МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВПО МГТУ «СТАНКИН»)

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ. ЛАБОРАТОРНЫЕ РАБОТЫ

Учебно-методическое пособие

В пособии приведен алгоритм выполнения лабораторных работы по дисциплине «Проектирование информационных систем», включающий задания, требования к заданиям, а также методические рекомендации по выполнению заданий.

Предназначено для студентов направления подготовки бакалавров 09.03.03 «Прикладная информатика».

2022

1. ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

1.1 ЛАБОРАТОРНАЯ РАБОТА 1. UML. Диаграмма вариантов использования

В соответствии с определенной для курсового проекта темой исследования:

- Разработать диаграмму вариантов использования.
- Оформить отчет по лабораторной работе.
- Разместить отчет в электронной образовательной среде.

Требования к разработке диаграммы:

- Диаграмма должна содержать не менее 3 актеров
- Диаграмма должна содержать не менее 6 вариантов использования (не включая отношения включения, расширения и обобщения)
- Обязательное использование отношений включения, расширения, обобщения (не менее 1 каждого типа).
- Для каждого варианта использования создать краткое текстовое описание по форме:
  - *Краткое описание поведения*, реализуемого в варианте использования;
  - *Предусловия* – условия, которые должны быть соблюдены, прежде чем вариант использования может быть задействован. Например, таким условием может быть завершение выполнения другого варианта или наличие у пользователя прав доступа.
  - *Основной поток событий* описывает, что должно происходить во время выполнения варианта использования в наиболее типовом случае. В этом случае варианты использования связаны с базовыми отношениями включения.
  - *Альтернативные потоки событий* описывают исключительные ситуации (например, ввод неправильного пароля или необходимость выполнения дополнительных действий).
  - *Постусловия* – условия, которые должны быть выполнены после завершения варианта использования. Например, таким условием может быть формирование отчетной документации.

1.2 ЛАБОРАТОРНАЯ РАБОТА 2. UML. Диаграмма деятельности

В соответствии с определенной для курсового проекта темой исследования:

- Разработать диаграммы деятельности.
- Оформить отчет по лабораторной работе.
- Разместить отчет в электронной образовательной среде.

Требования к разработке диаграммы:

- Диаграммы должны быть сформированы для не менее 2 вариантов использования (по выбору).
- Каждая диаграмма должна содержать не менее 15 действий.
- Обязательное использование дорожек Swimlanes

1.3 ЛАБОРАТОРНАЯ РАБОТА 3. UML. Диаграмма последовательности, диаграмма состояний

В соответствии с определенной для курсового проекта темой исследования:

- Разработать диаграммы последовательности
- Разработать диаграммы состояний.
- Оформить отчет по лабораторной работе.
- Разместить отчет в электронной образовательной среде.

Требования к разработке диаграммы последовательности:

- Представить диаграммы последовательности для не менее 1 варианта использования и связанной с ним диаграммы активностей
- Обязательное использование на диаграмме не менее 6 объектов системы
- Обязательное использование фреймов взаимодействия (не менее 1 фрейма)

Требования к разработке диаграммы состояний:

- Представить диаграмму для не менее 1-го объекта.
- В диаграмме представить не менее 4-х состояний объекта.
- Обязательное использование внутренних активностей (internal activities): Entry, Do, Exit
- Обязательное обозначение событий-переходов.
- Текстовое обоснование предложенного решения.

Лабораторная работа 1. UML. Диаграмма вариантов использования

Цель работы

Применить диаграмму вариантов использования для описания требований к разрабатываемой системе. Ознакомиться с инструментами, позволяющими строить UML диаграммы.

Теоретические сведения

Общая информация. Визуальным моделированием (visual modeling) называется способ представления идей и проблем реального мира с помощью моделей.

Модель — это абстракция, описывающая суть сложной проблемы или структуры без акцента на несущественных деталях, тем самым делая ее более понятной. Разработка программного обеспечения — не исключение. При построении сложной системы строятся ее абстрактные визуальные модели.

В настоящее время в области проектирования информационных систем с успехом применяется визуальное моделирование с помощью унифицированного языка моделирования UML.

С помощью UML можно детально описать систему, начиная разработку с концептуальной модели с ее бизнес-функциями и процессами, а также описать особенности реализации системы, такие как классы программного обеспечения системы, схему базы данных.

Моделирование с помощью UML осуществляется поэтапным построением ряда диаграмм, каждая из которых отражает какую-то часть или сторону системы либо ее замысла.

Основные диаграммы UML:

1) вариантов использования (use case diagram);
2) классов (class diagram);
3) кооперации (collaboration diagram);
4) последовательности (sequence diagram);
5) состояний (statechart diagram);
6) деятельности (activity diagram);

7) компонентов (component diagram);

8) развертывания (deployment diagram).

Для сопровождения процесса построения, анализа и документирования модели, а также проверки модели и генерации программных кодов разработчики используют специально для этих целей созданные CASE-инструменты проектирования систем.

CASE (Computer-Aided Software Engineering) — это набор инструментов и методов программной инженерии для проектирования программного обеспечения, цель которого — обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов.

Для выполнения лабораторных работ можно использовать любой графический редактор, поддерживающий нотацию UML, или специализированные CASE средства. Рекомендованным инструментом является свободное программное средство StarUML.

Перед началом работы в StarUML

Для того, чтобы избежать проблем с автоматическим построением диаграммы взаимодействия на основе диаграмм последовательности, перед началом работы (созданием проекта) требуется установить в качестве используемого в системе разделителя целой и дробной части точку («.»). Изменить эту настройку можно в перейдя в Панель управления/Языки и региональные стандарты/Форматы/Дополнительные параметры/Разделитель целой и дробной части. Такое требование связано с неправильным использованием языковых стандартов разработчиками StarUML.

Создание нового проекта. Основная структурная единица в StarUML — это проект. Проект сохраняется в одном файле в формате XML с расширением «.uml». Проект может содержать одну или несколько моделей. Структура модели (набор рекомендованных диаграмм и их группировка) определяется выбранной методологией. StarUML поддерживает несколько распространённых методологий: 4+1 View Model, Методология Rational, UML Component и подход StarUML (*рис. 11.1*). Воспользуемся методологией Rational.

Главное окно StarUML

В верхней части окна расположено главное меню, кнопки быстрого доступа. Слева расположена панель элементов (Toolbox) с изображениями элементов диаграммы. Набор иснтрументов зависит от выбранной диаграммы. В центре находится рабочее поле диаграммы — область для создания диаграммы. Справа находится инспектор модели, на котором можно найти вкладки навигатора модели (Model Explorer), навигатора диаграмм (Diagram explorer), окно редактора свойств (Properties), окно документирования элементов модели (Documentation) и редактор вложений (Attachments) (*рис. 11.2*).

