RUNA WFE. Графический редактор бизнес-процессов. Руководство разработчика

Версия 2.1

Оглавление

1ВВЕДЕНИЕ

За основу графического редактора бизнес процессов RUNA WFE взят графический редактор бизнес процессов JBOSS JBPM, который был модифицирован соответственно требованиям компании RUNA. RUNA WFE 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.

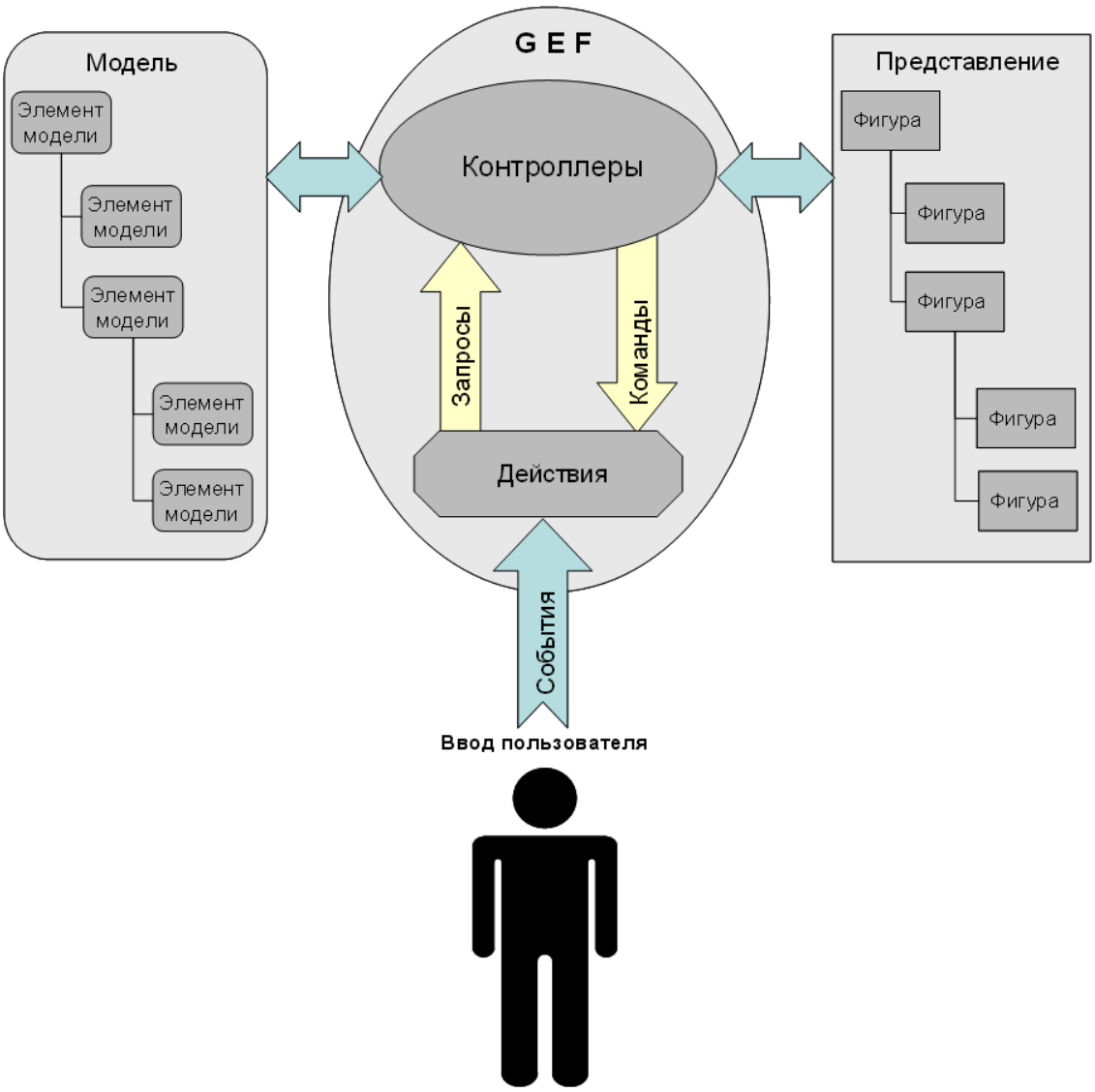

Рис. 1 Обобщенная архитектура GEF.

2МОДУЛИ 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.db	В текущей реализации модуль не используется.
org.jbpm.feature	Модуль объединяет модули редактора в группу.
org.jbpm.help	Модуль справочной подсистемы редактора. В текущей реализации не содержит справочных данных .
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 RUNA WFE.
ru.runa.jbpm.ui	Модуль приложения графического редактора бизнес процессов RUNA WFE. 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.

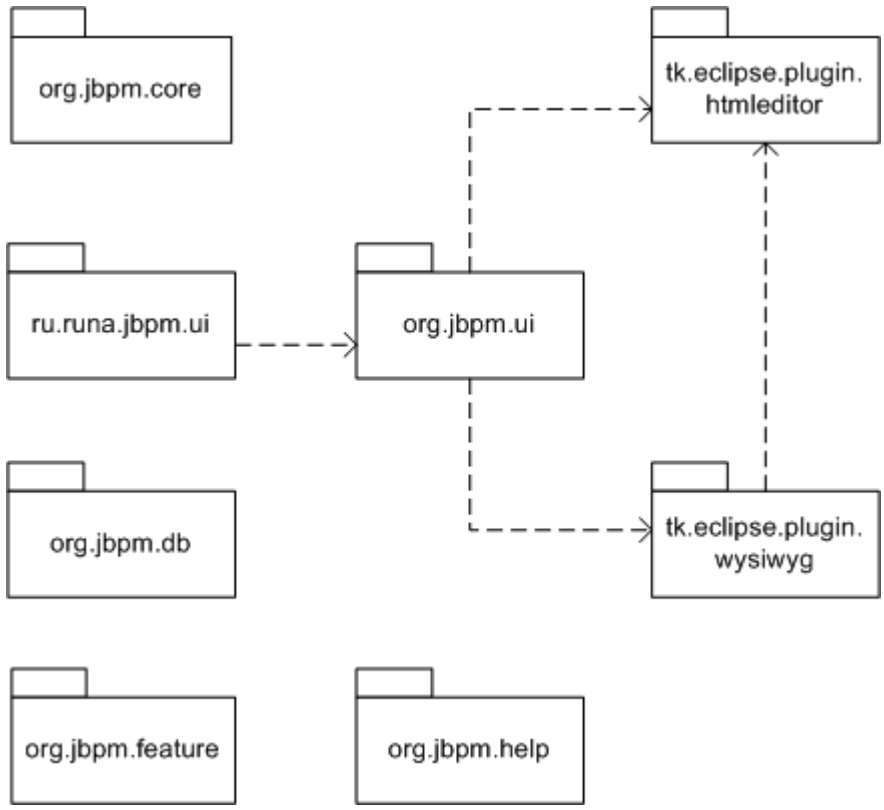

Рис. 2 Зависимости модулей RUNA WFE.

