelerator – устаревшее, не рекомендуется использовать (Deprecated);

- definitionId – идентификатор команды связанной с данным действием;

- menubarPath – путь локализации действия в структуре меню;

- toolbarPath – путь локализации действия в линейке инструментов;

- icon – относительный путь к файлу, содержащему изображение для данного элемента;

- disabledIcon – относительный путь к файлу, содержащему изображение для неактивного элемента;

- hoverIcon - относительный путь к файлу, содержащему изображение для элемента, когда указатель мыши находится над ним;

- tooltip – текст всплывающей подсказки;

- helpContextId – идентификатор контекстной справки;

- style – атрибут представления действия (push, radio, toggle, pulldown);

- state – необязательный атрибут начального состояния;

- pulldown – устаревшее, не рекомендуется использовать (Deprecated);

- class – полный путь к классу обработчику действия. Класс должен реализовывать интерфейс org.eclipse.ui.IWorkbenchWindowActionDelegate или org.eclipse.ui.IWorkbenchWindowPulldownDelegate. Если атрибут retarget установлен в true, данный атрибут игнорируется;

- retarget – если данный атрибут установлен в true, используется глобальный обработчик действия;

- allowLabelUpdate – используется если атрибут retarget установлен в true. Если данный атрибут установлен в true то label и tooltip данного действия замещаются атрибутами глобального обработчика;

- enablesFor – если данный атрибут не задан, он игнорируется. Определяет сколько элементов должно быть выбрано, чтобы выполнить данное действие.

1. После введения полного пути классу обработчику действия (атрибут class), выбрать ссылку class. В результате среда Eclipse сгенерирует класс обработчик действия по введенному пути, содержащий методы-заглушки. Для задания функциональности действия необходимо реализовать функциональность этих методов.

Appendix A

Обнаружение расширений механизмом загрузки классов, реализующих интерфейсы

Применения: ActionHandler, DecisionHandler, AssignmentHandler, WebFormat

Библиотеку (JAR файл), содержащую класс, реализующий один из вышеперечисленных интерфейсов нужно положить в папку {GPD}/plugin/org.jbpm.core/lib. После этого перезапустить GPD.

Замечание 1: если класс имеет зависимости на сторонние библиотеки (использование в секции imports), то эти библиотеки необходимо положить туда же.

Замечание 2: иногда из-за кеша JDT новые элементы не появляются. Для этого при выключенном GPD необходимо удалить папку {GPD}/workspace/.metadata/.plugins/org.eclipse.jdt.core и запустить GPD снова.

Взаимодействие с MS INFOPATH (Только для ос windows)

Архитектура

В качестве визуального редактора форм взят Microsoft InfoPath 2007, входящий в дистрибутив Microsoft Office 2007. Он позволяет создавать шаблоны форм используя стандартную и настраиваемую палитры элементов формы. Для использования на формах элементов, специфичных для системы RunaWFE, мы будем использовать элементы которые добавим в палитру InfoPath.

Внимание: для корректной работы требуется библиотека RunaGPDInfoPathSupport.dll, которая должна лежать в C:\WINDOWS\SYSTEM32. Она обеспечивает работу с архивными файлами форм (**XSN**), которые находятся в формате **CAB**.

Таблица поддерживаемых RunaWFE стандартных элементов InfoPath.

InfoPath element	Описание
Text Box	Ввод текста в одной строке
Drop-Down List Box	Выпадающий список
Check Box	Флажок
Rich Text Box	Поле для ввода текста, многострочный текст
Date Picker	Выбор даты с помощью календаря
File Attachment	Загрузка файла и скачивание файла
Картинка	Рисунок на форме
ГиперССылка	Открывается в новом окне

Помимо этих стандартных элементов InfoPath позволяет добавить дополнительные элементы, построенные на основе TemplatePart's (шаблонов) или ActiveX (COM компонентов).

Процесс создания ActiveX элементов InfoPath

Используемая среда разработки **Microsoft Visual Studio 2007**.

Добавляемые элементы InfoPath являются ActiveX объектами, реализующими интерфейсы UserControl, IObjectSafety, ICOMControl. В рамках COM модели они должны иметь уникальный GUID. Утилита для создания GUID называется GUIDGEN.EXE и поставляется вместе с MS Visual Studio. Мы будем использовать .Net Framework для простоты, хотя можно обойтись и без него.

Создаем новый проект в MS Visual Studio (Visual C# projects) типа **Windows Class Library**.

В свойствах созданного проекта на вкладке Configuration Properties -> Build выбираем: **Register for COM Interop = true**.

Генерируем новый **GUID**.

Делаем копию template класса, реализующего элемент InfoPath и меняем GUID

(Опционально) Изменяем метод **OnPaint**. Этот метод вызывается всякий раз при отображении элемента на форме InfoPath, соответственно если хочется придать ему красивый вид – то рисовать нужно здесь.

Обращаем внимание на метод **OnSizeChanged** в классе. Он вызывается всякий раз когда пользователь InfoPath на форме пытается изменить размеры этого элемента. Соответственно здесь можно явно контролировать размеры элемента на форме.

После всего нажимаем Project Build, ждем компилирования и создания новой DLL, которая готова к установке на локальный компьютер.

Для установки на других компьютерах требуются дополнительные действия. Подписывание DLL (**Strong Name**) – нужно сгенерировать утилитой **sn.exe** (VS utility) файл ключа (sn.exe –k keyfile.snk), добавить его в проект и в файле AssemblyInfo.cs записать значения [assembly: AssemblyKeyFile("..\\..\\keyfile.snk")], [assembly: AssemblyKeyName("ActorFullNameDisplay")] вместо дефолтных значений.