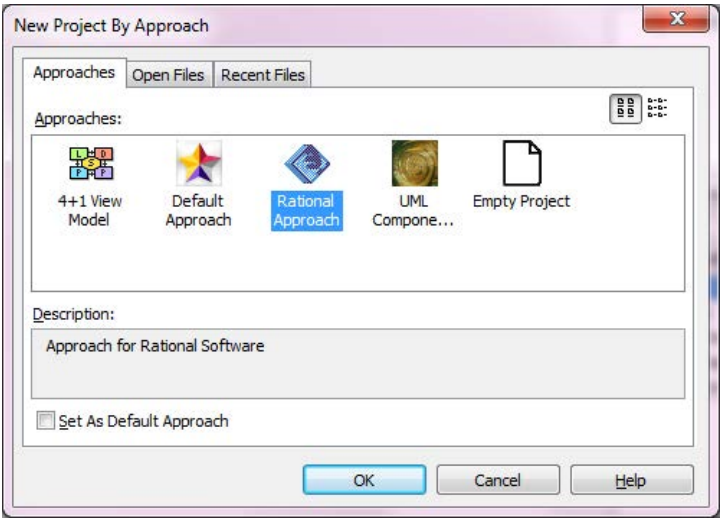

Рис. 11.1. Окно выбора типа проекта

В зависимости от выбранного подхода на навигаторе модели будут отображены различные пакеты представлений модели. Каждый пакет представления будет содержать элементы моделей и диаграмм, которые мы создадим.

Подход Rational предлагает нам следующие представления модели:

– Use Case View — представление требований к системе, описывает, что система должна делать;

– Logical View — логическое представление системы, описывает, как система должна быть построена;

– Component View — представление реализации, описывает зависимость между программными компонентами;

– Deployment View — представление развертывания, описывает аппаратные элементы, устройства и программные компоненты и размещение компонентов.

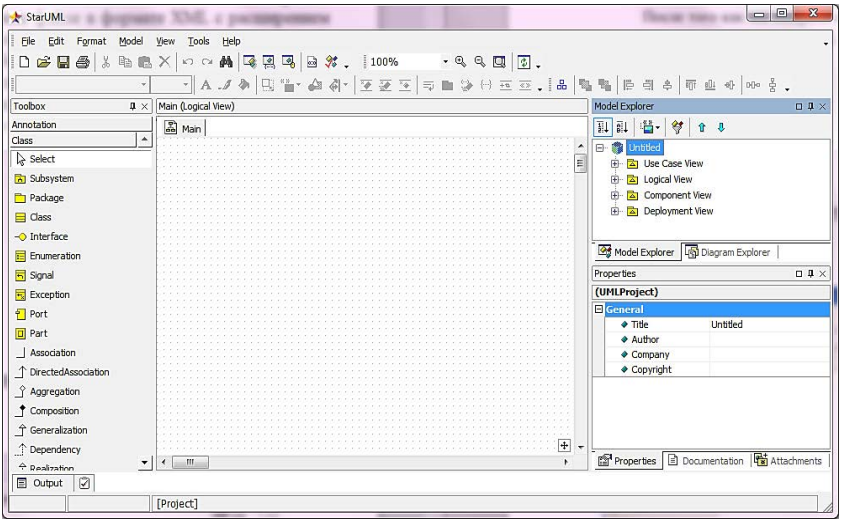

Рис. 11.2. Главное окно StarUML

Диаграмма вариантов использования. Поведение системы (функции, которые она выполняет) описывают с помощью функциональной модели, которая отображает варианты использования (use cases, системные прецеденты), системное окружение (действующих лиц, актеров, actors) и связи между ними.

Диаграмма вариантов использования — это диаграмма, на которой изображаются отношения между актерами и вариантами использования.

С помощью этой диаграммы можно:

– определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;

– сформулировать общие требования к функциональному поведению проектируемой системы;

– разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;

– подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Диаграмма вариантов использования (прецедентов) представляет собой граф, в вершинах которого расположены актеры или прецеденты, связи между вершинами — это разного вида отношения.

Актером (действующее лицо, actor) называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая служит источником воздействия на моделируемую систему так, как определит разработчик.

На диаграммах вариантов использования актер изображается в виде человечка, под которым записывается его имя (*рис. 11.3*).

Преподаватель

Рис. 11. 3. Действующее лицо

Вариант использования (прецедент, use case) — описание множества последовательности действий (включая варианты), выполняемых системой для того, чтобы актер мог получить определенный результат.

Каждый вариант использования должен быть независимым в том смысле, что если он всегда выполняется совместно с другим вариантом использования, то, скорее всего, это один прецедент, а не два, либо они связаны отношением включения или расширения, о чем речь пойдет позже.

Актер должен получить некоторый результат по завершению исполнения прецедента. Например, после исполнения прецедента в системе произойдут некоторые изменения: появятся новые данные, изменится поведение.

Каждый вариант использования описывает законченный процесс, переводящий систему из состояния готовности в новое состояние готовности. Вариант использования не может

прерываться — он должен исполняться от начала до конца. По завершению одного варианта использования может начаться любой другой вариант использования (т. е. эта диаграмма не показывает порядок действий, только набор возможных прецедентов).

Прецедент описывает, что делает система, но никак не уточняет, как она это делает.

На диаграмме прецедент изображается в виде эллипса. Имя прецедента может состоять из нескольких слов и знаков препинания (за исключением двоеточия), как правило, имя выбирают в виде словосочетания или глагольного выражения (*рис. 11.4*).

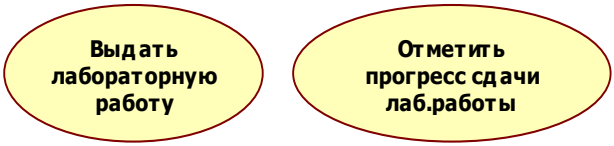

Рис. 11.4. Варианты использования

Одним из наиболее важных (и дорогостоящих) этапов проектирования информационных систем является этап определения требований к системе. Если требования заказчика информационной системы разработчиками будут определены не корректно, то в итоге заказчик может получить совсем не ту систему, которую он ожидал.

Моделирование прецедентов и актеров помогает нам лучше понять требования, предъявляемые к системе, и согласовать их с заказчиком с помощью демонстрации и обсуждения диаграммы прецедентов. Прецеденты и актеры — это отражение требований к системе, они показывают, кто и для чего будет использовать будущую систему.

Рассмотрим пример. Требуется разработать комплекс «Автоматизированное рабочее место преподавателя». В задачи комплекса входит обеспечение следующих возможностей:

– составление и редактирования графика лабораторных работ;

– отмечание успешных и проваленных попыток сдачи лабораторных работ студентами;

– указывать темы РГР, курсовых работ и проектов;

– следить за текущей успеваемостью в разрезе группы, курса (для заведующего кафедрой) и факультета (для декана);

– отмечать пропуски на лабораторных занятиях и лекциях;

– запрашивать списки групп из автоматизированной системы «Деканат».

Проанализировав постановку задачи, приходим к выводу, что будет полезно выделить следующие действующие лица:

– преподаватель — преподаватель, ведущий практические занятия и лабораторные;

– лектор — преподаватель, ведущий лекции;

– руководитель — заведующие кафедрами, деканы;

– ассистент — ответственный за ввод данных в систему;

– АС «Деканат» — сервис автоматизированной системы «Деканат».

Обратите внимание, что актер — это роль, а не конкретный представитель этой роли. Например, в разное время заведующий кафедрой может выступать в роли лектора, преподавателя, ведущего практические занятия или руководителя.