3НАСТРОЙКА ECLIPSE ДЛЯ РАБОТЫ С ГРАФИЧЕСКИМ РЕДАКТОРОМ

Для работы с редактором используется Eclipse 3.1.2. В Eclipse надо установить следующие плагины:

- ✓ WTP
- ✓ WST
- ✓ GEF

Замечание: Все эти пакеты уже установлены в дистрибутив Eclipse wtp-all-in-one-sdk

4ПАКЕТЫ МОДУЛЯ ORG.JBPM.UI

Пакеты модуля org.jbpm.ui реализуют базовую функциональность редактора бизнес процессов на основе GEF. Пакеты соответствуют компонентам архитектуры GEF и содержат классы реализующие соответствующие компоненты архитектуры GEF. Кроме того модуль содержит пакеты, реализующие интерфейс пользователя. Перечень и назначение пакетов модуля приведены в Табл. 3.

Табл. 3 Пакеты модуля org.jbpm.ui.

Наименование пакета	Назначение пакета
org.jbpm.ui.action	Пакет содержит классы реализующие действия (actions), выполняемые GEF при взаимодействии с пользовательским интерфейсом графического редактора.
org.jbpm.ui.command	Пакет содержит классы, наследующие от org.eclipse.gef.commands.Command. Эти классы реализуют команды выполняемые контроллерами и изменяющие модель, в ответ на запросы пользовательского интерфейса редактора.
org.jbpm.ui.contributor	Пакет содержит классы создающие объекты элементов модели, объекты фигур, а также соответствующие им графические и иерархические контроллеры.
org.jbpm.ui.dialog	Пакет содержит класс обработчика дескриптора элементов, выбранных в диалоговом окне.
org.jbpm.ui.editor	Пакет содержит классы редакторов визуальных компонентов GUI редактора бизнес процессов.
org.jbpm.ui.factory	Пакет содержит классы-фабрики элементов и адаптеров.
org.jbpm.ui.figure	Пакет содержит классы реализующие изображения фигур в окне графического редактора.
org.jbpm.ui.model	Пакет содержит классы элементов модели бизнес процессов.
org.jbpm.ui.outline	Пакет содержит классы реализующие иерархическое представление в окне графического редактора.
org.jbpm.ui.part.graph	Пакет содержит классы реализующие графические контроллеры объектов модели.
org.jbpm.ui.part.tree	Пакет содержит классы реализующие иерархические контроллеры объектов модели.
org.jbpm.ui.policy	Пакет содержит классы реализующие политики (поведение) обработки данных контроллерами.
org.jbpm.ui.prefs	Пакет содержит классы параметров элементов окна настроек («Параметры») модуля.
org.jbpm.ui.properties	Пакет содержит классы редактора свойств ячейки
org.jbpm.ui.resource	Пакет содержит класс сообщений, а также дескрипторы бизнес процесса и формы
org.jbpm.ui.util	Пакет содержит вспомогательные классы редактора бизнес процессов.
org.jbpm.ui.view	Пакет содержит классы отображения окна иерархического представления модели.
org.jbpm.ui.wizard	Пакет содержит классы реализующие мастера (Wizard) создания объектов графического редактора.

Рис. 3 Взаимосвязи классов графического редактора WFE.

В подразделах ниже описаны основные пакеты модуля org.jbpm.ui.

4.1Пакет org.jbpm.ui.figure

Пакет содержит классы визуального представления (фигуры) элементов модели бизнес процессов. Классы пакета наследуют от базового класса org.eclipse.draw2d.Figure библиотеки Draw2d. Диаграмма наследования классов показана на Рис. 4. Назначение классов приведено в Табл. 4.

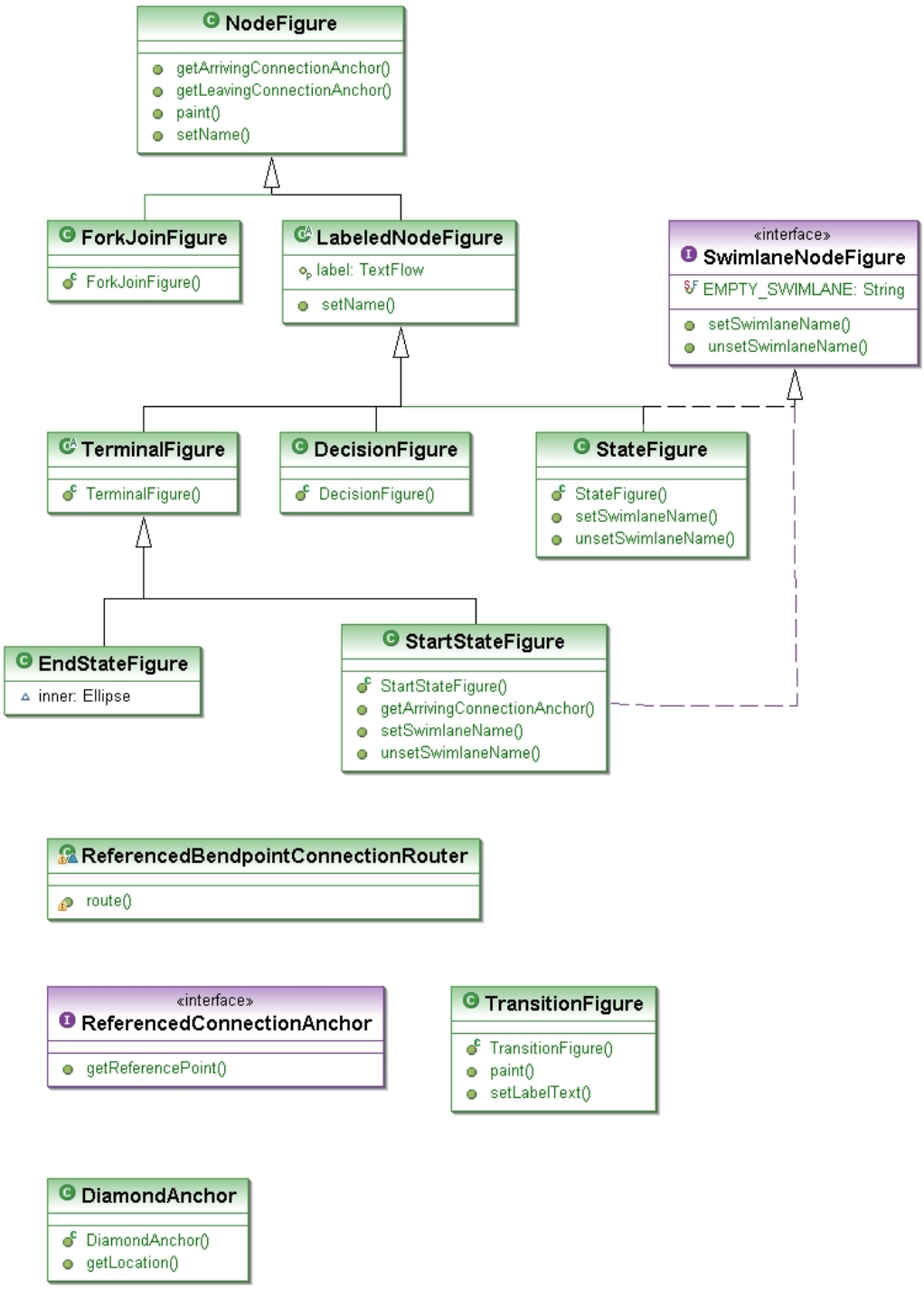
NodeFigure
getArrivingConnectionAnchor()
getLeavingConnectionAnchor()
paint()
setName()
ForkJoinFigure
ForkJoinFigure()
LabeledNodeFigure
label: TextFlow
setName()
«interface»
SwimlaneNodeFigure
EMPTY_SWIMLANE: String
setSwimlaneName()
unsetSwimlaneName()
TerminalFigure
TerminalFigure()
DecisionFigure
DecisionFigure()
StateFigure
StateFigure()
setSwimlaneName()
unsetSwimlaneName()
EndStateFigure
inner: Ellipse
StartStateFigure
StartStateFigure()
getArrivingConnectionAnchor()
setSwimlaneName()
unsetSwimlaneName()
ReferencedBendpointConnectionRouter
route()
«interface»
ReferencedConnectionAnchor
getReferencePoint()
TransitionFigure
TransitionFigure()
paint()
setLabelText()
DiamondAnchor
DiamondAnchor()
getLocation()

Рис. 4 Диаграмма наследования классов представлений графических элементов модуля org.jbpm.ui.figure.