Далее элемент необходимо добавить в GAC утилитой **gacutil.exe** (VS utility) и зарегистрировать как COM объект утилитой **regasm.exe (.Net utility**).

Бизнес-логика компонента может обращаться к RunaWFE и не ограничена по функциональности.

Из интерфейса MS InfoPath 2007 на закладке Controls (Элементы управления) мы можем по нажатию ссылки Add or Remove custom controls зарегистрировать ActiveX компоненты в качестве элементов, но не для ActiveX, находящихся в .Net категории. Для этой категории мы вынуждены вручную писать описания компонент и ложить их в определенное место.

Путь к файлам описания настраиваемых элементов InfoPath:

${USER_HOME}\Local Settings\Application Data\Microsoft\InfoPath\Controls

Пример:

C:\Documents and Settings\dofs\Local Settings\Application Data\Microsoft\InfoPath\Controls

Описатели элементов должны иметь расширение .**ict**, для того чтобы InfoPath их правильно прочитал при загрузке.

Файлы в этой директории удобно именовать

{GUID}.ict (пример: {C4134657-1B43-4968-9913-FAE952F685A0}.ict), где GUID – GUID элемента InfoPath.

Файл должен быть в кодировке UTF-8.

Формат файла:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ict:control name="CONTROL_NAME">
<ict:designTimeProperties>
<ict:iconBitmap>CONTROL_ICON</ict:iconBitmap>
</ict:designTimeProperties>
<ict:deployment>
<ict:controlPackage classid="{ACTIVEX_GUID}"/>
</ict:deployment>
</ict:control>
```

, где

CONTROL_NAME = название элемента в палитре

CONTROL_ICON = иконка в палитре, в файле представлена в формате Base64

ACTIVEX_GUID = GUID ActiveX.

Если файл некорректно создан, InfoPath выведет сообщение с детализаций ошибки при следующем запуске. Если он не выводит сообщения и не добавляет элемент в палитру – проверьте правильность папки, в которую вы положили ict файл.

В качестве примера смотрите существующие компоненты.

Процесс создания Template Parts элементов InfoPath

Эти компоненты ограничены в функциональности DataSource, который будет отработан при инициализации формы в режиме исполнения. Этот тип элемента удобен для вывода разнообразных списков (используется для вывода акторов).

Публикация состоит в регистрации **XTP** файлов шаблонов в InfoPath из меню Добавить/Удалить элементы.

1. *В качестве примера смотрите существующие шаблоны.*

Литература.

1. Практически исчерпывающую информацию по технологиям упоминавшимся в документе можно получить на сайте http:// www. eclipse. org/ [9]. Кроме того, среда разработки Eclipse имеет развитую справочную систему, которая включает справочную информацию, ссылки на информационные ресурсы по используемым технологиям, примеры. Плагины обновлений, добавляемые с http:// www. eclipse. org/ [9], как правило включают справочную информацию, автоматически подключающуюся к справочной системе среды Eclipse.

3. Для разработчика плагинов можно рекомендовать книгу Building Commercial Quality Eclipse Plug-ins By Eric Clayberg, Dan Rubel. Publisher: Addison WesleyProfessional.ISBN: 032142672X; Published: Mar 22, 2006; Copyright 2006; Dimensions 7x9-1/4; Pages: 864; Edition: 2nd..

2. Информацию об OSGi Framework можно получить на сайте OSGi альянса:

http://www.osgi.org/osgi_technology/index.asp?section=2 [10] .

3. Документация по GEF располагается на сайте: http:// www. eclipse. org/ gef/ reference/ articles. html [11]. Полезная для разработчиков информация содержится в документах http:// wiki. eclipse. org/ index. php/ GEF_Developer_FAQ [12] и http:/ / wiki. eclipse. org/ index. php/ GEF_Troubleshooting_Guide#Draw2D_common_mistakes [13]. Применение компонентов GEF на примере создания редактора схемы базы данных подробно рассматривается в статье http://www. eclipse. org/articles/ Article-GEF-editor/gef-schema-editor.html [14].

4. GEF-tutorials:

http://www-128.ibm.com/developerworks/opensource/library/os-gef/ [15]

http://eclipsewiki.editme.com/GefDescription [16]

Примечания

[1] http://download.eclipse.org/technology/babel/babel_language_packs/
[2] http://www.eclipse.org/gef/translations/translation.html
[3] https://runawfe.svn.sourceforge.net/svnroot/runawfe/RunaWFE-3.x/trunk/gpd
[4] http://wiki.eclipse.org/FAQ_What_are_extensions_and_extension_points
[5] http://www.vogella.de/articles/EclipseExtensionPoint/article.html
[6] http://www.ibm.com/developerworks/library/os-ecplug
[7] http://www.eclipse.org/articles/Article-Your%20First%20Plug-in/YourFirstPlugin.html
[8] http://www.freemarker.org/
[9] http://www.eclipse.org/
[10] http://www.osgi.org/osgi_technology/index.asp?section=2
[11] http://www.eclipse.org/gef/reference/articles.html
[12] http://wiki.eclipse.org/index.php/GEF_Developer_FAQ
[13] http://wiki.eclipse.org/index.php/GEF_Troubleshooting_Guide#Draw2D_common_mistakes
[14] http://www.eclipse.org/articles/Article-GEF-editor/gef-schema-editor.html
[15] http://www-128.ibm.com/developerworks/opensource/library/os-gef/
[16] http://eclipsewiki.editme.com/GefDescription