Теперь выделим варианты использования на основе перечисленных функциональных требований:

– работа с графиком лабораторных работ;

– отмечание успеваемости;

– указание тем РГР, курсовых работ и проектов;

– отмечание посещения лекций;

– отмечание посещения лаб.работ;

– загрузка списков групп;

– создание лабораторных работ;

– создание РГР, курсовых работ и проектов;

– построение отчётов успеваемости и посещаемости.

Отношения

Связи и взаимоотношения, существующие между элементами модели, в UML описываются с помощью отношений, изображаемых на диаграммах.

Отношение (relationship) — это семантическая связь между отдельными элементами модели.

Между актерами и прецедентами диаграммы вариантов использования могут существовать отношения, показывающие, что экземпляры действующих лиц и вариантов использования взаимодействуют друг с другом.

Действующие лица могут находиться в отношениях с несколькими прецедентами и между собой, равно как и прецеденты могут быть связаны между собой отношениями особого рода.

Всего на диаграммах вариантов использования допустимыми являются следующие типы отношений:

– обобщения (generalization relationship) — отношение между двумя актерами, либо двумя вариантами использования;

– ассоциации (association relationship) — отношение между актером и вариантом использования;

– включения (include relationship) — отношения между вариантами использования;

– расширения (extend relationship) — отношения между вариантами использования.

Ассоциация — это структурное отношение, показывающее, что объект неким образом связан с другим объектом. Отношение этого типа используется не только на диаграммах прецедентов, но и на других диаграммах. Ассоциативное отношение может быть направленным. В этом случае направление связи показывает, кто является инициатором (актер или разрабатываемая система). Например, если отношение направленно от актера к прецеденту, это означает, что актер инициирует выполнение прецедента.

Обобщение (Generalization) — это отношение между общей сущностью и ее конкретным воплощением. На диаграммах обобщение обозначается стрелкой с не закрашенным треугольником на конце, направленной от частного элемента к общему. По смыслу это отношение очень похоже на отношение «является родителем» в ООП.

Отношения включения и расширения связаны с дополнительной информацией, сопутствующей диаграмме вариантов использования.

Для более подробного (больше, чем имя) описания элементов модели можно использовать следующие подходы:

– краткое текстовое описание (в поле Documentation);

– связанные диаграммы (контекстное меню Add diagram) — диаграммы, более подробно описывающие конкретный элемент модели (например, алгоритм действий в конкретном варианте использования);

– вложения (Attachments) — прикреплённые документы (например, *.docx содержащие подробные описания).

Для описания элементов диаграммы вариантов использования часто используют:

– связанные диаграммы последовательностей и кооперацией;

– вложенные документы — сценарии использования.

Сценарии применяются для текстового описания потоков событий. Поток событий — это определенная последовательность действий, которая описывает действия актеров и поведение моделируемой системы в форме текста.

Обычно потоки событий записываются в виде пронумерованной последовательности действий на естественном языке (хотя иногда применяют и псевдокод).

Каждый сценарий состоит из описания основного потока событий, его вариантов и исключительного потока.

Основной поток событий описывает ожидаемый вариант происходящих событий (например, все данные оказались правильными, файлы были найдены, места на сервере хватило).

Альтернативные потоки описывают различные варианты отклонения от обычного потока (например, выборка списка товара по дате, а не виду) или необязательные шаги (выбор цвета товара дополнительно к цене).

Исключительный поток показывает поведение системы в случае возникновения ошибок.

Если описание потока событий занимает много места, оно теряет выразительность. В таких случаях применяют следующие приёмы.

Если часть потока событий повторяется в нескольких вариантах использования, эту часть выносят в отдельный вариант использования. Прецеденты, из которых удалили эту часть, соединяют с новым вариантом использования отношением «включает». В описании включающих прецедентов указывается, что часть потока находится в другом прецеденте.

Необязательные этапы потока событий можно вынести в отдельный прецедент. Тогда этот новый прецедент будет «расширять» оригинальный. В описании оригинального потока указывается точка расширения — место в котором могут «подцепляться» прецеденты для расширения, а в расширяющих прецедентах указывается к какой именно точке расширения они относятся.

Ход работы

Для того, чтобы перейти к созданию диаграммы вариантов использования разверните папку Use Case View и откройте диаграмму Main с помощью двойного щелчка. Обратите внимание, что панель инструментов теперь содержит инструменты для создания диаграммы вариантов использования (*рис. 11.5*):

– Actor — создание новых актеров;

– Association и Directed Association — создание ассоциаций и направленных ассоциаций;

– Generalization — обобщение;

– Include и Extend — включение (направлено от включающего к включаемому) и расширение (направлено от расширяющего к расширяемому);

– System Boundary — указание границ системы.

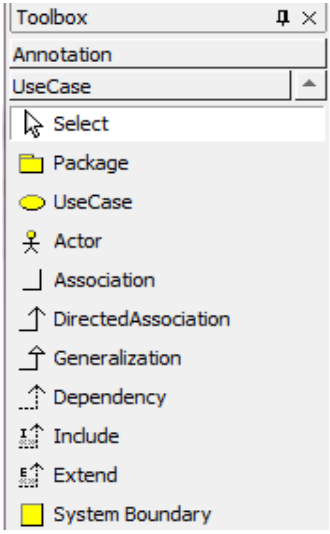

Рис. 11.5. Панель инструментов
для работы с Use Case диаграммой

Пример диаграммы вариантов использования приведён на *рис. 11.6*.

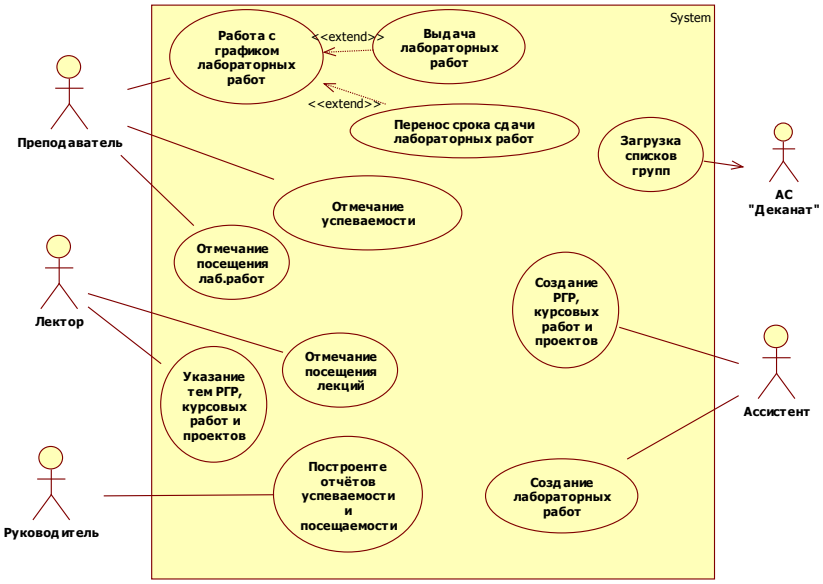

Рис. 11.6. Диаграмма вариантов использования «АРМ Преподавателя»

Фрагмент сценария к варианту использования «Работа с графиком лабораторных работ».