Табл. 4 Классы пакета org.jbpm.ui.figure.

Наименование класса	Назначение класса
DecisionFigure	Класс реализует изображение узловой фигуры «Исключающий выбор».
DiamondAnchor	Класс реализует точку привязки (anchor) связи внутри фигуры.
EndStateFigure	Класс реализует изображение узловой фигуры «Конечное состояние».
ForkJoinFigure	Класс реализует изображение узловых фигур «Расщепление» и «Синхронизация».
LabeledNodeFigure	Абстрактный базовый класс. Определяет управление именем (label) узловой фигуры.
NodeFigure	Базовый класс. Определяет базовое поведение узловой фигуры.
ReferencedBendpointConnectionRouter	Класс реализует маршрутизацию линии для заданной связи (перехода состояния).
ReferencedConnectionAnchor	Интерфейс переопределяет метод получения базисной (reference) точки.
StartStateFigure	Класс реализует изображение узловой фигуры «Старт» (начальное состояние).
StateFigure	Класс реализует изображение узловой фигуры «Состояние».
SwimlaneNodeFigure	Интерфейс определяет статическую константу EMPTY_SWIMLANE и декларирует методы установки setSwimlaneName(String swimlaneName) и сброса unsetSwimlaneName.
TerminalFigure	Абстрактный базовый класс для фигур начального и конечного состояний.
TransitionFigure	Класс реализует изображение фигуры «Переход» (переход состояния).

4.2Пакет org.jbpm.ui.editor

Наименование класса	Назначение класса
DesignerActionRegistry	Контейнер для действий (actions), выполняемых в графическом редакторе. Добавляет действия в

	стек действий.
DesignerContentProvider	Класс реализует методы для предоставления данных, описывающих элементы модели.
DesignerDropTargetListener	Расширяет класс TemplateTransferDropTargetListener для конвертации в фабрику.
DesignerEditor	Класс содержит методы реализующие функции графического редактора.
DesignerEditorActionBarContributor	Класс для инсталляции/деинсталляции и управления элементами меню и соответствующими окнами графического редактора бизнес процессов.
DesignerGraphicalEditorPart	Класс графического представления бизнес процессов в GUI WFE RUNA.
DesignerPaletteRoot	Класс палитры графических элементов бизнес процессов в GUI WFE RUNA.
DesignerSwimlaneEditorPage	Класс редактора роли бизнес процесса.
DesignerVariableEditorPage	Класс редактора переменных состояния бизнес процесса.
ImageHelper	Вспомогательный класс для формирования изображения графического представления бизнес процессов в GUI WFE RUNA.
PaletteFlyoutPreferences	Класс для сохранения/загрузки настроек палитры графических элементов бизнес процессов в GUI WFE RUNA.

4.3Пакет org.jbpm.ui.model

Пакет org.jbpm.ui.model содержит классы элементов модели графического редактора. Диаграмма наследования классов элементов модели показана на Рис. 5. Назначение классов приведено в Табл. 5.

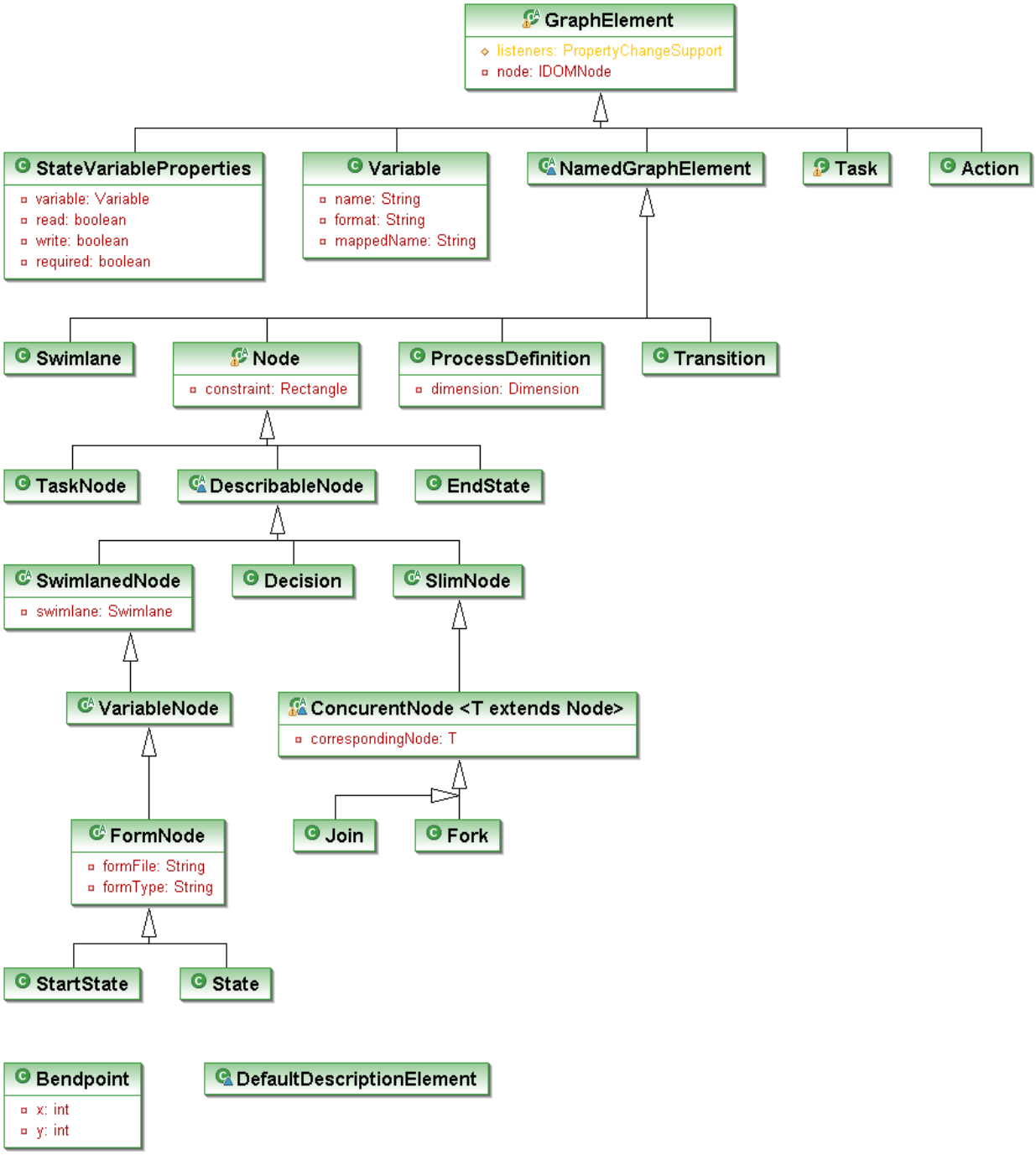

Рис. 5 Диаграмма наследования классов графических элементов модуля org.jbpm.ui.model.

Табл. 5 Классы пакета org.jbpm.ui.model.

Наименование класса	Назначение класса
Action	Класс элемента действие (action). Наследует от абстрактного класса GraphElement. Реализует методы, которые: - получают и устанавливают имя класса делегата и конфигурацию элемента; - получают и устанавливают атрибут event-type (тип события).
Active	Интерфейс декларирует методы для:

	- добавления, удаления и получения списка действий (action) элемента.
Bendpoint	Класс элемента модели. Определяет точку перегиба элемента «Переход» (соединение). Определяет методы: - получения координат точки перегиба; - установки местоположения точки перегиба.
ConcurentNode	Абстрактный класс узлового элемента модели. Наследует от класса SlimNode. Является базовым классом для классов Join и Fork. Определяет класс для корреспондирующих узлов Join или Fork. Реализует метод propertyChange интерфейса PropertyChangeListener.
Decision	Класс узлового элемента модели «Исключающий выбор». Наследует от абстрактного класса DescribableNode. Реализует методы для: - полученият списка действий для узла; - добавления, удаления действия (action) в иерархии действий; - получения и установки свойств узла; - проверки возможности добавления и переподключения соединения («Переход»).
DefaultActionElement	Класс реализует методы интерфейса Active, которые: - добавляют, удаляют действия (action) и получают список действий элемента.
DefaultDelegationElement	Класс реализует методы интерфейса Delegable, которые: - получают и устанавливают имя класса делегата элемента; - получают и устанавливают конфигурацию класса делегата элемента.
DefaultDescriptionElement	Класс реализует методы интерфейс Describable, которые: - получают и устанавливают свойство «описание» элемента (description). Класс использует статические константы интерфейса NotificationMessages.
DefaultStateActionElement	Наследует от класса DefaultActionElement. Переопределяет метод добавления действия (action).
Delegable	Интерфейс декларирует методы для: - получения и установки имени класса делегата элемента; - получения и установки конфигурации класса делегата элемента.
Describable	Интерфейс декларирует методы для: - получения и установки свойства «описание» элемента (description).
DescribableNode	Абстрактный класс узлового элемента модели, имеющего свойство «Описание». наследует от абстрактного класса Node. Является базовым для

	классов Decision, SlimNode, SwimlanedNode. Определяет свойство «Описание» (description) для элемента модели. Реализует методы для получения и установки значения данного свойства.
ElementType	Класс определяет методы, выполняющие конфигурирование типов элементов модели: - инициализацию и отображение типов элементов из манифеста модуля; - получение типов элементов из коллекций; - создание интерфейсов ElementContributor для типов элементов.
EndState	Класс узлового элемента модели «Конец». Наследует от класса Node. Класс определяет методы: - получения префикса имени элемента «end»; - установки размера области графического элемента; - проверок возможностей добавления и подключения входящих и исходящих соединений.
EventTypes	Интерфейс определяет статические константы для типов событий.
Fork	Класс узлового элемента модели «Расщепление». Класс определяет: - префикс имени элемента «fork»; - корреспондирующий класс узла Join («Синхронизация»); - выполняет проверку возможности добавления и переподключения входящих и исходящих соединений («Переход»).
FormNode	Абстрактный класс узлового элемента модели. Наследует от VariableNode. Является базовым классом для классов StartState, State. Определяет форму для элементов «Состояние» и «Старт».
GraphElement	Абстрактный класс. Реализует методы интерфейсов: EventTypes, IPropertySource, NotificationMessages, INodeAdapter. Класс определяет методы, выполняющие: - инициализацию узлового элемента модели из XML файла; - получения уровня иерархии и типа элемента; - получения, добавления, удаления элементов в иерархии элементов; - добавления и удаления слушателей (listener) событий изменения свойств.
InternalState	Маркерный интерфейс для маркирования состояний процесса между начальным и конечным состояниями.
Join	Класс узлового элемента модели «Синхронизация». Класс определяет:

	- префикс имени элемента «join»; - корреспондирующий класс узла Fork («Расщепление»); - выполняет проверку возможности добавления и переподключения входящих и исходящих соединений («Переход»).
NamedGraphElement	Абстрактный класс. Наследует от класса GraphElement. Является базовым для классов Node, ProcessDefinition, Swimlane, Transition. Реализует методы для получения и установки имени узла.
Node	Абстрактный класс узлового элемента модели. Наследует от класса NamedGraphElement. Является базовым для классов DescribableNode, EndState, TaskNode. Определяет узловой элемент модели. Реализует методы для: - получения префикса имени; - получения и задания области узла; - получения и формирования имен исходящих переходов; - проверки элемента как родителя данного элемента; - получения списков входящих и исходящих соединений (элементов «Переход»); - добавления и удаления исходящих соединений; В классе декларируются методы проверки возможности добавления и подключения входящих и исходящих соединений.
NotificationMessages	Интерфейс определяет статические константы для сообщений.
ProcessDefinition	Класс процесса. Наследует от класса NamedGraphElement. Класс Реализует методы интерфейсов Active и Describable. Методы интерфейса Active реализуются с использованием методов класса DefaultActionElemen, которые добавляют, удаляют действия, получают список действий. Методы интерфейса Describable реализуются с использованием методов класса DefaultDescriptionElement, которые получают устанавливают свойство «Описание» для процесса. Класс определяет методы, которые: - получают и задают размеры элемента «процесс»; - инициализируют узел и устанавливают его имя «process»; - формируют имена узлов процесса, ролей, переменных состояний; - добавляют, удаляют, получают списки узлов и ролей процесса; - определяют свойство «Описание» процесса; - определяет равенство объектов.

SlimNode	Абстрактный класс узлового элемента модели. Наследует от класса DescribableNode. Является базовым классом для ConcurentNode. Реализует методы: - получения списка действий для узла, - добавления, удаления действия (action) в иерархии действий; - устанавления геометрических размеров области узла.
StartState	Класс узлового элемента модели «Старт». Наследует от абстрактного класса FormNode. Класс определяет методы: - получения префикса имени элемента «start»; - установки размера области графического элемента; - получения входящих и исходящих соединений; - получения, добавления, удаления задач (tasks); - получения и удаления роли (swimlane); - проверок возможностей добавления и подключения входящих и исходящих соединений.
State	Класс узлового элемента модели «Состояние». Наследует от абстрактного класса FormNode. Класс определяет методы: - добавления и удаления действий (action); - получения списка действий; - получения префикса имени элемента «state»; - получения и удаления роли (swimlane); - присваивания и получения присвоенной роли; - проверок возможностей добавления и подключения входящих и исходящих соединений.
StateVariableProperties	Класс графического элемента модели переменной состояния. Наследует от абстрактного класса GraphElement. Определяет методы для: - установки и проверки свойств переменных элемента «Состояние» (State).
Swimlane	Класс элемента модели «Роль». Класс определяет методы: - получающие и задающие свойство конфигурация узла; - получающие и задающие класс делегат. - получающие и задающие свойство описание узла.
SwimlanedNode	Абстрактный класс узлового элемента модели содержащего роль. Наследует от класса DescribableNode. Является базовым классом для класса VariableNode. Определяет свойство «Роль» для узла. Определяет методы для установки, получения, удаления свойства «Роль». Переопределяет методы getPropertyValue и