1. Пользователь выбирает дисциплину.
2. Система показывает график выдачи и срока сдачи всех лабораторных работ по выбранному предмету.
3. Точка расширения «Работа с лабораторными работами».
4. Пользователь нажимает «вернуться назад».
5. Система показывает экран работы с дисциплиной.
6. Варианты использования «Перенос срока сдачи лабораторных работ» и «Выдача лабораторных работ» расширяют этот вариант использования в точке «Работа с лабораторными работами», что должно быть указано в их сценарии.
7. Точка расширения «Работа с лабораторными работами».
8. Сроки выдачи и сдачи лабораторных работ показаны в виде прямоугольников на оси времени.
9. Пользователь перетягивает один из углов прямоугольника, изображающего срок сдачи работы.
10. Срок сдачи лабораторной работы изменяется на графике.

11. Система показывает экран работы с дисциплиной.
12. Пользователь выбирает пункт «График лабораторных работ».

Контрольные вопросы

1. Что называют Визуальным моделированием?
2. Что такое модель?
3. Перечислите основные диаграммы UML.
4. Что такое StarUML?
5. Для чего применяются диаграммы использования?

Лабораторная работа 2. UML. Диаграмма деятельности

Цель работы

Применить диаграмму деятельности для описания поведения разрабатываемой системы. Ознакомиться с инструментами, позволяющими строить диаграммы деятельности.

Теоретические сведения

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности. Диаграмма деятельности — это своеобразная блок-схема, которая описывает последовательность выполнения операций во времени. Их можно использовать для моделирования динамических аспектов поведения системы.

Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии.

В диаграммах деятельности используются пиктограммы «действие», «переход», «выбор» и «линии синхронизации». В языке UML действие изображается в виде прямоугольника с закругленными углами, переходы — в виде направленных стрелок, элементы выбора — в виде ромбов, линии синхронизации — в виде горизонтальных и вертикальных линий.

Состояние действия является специальным случаем состояния с некоторым входным действием и, по крайней мере, одним выходящим из состояния переходом. Когда действие или деятельность в некотором состоянии завершается, поток управления сразу переходит в следующее состояние действия или деятельности. Для описания этого потока используются переходы, показывающие путь из одного состояния действия или деятельности в другое.

Простые последовательные переходы встречаются наиболее часто, но их одних недостаточно для моделирования любого потока управления. Как и в блок-схеме, вы можете включить в модель выбор, который описывает различные пути выполнения в зависимости от значения некоторого булевского выражения.

Простые и ветвящиеся последовательные переходы в диаграммах деятельности используются чаще всего. Однако можно встретить и параллельные потоки, и это особенно характерно для моделирования бизнес-процессов.

В UML для обозначения разделения и слияния таких параллельных потоков выполнения используются линии синхронизации, которые рисуется в виде жирной вертикальной или горизонтальной линии.

При моделировании течения бизнес-процессов иногда бывает полезно разбить состояния деятельности на диаграммах деятельности на группы, каждая из которых представляет отдел компании, отвечающий за ту или иную работу.

В UML такие группы называются дорожками, поскольку визуально каждая группа отделяется от соседних вертикальной чертой, как плавательные дорожки в бассейне. Каждой присутствующей на диаграмме дорожке присваивается уникальное имя. Каждая дорожка представляет сферу ответственности за часть всей работы, изображенной на диаграмме, и может быть реализована одним или несколькими классами.

Ход работы

Для создания диаграммы деятельности воспользуйтесь контекстным меню представления системы в навигаторе модели. Основными элементами диаграммы являются состояния действия и переходы. Как и в диаграмме состояний выделяют несколько видов состояний действия (*рис. 15.1*).

Простое состояние деятельности (ActionState) — содержит название выполняемого действия.

Ссылочное состояние деятельности (SubactivityState) — изображает состояние, детальное описание которого вынесено в отдельную диаграмму.

Начальное состояние всегда присутствуют на диаграмме деятельности. Конечное состояние может отсутствовать, если описывается бесконечный алгоритм.

Узел принятия решения используется для изображения условных переходов (*рис. 15.2*).

Узел синхронизации используется для изображения нескольких потоков управления (*рис. 15.3*).

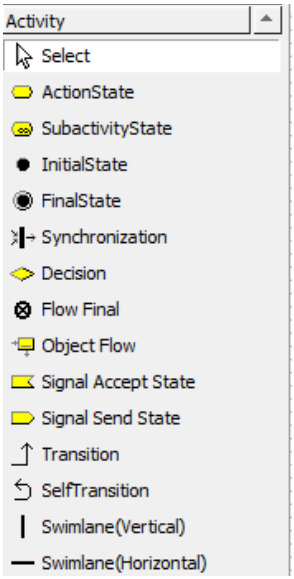

Рис. 15.1. Панель инструментов диаграммы деятельности

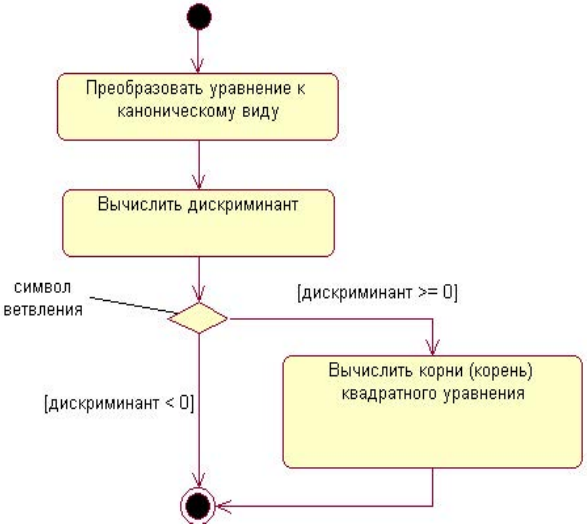

Рис. 15.2. Пример узла принятия решения

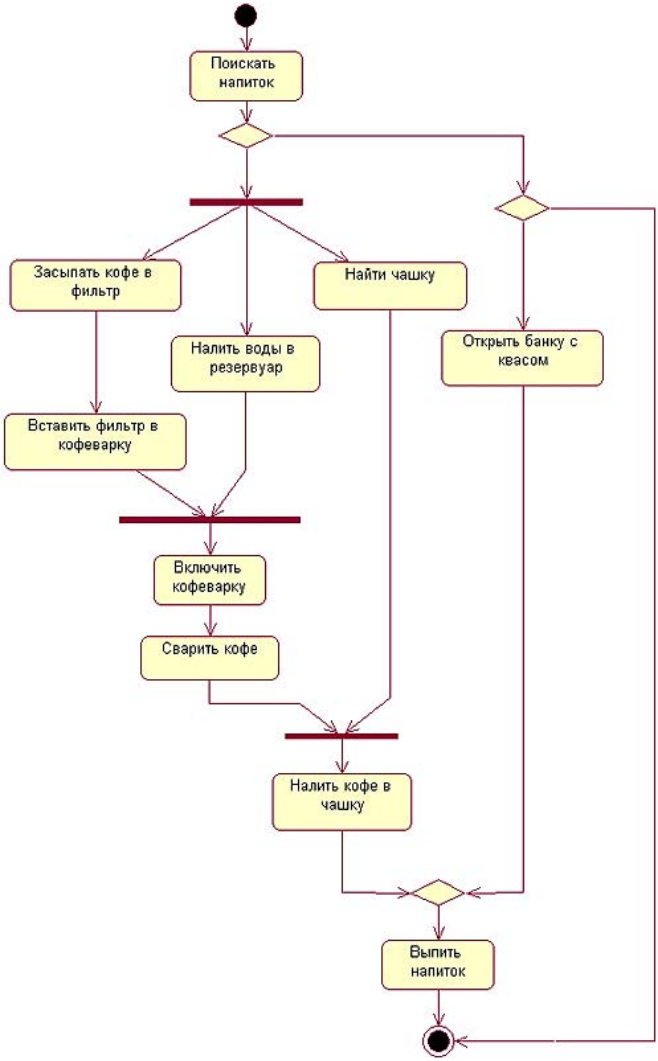

Рис. 15.3. Пример ветвления на диаграмме деятельности

Поток объектов показывает связь между состояниями обрабатываемых объектов и этапов алгоритма. Состояния принятия и отправки сигналов позволяют показать асинхронные взаимодействия.