	setPropertyValue для получения и установки PROPERTY_SWIMLANE.
Task	Класс графического элемента модели «Задание» (task). Наследует от абстрактного класса GraphElement. Определяет методы для: - получения и установки имени элемента; - получения имени родительского элемента; - получения и установки даты выполнения задания; - получения, добавления, удаления узлов присвоения (assignment), контроллера (controller); - получения и установки узлов присвоения и контроллера; - получения и установки типа конфигурации узлам присвоения и контроллера; - получения и добавления конфигурационной информации узлам присвоения и контроллера; - добавления и получения переменных и списков переменных контроллера; - проверки наличия и установки атрибута "blocking"; - проверки возможности присваивания заданного имени.
TaskNode	Класс узлового элемента модели. Наследует от абстрактного класса Node. Класс определяет методы: - формирования имени элемента; - получения, добавления, удаления элемента; - получения списка элементов; - получения дочернего объекта класса Task; - получения дочернего объекта класса Transition (Переход); - проверки возможности добавления и подключения входящих и исходящих соединений (элементов «Переход»).
Transition	Класс элемента модели «Переход». Наследует от абстрактного класса NamedGraphElement. Переопределяет его абстрактный метод canSetNameTo (String name), который проверяет наличие элемента «Переход» с именем name у родительского элемента данного элемента «Переход». Переопределяет методы добавления, удаления, а также получения списка действий. Определяет методы добавления, удаления, установки точек перегиба (bendpoint) фигуры «Переход», а также получения списков точек перегиба. Определяет методы получения и установки элемента-источника и элемента-приемника для данного элемента «Переход». Определяет метод получения имени данного

	элемента «Переход», имен элементов источника и приемника. Определяет метод установки имени данного элемента «Переход».
Variable	Класс элемента модели «Переменная состояния». Наследует от класса GraphElement. Класс определяет методы: - установки и получения значений переменных name, format, mappedName; - определения равенства объектов класса.
VariableNode	Абстрактный класс узлового элемента модели. Наследует от класса SwimlanedNode. Является базовым классом для класса FormNode. Определяет методы: - добавления и удаления свойств переменных состояния элементов «Состояние», «Старт». - получения и установки списков свойств переменных состояния.

4.4Пакет контроллеров org.jbpm.ui.part

Контроллеры графического редактора разделены по пакетам графических контроллеров org.jbpm.ui.part.graph и контроллеров иерархического представления org.jbpm.ui.part.tree.

4.4.1Пакет контроллеров графических элементов org.jbpm.ui.part.graph

Пакет контроллеров графических элементов содержит классы контроллеров представления графических элементов модели на диаграмме бизнес процессов графического редактора. Диаграмма наследования классов контроллеров графических элементов показана на Рис. 6. Назначение классов приведено в Табл. 6.

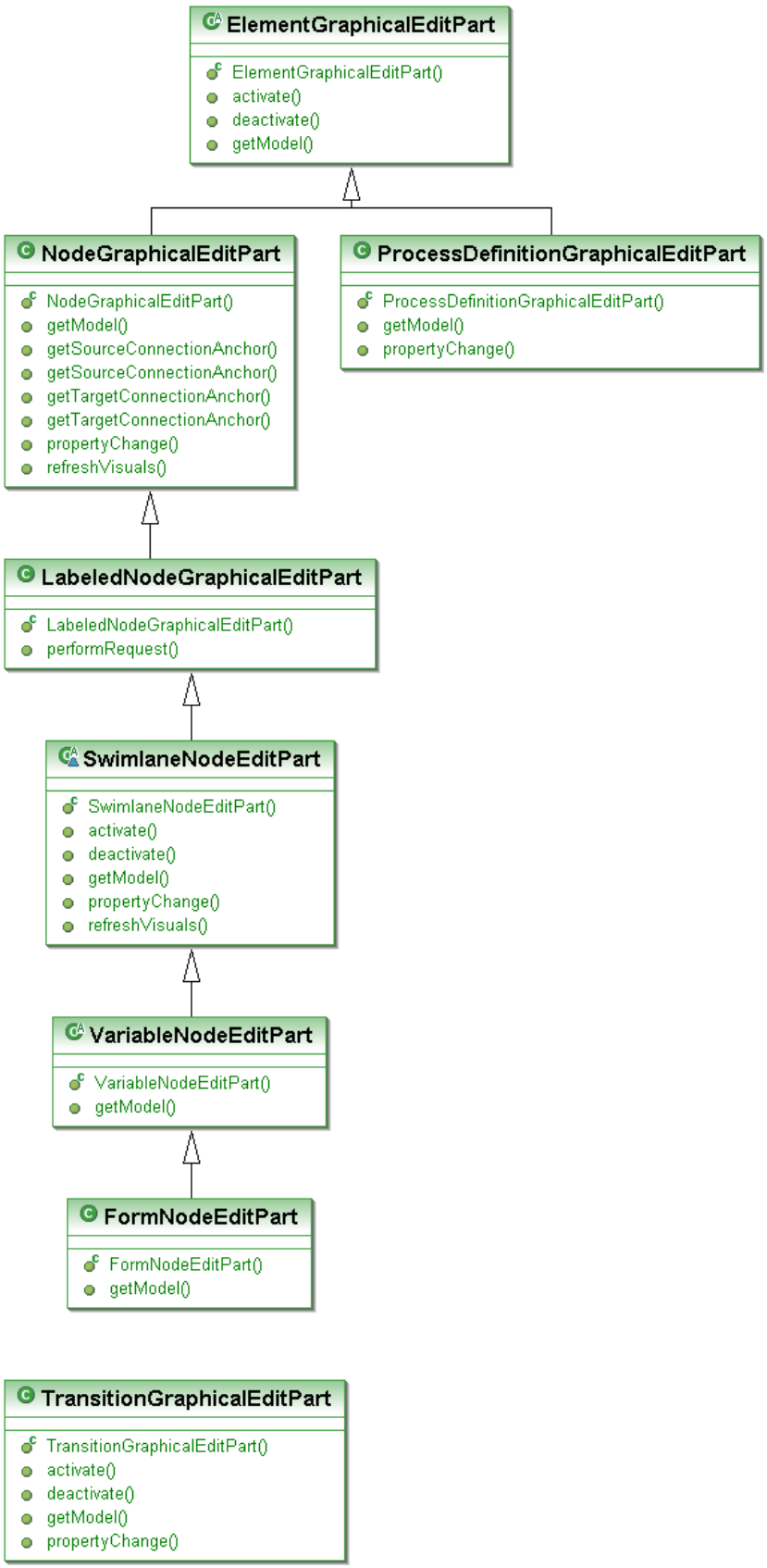
ElementGraphicalEditPart
ElementGraphicalEditPart()
activate()
deactivate()
getModel()
NodeGraphicalEditPart
NodeGraphicalEditPart()
getModel()
getSourceConnectionAnchor()
getSourceConnectionAnchor()
getTargetConnectionAnchor()
getTargetConnectionAnchor()
propertyChange()
refreshVisuals()
ProcessDefinitionGraphicalEditPart
ProcessDefinitionGraphicalEditPart()
getModel()
propertyChange()
LabeledNodeGraphicalEditPart
LabeledNodeGraphicalEditPart()
performRequest()
SwimlaneNodeEditPart
SwimlaneNodeEditPart()
activate()
deactivate()
getModel()
propertyChange()
refreshVisuals()
VariableNodeEditPart
VariableNodeEditPart()
getModel()
FormNodeEditPart
FormNodeEditPart()
getModel()
TransitionGraphicalEditPart
TransitionGraphicalEditPart()
activate()
deactivate()
getModel()
propertyChange()

Рис. 6 Диаграмма наследования классов контроллеров графических элементов.

Табл. 6 Классы пакета org.jbpm.ui.part.graph.

Наименование класса	Назначение класса
ElementGraphicalEditPart	Класс контроллера графического элемента. Наследует от org.eclipse.gef.editparts.AbstractGraphicalEditPart. Переопределяет методы: - получения элемента объекта модели; - получения визуального представления элемента; - активации и деактивации контроллера элемента.
FormNodeEditPart	Класс контроллера узлового элемента содержащего форму. Наследует от класса VariableNodeEditPart. Переопределяет метод получения объекта модели, для получения объекта содержащего форму.
LabeledNodeGraphicalEditPart	Класс контроллера узлового элемента имеющего метку (обозначение). Переопределяет метод выполнения запроса, а также определяет private методы для редактирования метки.
NodeGraphicalEditPart	Класс контроллера узлового элемента. Наследует от класса ElementGraphicalEditPart. Переопределяет и реализует методы: - получения узлового элемента объекта модели; - получения и создания визуального представления узла; - создания политик (линий поведения) контроллера; - создания анкеров источника и приемника для входящих и исходящих соединений; - получения списков входящих и исходящих соединений; - обновления визуального представления узла; - изменения свойств узла.
ProcessDefinitionGraphicalEditPart	Класс контроллера процесса. Наследует от класса ElementGraphicalEditPart. Переопределяет и реализует методы: - получения узлового элемента процесса модели; - получения списка дочерних элементов; - создания политик (линий поведения) контроллера процесса; - изменения свойств узла.
SwimlaneNodeEditPart	Абстрактный класс контроллера узлового элемента содержащего роль (Swimlane). Наследует от класса LabeledNodeGraphicalEditPart. Переопределяет методы базовых классов для: - обновления визуального представления роли;

	- получения узлового элемента объекта модели содержащего роль; - получения объекта модели «роль» (Swimlane); - активации и деактивации контроллера; - изменения свойств роли.
TransitionGraphicalEditPart	Класс контроллера элемента Transition («Переход») . Наследует от org.eclipse.gef.editparts.AbstractConnectionEditPart. Переопределяет методы: - получения объекта модели соединение («Переход»); - создания и обновления визуального представления соединения; - получения списка точек перегиба соединения; - создания политик (линий поведения) контроллера соединения; - активации и деактивации контроллера соединения; - изменения свойств соединения.
VariableNodeEditPart	Абстракный класс контроллера узлового элемента состояния имеющего переменные состояния. Наследует от класса SwimlaneNodeEditPart. Переопределяет метод получения объекта модели, для получения объекта имеющего переменные состояния.

4.4.2Пакет контроллеров иерархического представления org.jbpm.ui.part.tree

Пакет контроллеров иерархического представления элементов содержит классы контроллеров иерархического представления элементов модели в окне графического редактора.

Диаграмма наследования классов контроллеров иерархического представления элементов показана на Рис. 7. Назначение классов приведено в Табл. 7.

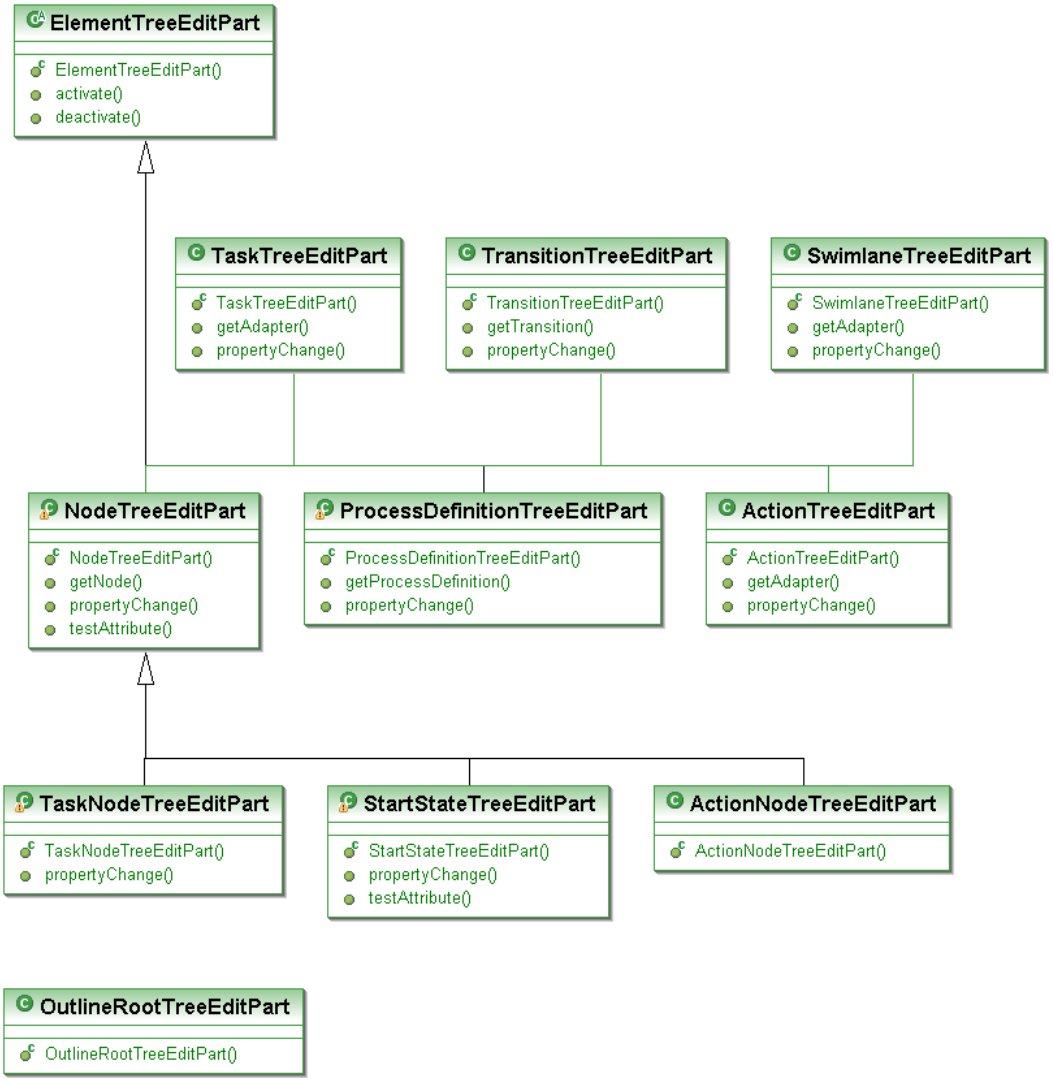

Рис. 7 Диаграмма наследования классов контроллеров иерархического представления.

Табл. 7 Классы пакета org.jbpm.ui.part.tree.