Дорожки позволяют показать, каким объектом выполняются те или иные шаги алгоритма (*рис. 15.4*).

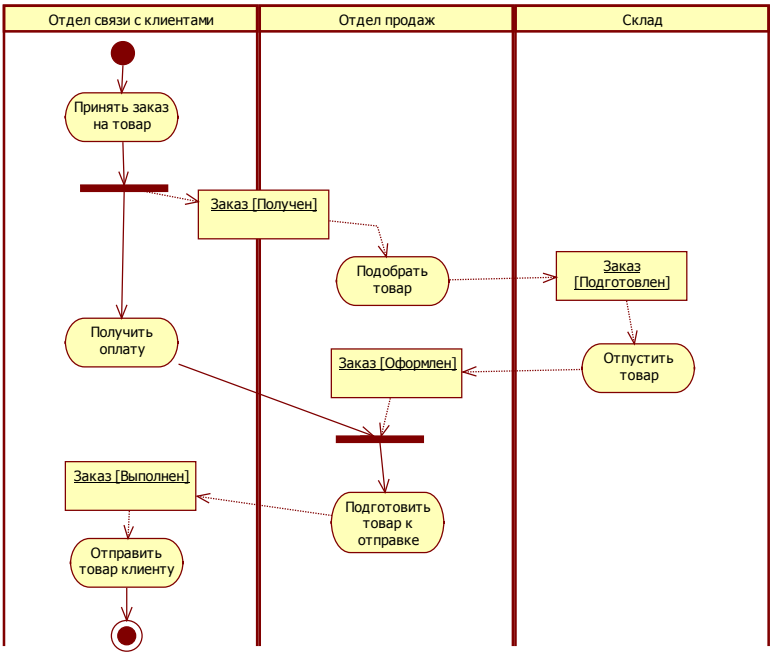

Рис. 15.4. Пример диаграммы деятельности с дорожками

Контрольные вопросы

1. Перечислите основные диаграммы UML.
2. Что такое StarUML?
3. Для чего применяются диаграммы деятельности?
4. Перечислите основные компоненты диаграммы деятельности.
5. В чем заключаются отличия статических и динамических UML диаграмм?

Лабораторная работа 3. UML. Диаграмма последовательности

Цель работы

Применить диаграмму последовательности для описания разрабатываемой системы. Ознакомиться с инструментами, позволяющими строить диаграммы последовательности.

Теоретические сведения

Одной из характерных особенностей систем различной природы и назначения является взаимодействие между собой отдельных элементов, из которых образованы эти системы.

Различные составные элементы систем не существуют изолированно, а оказывают определенное влияние друг на друга, что и отличает систему как целостное образование от простой совокупности элементов.

В языке UML взаимодействие элементов рассматривается в информационном аспекте их коммуникации, т. е. взаимодействующие объекты обмениваются между собой некоторой информацией. При этом информация принимает форму законченных сообщений.

Хотя сообщение и имеет информационное содержание, оно приобретает дополнительное свойство оказывать направленное влияние на своего получателя. Это хорошо согласуется с распространённым взглядом на ООП, когда любые виды информационного взаимодействия между элементами системы сведётся к отправке и приему сообщений между ними.

Диаграмма последовательности — это диаграмма взаимодействия, которая выделяет упорядочение сообщений по времени.

Ход работы

Создание диаграммы последовательности в StarUML

Для того, чтобы добавить диаграмму последовательности, воспользуйтесь пунктом добавить диаграмму контекстного меню логического представления в навигаторе модели.

Основные элементы диаграммы последовательности

Основными элементами последовательностей являются объекты с линиями жизни и сообщения (*рис. 12.1*).

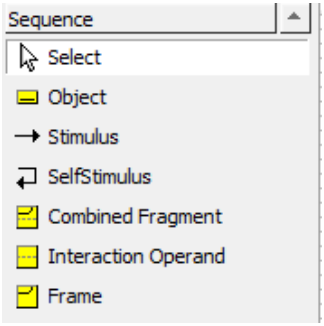

Рис. 12.1. Панель инструментов диаграммы последовательностей

На диаграмме последовательности изображаются исключительно те объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами. Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет два измерения. Одно — слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии.

Крайним слева на диаграмме изображается объект, который является инициатором взаимодействия. Правее изображается другой объект, который непосредственно взаимодействует с первым. Таким образом, все объекты на диаграмме последовательности образуют некоторый порядок, определяемый степенью активности этих объектов при взаимодействии друг с другом.

Второе измерение диаграммы последовательности — вертикальная временная ось, направленная сверху вниз. Начальному моменту времени соответствует самая верхняя часть диаграммы. При этом взаимодействия объектов реализуются посредством сообщений, которые посылаются одними объектами другим. Сообщения изображаются в виде горизонтальных стрелок

с именем сообщения и также образуют порядок по времени своего возникновения. Сообщения, расположенные на диаграмме последовательности выше, инициируются раньше тех, которые расположены ниже. При этом масштаб на оси времени не указывается, поскольку диаграмма последовательности моделирует лишь временную упорядоченность взаимодействий типа «раньше-позже».

Линия жизни объекта изображается пунктирной вертикальной линией, ассоциированной с каждым объектом на диаграмме последовательности. Линия жизни служит для обозначения периода времени, в течение которого объект существует в системе и, следовательно, может потенциально участвовать во всех ее взаимодействиях. Если объект существует в системе постоянно, то и его линия жизни должна продолжаться по всей плоскости диаграммы последовательности от самой верхней ее части до самой нижней

Отдельные объекты, исполнив свою роль в системе, могут быть уничтожены, чтобы освободить занимаемые ими ресурсы. Для таких объектов линия жизни обрывается в момент его уничтожения. Для обозначения момента уничтожения объекта в языке UML используется специальный символ — косой крест на конце линии жизни объекта. Ниже этого символа пунктирная линия не изображается, поскольку соответствующего объекта в системе уже нет, и этот объект должен быть исключен из всех последующих взаимодействий.