Наименование класса	Назначение класса
ActionNodeTreeEditPart	Класс контроллера действия (action) узлового элемента модели. Наследует от NodeTreeEditPart. В конструкторе класса выполняется проверка

	того, что элемент может выполнять действия (реализует интерфейс Active).
ActionTreeEditPart	Класс контроллера элемента действия (action). Наследует от ElementTreeEditPart. Определяет методы: - получения элемента action модели из текущего контроллера; - обновления визуального представления элемента action текущего контроллера при изменении его свойств; - получения адаптера визуального представления элемента.
ElementTreeEditPart	Абстрактный класс контроллера элемента. Наследует от AbstractTreeEditPart. Определяет методы: - получения графического элемента объекта модели; - активации и деактивации контроллера.
NodeTreeEditPart	Класс контроллера узлового элемента модели. Определяет методы: - получения узлового элемента модели; - получения списка дочерних элементов элемента, в том числе действий (actions); - обновления визуального представления элемента; - обновления дочерних элементов при изменении свойств элемента; - проверки того что элемент является узлом.
OutlineRootTreeEditPart	Класс корневого контроллера. Наследует от AbstractTreeEditPart. Определяет методы для получения списка процессов модели.
ProcessDefinitionTreeEditPart	Класс контроллера элемента процесса (process definition). Наследует от ElementTreeEditPart. Определяет методы: - получения элемента process definition модели из текущего контроллера; - получения списка дочерних элементов (узлов, ролей, действий); - обновления элемента и его дочерних элементов при изменении его свойств.
StartStateTreeEditPart	Класс контроллера элемента начального состояния «Старт». Наследует от NodeTreeEditPart. Определяет методы: - получения списка дочерних элементов состояния «Старт»; - обновления дочерних элементов при изменении свойств элемента; - проверки, что элемент и его родительские элементы не имеют заданий (атрибут "hasTask" имеет значение "false").
SwimlaneTreeEditPart	Класс контроллера элемента «Роль» (swimline). Наследует от ElementTreeEditPart. Определяет

	методы: - получения элемента swimline модели из текущего контроллера; - обновления визуального представления элемента swimline текущего контроллера; - обновления элемента и его дочерних элементов при изменении его свойств; - получения адаптера визуального представления элемента.
TaskNodeTreeEditPart	Класс контроллера узла задание. Наследует от NodeTreeEditPart. Определяет методы: - получения списка дочерних элементов типа задание (task) и исходящее соединение (transition); - обновления визуального представления узла задания контроллера; - получения списка заданий из узлового элемента модели; - получения списка заданий из текущего узлового элемента контроллера; - приведения в соответствие списки заданий текущего узлового элемента контроллера и соответствующего ему элемента модели при добавлении или удалении элементов;
TaskTreeEditPart	Класс контроллера элемента задания (task). Наследует от ElementTreeEditPart. Определяет методы: - получения элемента task модели из текущего контроллера; - обновления визуального представления элемента task текущего контроллера при изменении его свойств; - получения адаптера визуального представления элемента.
TransitionTreeEditPart	Класс контроллера элемента «Переход» (transition). Наследует от ElementTreeEditPart. Определяет методы: - получения элемента transition модели из текущего контроллера; - обновления визуального представления элемента transition текущего контроллера; - получения списка заданий элемента transition текущего контроллера; - обновления элемента и его дочерних элементов при изменении его свойств.

4.5Пакет политик org.jbpm.ui.policy

Пакет политик содержит классы реализующие выполнение задач редактирования (запросов) контроллерами. Назначение классов приведено в Табл. 8.

Табл. 8 Классы пакета org.jbpm.ui.policy.

Наименование класса	Назначение класса
NodeComponentEditPolicy	Класс реализует политику удаления узла по запросу. Содержит метод createDeleteCommand, который создает команду NodeDeleteCommand (удаления узла).
NodeDirectEditPolicy	Класс реализует политику прямого редактирования узла по запросу. Содержит методы: - получения команды NodeSetNameCommand для узла; - отображения внесенных изменений.
NodeGraphicalNodeEditPolicy	Класс реализует политику создания и подключения соединения (Connection). Содержит методы: - получения команды для создания подключения соединения приемника; - получения команды для создания подключения соединения источника; - получения команды для переподключения соединения источника; - получения команды для переподключения соединения приемника.
ProcessDefinitionXYLayoutEditPolicy	Класс реализует политику размещения графического элемента на диаграмме. Содержит методы: - создания команды добавления потомка; - получения команды изменения области узла потомка; - получения команды создания узла потомка; - создания команды удаления потомка.
TransitionConnectionBendpointEditPolicy	Класс реализует политику управления точками перегиба (Bendpoint) соединения. Содержит методы: - получения команды создания точки перегиба соединения; - получения команды удаления точки перегиба соединения; - получения команды перемещения точки перегиба соединения.
TransitionConnectionEditPolicy	Класс реализует политику редактирования соединения (transition). Содержит метод получения команды удаления соединения.
TransitionConnectionEndpointsEditPolicy	Класс реализует политику управления концами соединения. Содержит методы:

	- добавления дескрипторов концов соединения (точек присоединения); - удаления дескрипторов концов соединения (точек присоединения).

5МОДИФИКАЦИЯ ФУНКЦИОНАЛЬНОСТИ РЕДАКТОРА

5.1Модификация представления графического элемента

Представления графических элементов содержатся в пакете org.jbpm.ui.figure. Диаграмма наследования, перечень и назначение классов представлений пакета org.jbpm.ui.figure приведены в разделе 4.1 настоящего документа.

Для задания параметров изображений фигур представлений графических элементов пакета org.jbpm.ui.figure, классы фигур переопределяют методы класса org.eclipse.draw2d.Figure. В общем случае, для изменения графического представления необходимо переопределить метод paintFigure в наследниках класса Figure библиотеки draw2d.

Для изображения фигур используются методы класса org.eclipse.draw2d.Graphics.

В таблице Табл. 9. представлены классы пакета org.jbpm.ui.figure и их методы, реализующие изображения фигур, используемых для моделирования процессов в RUNA WFE. Для модификации представления графических элементов необходимо внести изменения в представленные классы.

Табл. 9 Классы фигур и их методы формирования изображений.

Наименование класса	Методы формирования изображения
StateFigure (Состояние)	void paintFigure(Graphics g) – метод получает объект класса Graphics библиотеки draw2d, рисующий фигуру. В методе формируется минимальный окружающий фигуру прямоугольник (область фигуры), задаются координаты его левого верхнего угла, ширина и высота. Метод drawRoundRectangle класса Graphics рисует сформированный и переданный ему прямоугольник со скругленными углами.
DecisionFigure (Исключающий выбор)	void paintFigure(Graphics g) – метод получает объект класса Graphics библиотеки draw2d, рисующий фигуру. В методе формируется минимальный окружающий фигуру прямоугольник (область фигуры), вычисляются координаты середин его сторон, которые передаются в виде массива методу drawPolygon класса Graphics.

	Метод drawPolygon рисует ромб по заданным ему вершинам.
ForkJoinFigure (Расщепление, Синхронизация)	void paintFigure(Graphics g) – метод получает объект класса Graphics библиотеки draw2d, рисующий фигуру. В методе формируется минимальный окружающий фигуру прямоугольник (область фигуры), задаются координаты его левого верхнего угла. Методы setBackgroundColor и fillRectangle класса Graphics задают черный цвет заполнения и заполняют этим цветом прямоугольник.
StartStateFigure (Старт)	void addEllipse() – метод изображает окружность вызывая конструктор класса Ellipse библиотеки draw2d, заполняет её черным цветом и устанавливает её размеры и компоновку.
EndStateFigure (Конец)	void addEllipse() – метод изображает окружность вызывая конструктор класса Ellipse библиотеки draw2d, устанавливает её прозрачность, размеры и компоновку. Для изображения внутреннего черного круга метод вновь вызывает конструктор класса Ellipse, заполняет окружность черным цветом, устанавливает для неё меньшие размеры и добавляет в ранее изображенную оккружность.