Не обязательно создавать все объекты в начальный момент времени. Отдельные объекты в системе могут создаваться по мере необходимости. В этом случае прямоугольник такого объекта изображается не в верхней части диаграммы последовательности, а в той ее части, которая соответствует моменту создания объекта. При этом прямоугольник объекта вертикально располагается в том месте диаграммы, которое по оси времени совпадает с моментом его возникновения в системе. Объект обязательно создается со своей линией жизни и, возможно, с фокусом управления.

В StarUML каждому объекту можно задать имя, выбрать класс из уже созданных в модели (свойство Classifier), указать, что элемент представляет совокупность объектов (IsMultiInstance). Все эти свойства задаются с помощью навигатора свойств объекта (*рис. 12.2*).

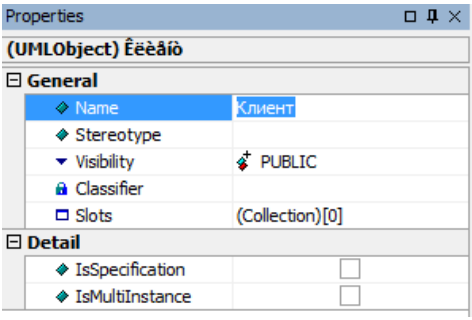

Рис. 12.2. Навигатор свойств объекта

Обратите внимание, что выбрать класс объекта (Classifier) можно только из уже созданных в модели классов (*рис. 12.3*), например, на диаграмме классов.

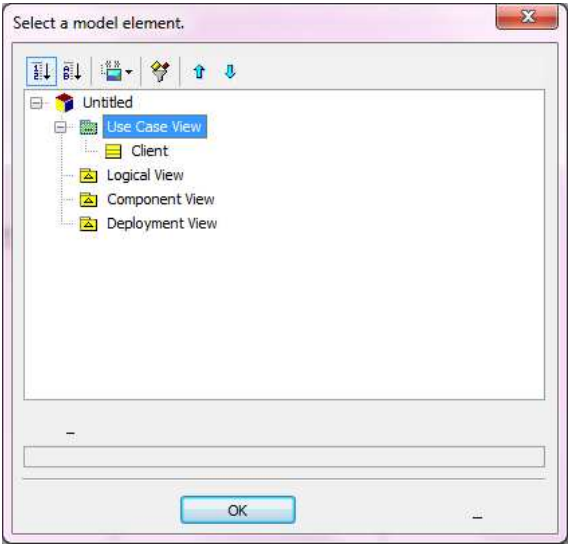

Рис. 12.3. Выбор существующего класса для объекта

Фокус управления. В процессе функционирования объектно-ориентированных систем одни объекты могут находиться в активном состоянии, непосредственно выполняя определенные действия или в состоянии пассивного ожидания сообщений от других объектов. Чтобы явно выделить подобную активность объектов, в языке UML применяется специальное понятие, получившее название фокуса управления. Фокус управления

изображается в форме вытянутого узкого прямоугольника, верхняя сторона которого обозначает начало получения фокуса управления объекта (начало активности), а ее нижняя сторона — окончание фокуса управления (окончание активности). Этот прямоугольник располагается ниже обозначения соответствующего объекта и может заменять его линию жизни, если на всем ее протяжении он является активным.

Периоды активности объекта могут чередоваться с периодами его пассивности или ожидания. В этом случае у такого объекта имеются несколько фокусов управления.

Важно понимать, что получить фокус управления может только существующий объект, у которого в этот момент имеется линия жизни. Если же некоторый объект был уничтожен, то вновь возникнуть в системе он уже не может. Вместо него лишь может быть создан другой экземпляр этого же класса, который, строго говоря, будет являться другим объектом.

Границы фокус управления на диаграмме можно изменять с помощью перетаскивания. Иногда некоторый объект может инициировать рекурсивное взаимодействие с самим собой. Речь идет о том, что наличие во многих языках программирования специальных средств построения рекурсивных процедур требует визуализации соответствующих понятий в форме графических примитивов. На диаграмме последовательности рекурсия обозначается небольшим прямоугольником, присоединенным к правой стороне фокуса управления того объекта, для которого изображается это рекурсивное взаимодействие.

Сообщения. Рассматриваемые взаимодействия описывается совокупностью сообщений, которыми участвующие в нем объекты обмениваются между собой. Сообщение представляет собой законченный фрагмент информации, который отправляется одним объектом другому. При этом прием сообщения инициирует выполнение определенных действий, направленных на решение отдельной задачи тем объектом, которому это сообщение отправлено.

Таким образом, сообщения не только передают некоторую информацию, но и требуют или предполагают от принимающего объекта выполнения ожидаемых действий. Сообщения могут инициировать выполнение операций объектом соответствующего класса, а параметры этих операций передаются вместе с сообщением. На диаграмме последовательности все сообщения

упорядочены по времени своего возникновения в моделируемой системе.

В языке UML могут встречаться несколько разновидностей сообщений:

1. **Вызов** (CALL) — используется для вызова процедур, выполнения операций или обозначения отдельных вложенных потоков управления. Принимающий сообщение объект получает фокус управления, становясь активным. Этот вид сообщений моделирует синхронные (блокирующие) взаимодействия.

2. **Посылка сигнала** (SEND) — используется для обозначения асинхронных взаимодействий.

3. **Создание и удаление** объекта (CREATE и DESTROY) — сообщения, которые обозначают начало и завершение линии жизни объекта. Используются в том случае, когда объект существует не на всём протяжении рассматриваемого взаимодействия.

4. **Возврата из вызова** (RETURN) — примером может служить простое сообщение о завершении некоторых вычислений без предоставления результата расчетов объекту-клиенту. В процедурных потоках управления эта стрелка может быть опущена, поскольку ее наличие неявно предполагается в конце активизации объекта. В то же время считается, что каждый вызов процедуры имеет свою пару — возврат вызова. Для непроцедурных потоков управления, включая параллельные и асинхронные сообщения, стрелка возврата должна указываться явным образом.

В StarUML тип сообщения задаётся в навигаторе свойств сообщения (*рис. 12.4*).

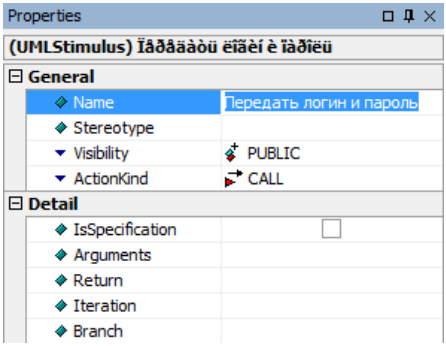

Рис. 12.4. Навигатор свойств сообщения

Циклы и ветвления. Рассмотрим варианты изображения циклов и ветвлений на диаграмме последовательности.

Условное выполнение или повторение отдельных сообщений можно показать с помощью специальных обозначений в надписи сообщения.

Для того, чтобы добавить условие к сообщению, впишите требуемое условие в свойство Branch навигатора свойств сообщения. Для указания количества повторений сообщения используйте свойство Iteration. Обычно повторения записывают в формате «n = 1..10», т. е. повторить десять раз для значений n от 1 до десяти.

Для обозначения условного выполнения группы сообщений или повторения целой последовательности сообщений используются фреймы (*рис. 12.5, рис. 12.6*).

Для изображения цикла используйте инструмент CombinedFragment, указав ему свойство InteractionOperator — «loop». Для изображения условий добавьте в CombinedFragment с InteractionOperator «opt» несколько InteractionOperand.

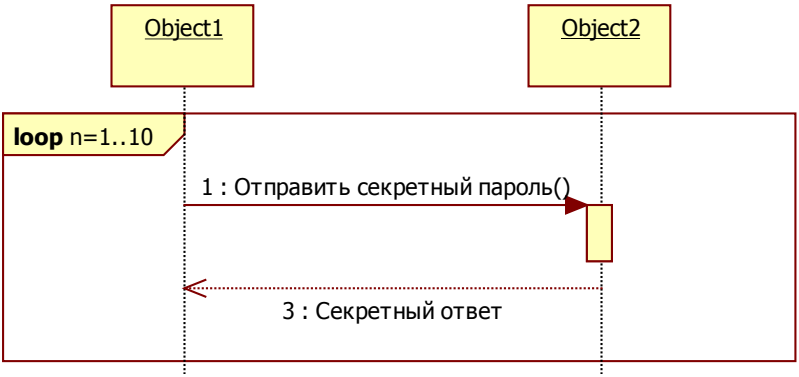

Рис. 12.5. Пример использования фреймов для создания цикла

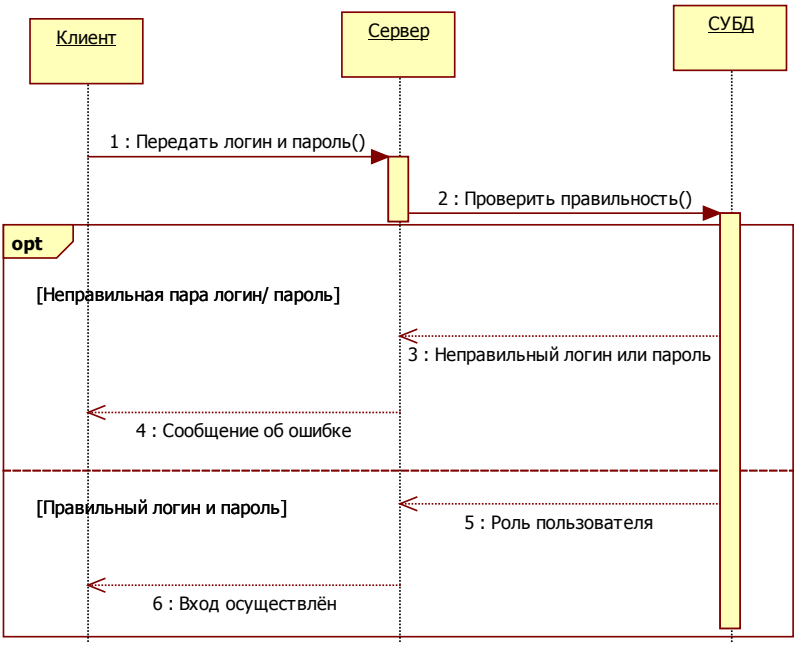

Рис. 12.6. Пример использование фреймов

Контрольные вопросы

1. Перечислите основные диаграммы UML.
2. Что такое StarUML?
3. Для чего применяются диаграммы последовательности?
4. Перечислите основные компоненты диаграммы последовательности.

Лабораторная работа 3. UML. Диаграмма состояний

Цель работы

Применить диаграмму состояний для описания поведения разрабатываемой системы. Ознакомиться с инструментами, позволяющими строить диаграммы состояний.

Теоретические сведения

В отличие от диаграммы классов, представляющей собой логическую модель статического представления моделируемой системы, диаграммы состояний описывают поведение моделируемой системы.

На диаграмме классов изображаются взаимосвязи структурного характера, не зависящие от времени или реакции системы на внешние события. Однако для описания процессов функционирования многих систем и их отдельных подсистем и элементов одного только статического представления оказывается недостаточно.

Рассмотрим простой пример. Любое техническое устройство, такое как телевизор, компьютер, автомобиль, телефонный аппарат в самом общем случае может характеризоваться такими своими состояниями, как «исправен» и «неисправен».

Интуитивно ясно, какой смысл вкладывается в каждое из этих понятий. Более того, использование по назначению данного устройства возможно только тогда, когда оно находится в исправном состоянии. В противном случае необходимо предпринять совершенно конкретные действия по его ремонту и восстановлению работоспособности.

Однако понимание семантики понятия состояния представляет определенные трудности.

Дело в том, что характеристика состояний системы не зависит (или слабо зависит) от логической структуры, зафиксированной в диаграмме классов. Поэтому при рассмотрении состояний системы приходится на время отвлечься от особенностей ее объектной структуры и мыслить другими категориями, образующими динамический контекст поведения моделируемой системы. Поэтому при построении диаграмм

состояний используются специальные понятия, рассмотренные ниже.

Каждая прикладная система характеризуется не только структурой составляющих ее элементов, но и некоторым поведением или функциональностью.

Для общего представления функциональности моделируемой системы предназначены диаграммы вариантов использования, которые на концептуальном уровне описывают поведение системы в целом.

Для более детального представления поведения на логическом уровне используют диаграммы состояний. Эти диаграммы помогают ответить на вопрос: «В процессе какого поведения система обеспечивает необходимую функциональность».

Диаграмма состояний описывает процесс изменения состояний объекта, подсистемы или системы в целом. При этом изменение состояния может быть вызвано воздействиями со стороны других объектов или извне.

Главное предназначение рассматриваемой диаграммы — описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла.

Диаграмма состояний представляет динамическое поведение сущностей на основе спецификации их реакции на восприятие некоторых конкретных событий.

Диаграмма состояний является графом специального вида, который представляет некоторый конечный автомат.

Ход работы

Создание диаграммы состояний в StarUML.

Для того, чтобы добавить диаграмму состояний, воспользуйтесь пунктом добавить диаграмму контекстного меню логического представления в навигаторе модели (*рис. 14.1*).

Основные элементы диаграммы состояний. Основными элементами диаграммы состояний являются состояния и переходы. В StarUML выделяется десять типов состояний и два способа создания переходов (*рис. 14.2*).