5.2Добавление нового графического элемента

Новые элементы графического редактора процессов RUNA WFE должны создаваться на основе GEF. Контроллеры GEF связывают элементы модели бизнес процесса и их графические представления.

5.2.1Создание элемента модели

Классы элементов модели графического редактора содержатся в пакете org.jbpm.ui.model.

Создание нового элемента модели следует начать с выбора базового класса, зависящего от назначения создаваемого элемента. Диаграмма наследования классов элементов модели показана на Рис. 5, перечень и назначение классов пакета org.jbpm.ui.model приведены в разделе 4.3 настоящего документа. Конечные («листовые») классы в иерархии классов реализуют текущий набор элементов модели RUNA WFE.

Для создания нового элемента модели следует:

1. Создать класс элемента модели, наследующий от класса с наиболее подходящими свойствами (см. Табл. 5);
2. Переопределить неподходящие и нереализованные методы базовых классов;
3. Определить методы, добавляющие новую функциональность создаваемому классу.

5.2.2 Создание графического представления элемента модели

Графические представления элементов модели соответствуют выбранной в RUNA WFE нотации представления элементов модели.

Классы графических представлений содержатся в пакете org.jbpm.ui.figure. Так как графическое представление модели должно отображать взаимосвязи зависимости элементов модели, классы графических представлений организованы в иерархию. Диаграмма наследования классов графических представлений элементов модели показана на Рис. 4, перечень и назначение классов пакета org.jbpm.ui.figure приведены в разделе 4.1 настоящего документа. Конечные («листовые») классы в иерархии классов реализуют текущий набор графических представлений элементов модели RUNA WFE. Так же, как и создание нового элемента модели, создание нового графического представления следует начать с выбора базового класса, зависящего от назначения создаваемого элемента.

Для создания нового графического представления (фигуры) следует:

1. Создать класс элемента модели, наследующий от класса с наиболее подходящими свойствами (см. Табл. 4);
2. Переопределить неподходящие и нереализованные методы базовых классов. Описание методов формирования изображений фигур для конечных классов графических представлений приведены в Табл. 9;
3. При необходимости, определить методы, добавляющие новую функциональность создаваемому классу фигуры.

5.2.3Добавление графического представления в палитру инструментов

Изображения графических элементов добавляются на диаграмму процесса графического редактора RUNA WFE с помощью палитры инструментов. Добавление инструмента в палитру инструментов выполняется путем добавления элемента в точке расширения org.jbpm.ui.elements с помощью редактора файла манифеста модуля plugin.xml.

Для добавления элемента в палитру:

1. На вкладке «Расширения» редактора манифеста, в контекстном меню точки расширения org.jbpm.ui.elements (открывается кликом правой клавиши мыши на точке расширения), добавить элемент.
2. В разделе «Сведения об элементе расширения» ввести имя элемента и выбрать ссылку «contributor». В открывшемся мастере атрибутов java выбрать:

- пакет - org.jbpm.ui.contributor;

- имя класса – конртрибутора;

- в текстовом окне «Интерфейсы» добавить реализуемый классом интерфейс ElementContributor. В результате будет сгенерирован класс реализации указанного интерфейса.

3. Реализовать методы класса – контрибутора для создания экземпляров:

- элемента модели;

- графического контроллера элемента модели;

- контроллера иерархического представления элемента модели;

- фигуры (графического представления) элемента модели;

Для создания экземпляров, часто целесообразно воспользоваться конструкторами соответствующих базовых классов.

5.3Добавление нового пункта меню

Меню графического редактора бизнес процессов RUNA WFE 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.

3. 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.

2. 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.

5.4Как добавить новый тег в "V" элемент конструктора форм

редактирование vartags.

Файл vartags.xml (находится в tk.eclipse.plugin.wysiwyg/vartags) содержит все доступные в графическом редакторе vartags в формате:

<vartag type="ru.runa.wf.web.html.vartag.ActorComboboxVarTag" image="ChooseActor.png" width="160" height="27" />

type - реальный java тип класса VarTag, обязателен

image - название рисунка только для графического отображения, опционально

Рисунки территориально находятся в проекте tk.eclipse.plugin.wysiwyg в папке *FCKeditor2.2\editor\plugins\RunaVarTags\im*

Рисунки также могут быть локализованы, т.е. для разных локалей будут браться разные рисунки, например для данного примера мы вправе положить в требуемую папку картинки ChooseActor.png, ChooseActor.ru.png. И если пользовательская локаль **RU**, то будет использована для отображения картинка ChooseActor.ru.png, иначе - ChooseActor.png.

Общий формат названия файлов картинок - ${fileName}.(locale).${fileExtension}.

width - ширина, опционально, по умолчанию 200

height - высота, опционально, по умолчанию 30.

Также для отображения и для списка выбора требуются имена вартегов. Их нужно задать в файлах локализации messages(.*).properties в формате:

ru.runa.wf.web.html.vartag.GroupMembersAutoCompletionVarTag=Group Members Auto Completion VarTag

т.е. ключом строки выступает java тип VarTag.

Если список после редактирования не изменился нужно удалить папку ${gpd}/workspace/.metadata/.plugins/tk.eclipse.plugin.wysiwyg, она будет создана заново

6СБОРКА RCP ПРИЛОЖЕНИЯ РЕДАКТОРА

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, имеющего расширение .product. Файл продукта может быть автоматически создан средой Eclipse при создании проекта модуля на основе шаблона RCP приложения. Кроме того, файл продукта может быть создан при создании конфигурации продукта. Просмотр и редактирование файла продукта выполняется в редакторе файла продукта среды разработки Eclipse. 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: Runa WFE GPD;
   - Конфигурация продукта основана на plug-in.
2. В разделе «Тестирование» выбрать ссылку «Синхронизировать» для синхронизации внесенных изменений с главным модулем продукта.
3. 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».
4. На вкладке «Обзор» в разделе «Экспорт» выбрать ссылку «Мастер экспорта продукта Eclipse».
5. 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.

Кроме того, для платформы Eclipse 3.1.2. пакет Eclipse-RCP-delta-pack можно получить с сайта http://archive.eclipse.org/eclipse/downloads/drops/R-3.1.2-200601181600/index.php.

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

7ЛИТЕРАТУРА.

1. Практически исчерпывающую информацию по технологиям упоминавшимся в документе можно получить на сайте http://www.eclipse.org/. 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/, 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 .

3. Документация по GEF располагается на сайте: http://www.eclipse.org/gef/reference/articles.html. Полезная для разработчиков информация содержится в документах http://wiki.eclipse.org/index.php/GEF_Developer_FAQ и http://wiki.eclipse.org/index.php/GEF_Troubleshooting_Guide#Draw2D_common_mistakes. 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.

4. GEF-tutorials:
http://www-128.ibm.com/developerworks/opensource/library/os-gef/
http://eclipsewiki.editme.com/GefDescription