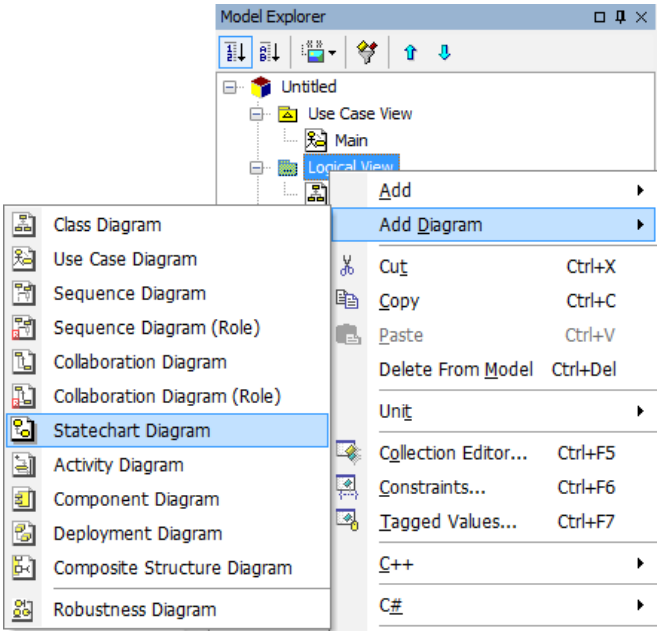

Рис. 14.1. Добавление диаграммы состояний в модель

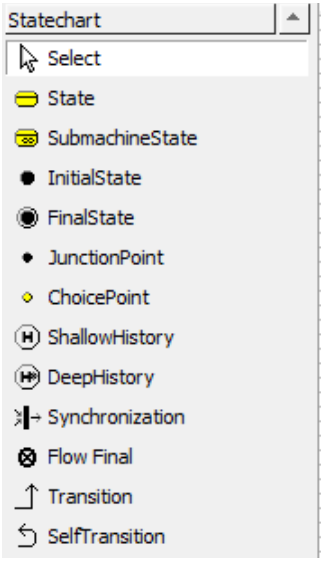

Рис. 14.2. Панель инструментов диаграммы состояний

Длительность нахождения системы в отдельном состоянии существенно превышает время, которое затрачивается на переход из одного состояния в другое. Можно считать, что переход объекта из состояния в состояние происходит мгновенно. Поведение моделируется как последовательное перемещение по графе состояний от вершины к вершине, по связывающим их дугам с учетом их ориентации.

Состояние — период в жизни объекта, на протяжении которого он удовлетворяет какому-то условию, выполняет определенную деятельность или ожидает некоторого события.

Выделяют следующие виды состояний:

– простые;

– составные (ортогональные и не ортогональные);

– специальные;

– ссылочные.

Простое состояние может содержать описание деятельности, входного и выходного действия, список внутренних переходов и отсроченных событий. Входное действие срабатывает перед тем как объект перейдёт в указанное состояние. Выходное действие срабатывает перед тем, как объект покинет состояние. Деятельность совершается в течение всего времени пребывания объекта в рассматриваемом состоянии.

В StarUML для простого состояния можно задать название, действия на входе (EntryActions), деятельность (DoActions), действия на выходе (ExitActions) (*рис. 14.3*).

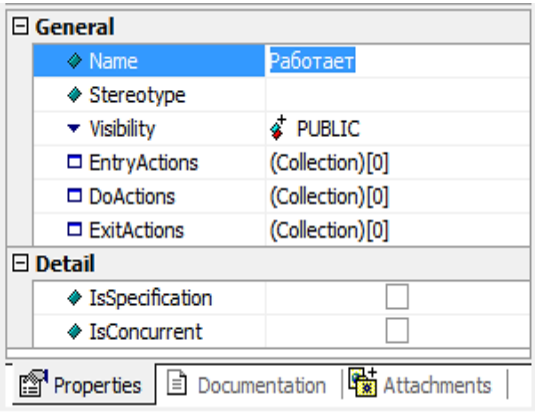

Рис. 14.3. Навигатор свойств состояния

Редактировать эти коллекции можно с помощью редактора коллекций (*рис. 14.4*). Свойства каждого действия можно изменять в навигаторе свойств, предварительно выбрав нужное действие в окне редактора.

Внутрь простого состояния можно добавить дочерние состояния и переходы.

Переходы между состояниями отображаются помеченными стрелками. Каждый переход кроме исходного и целевого состояния может характеризоваться событием перехода, сторожевым условием и действием на переход. Переход, для которого не указано событие перехода, может осуществиться только по завершении выполнения деятельности, указанной в состоянии. Переход, для которого указано событие перехода, срабатывает в случае возникновения события перехода.

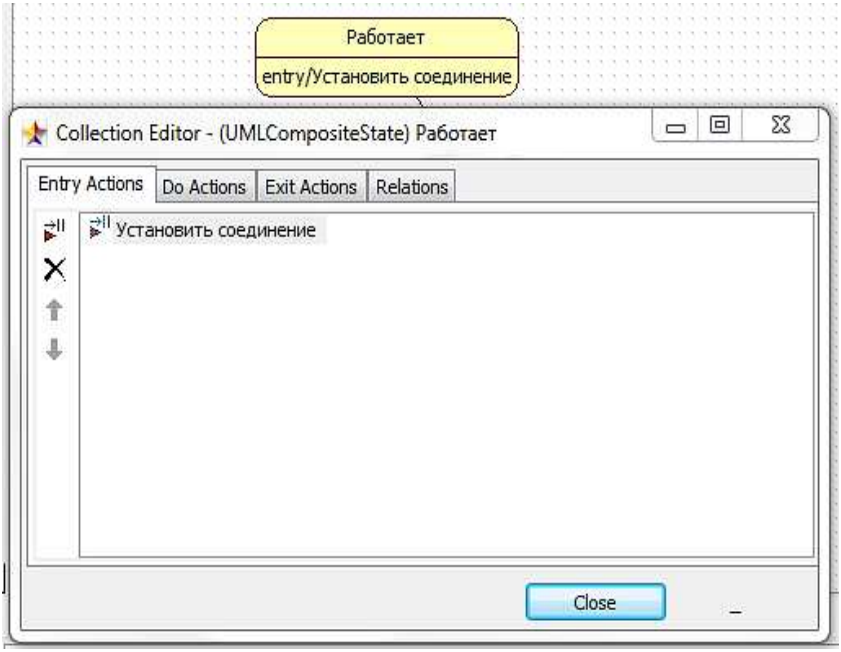

Рис. 14.4. Редактор коллекций. Редактирование входных действий

Когда выполняются указанные выше условия и переход готов сработать, говорят, что переход перешёл в возбуждённое состояние.

Диаграммы состояний используются для спецификации поведения, а значит, не должны содержать двусмысленностей. Если в какой-либо момент времени несколько переходов могут находиться в возбуждённом состоянии, возникает двусмысленность. Решить такую проблему можно указав охранное условие перехода — условие, которое должно быть истинным, чтобы возбуждённый переход сработал.

Обобщив изложенную выше информацию можно сформулировать следующие простые правила:

1. Переход без события может вести из простого состояния только в том случае, когда в состоянии указана деятельность.

2. Если несколько переходов из одного состояния имеют одно и то же событие (или все они не имеют событий), все они должны иметь взаимоисключающие охранные условия.

После того, как определён переход, который должен сработать, выполняется действие на выходе прошлого состояния. Затем выполняется действие при переходе. После этого выполняется входное действие нового состояния.

В StarUML можно задать события (Triggers), действия на переходе (Effects) и охранные условия (GuardCondition) для переходов в навигаторе свойств. Выделяются четыре вида событий:

1. Событие вызова (call event) — это событие, возникающее при вызове метода моделируемого класса.

2. Событие сигнала (signal event) — это событие, возникающее при посылке сигнала (асинхронном взаимодействии).

3. Событие таймера (time event) — это событие, которое возникает, когда истек заданный интервал времени с момента попадания автомата в данное состояние.

4. Событие изменения (change event) — это событие, которое возникает, когда некоторое логическое условие становится истинным, будучи до этого ложным.

Инструмент «переход» (Transition) используется для создания перехода между двумя состояниями, переход из состояния в само себя удобно создавать с помощью инструмента «переход в себя» (SelfTransition).

Ссылочное состояние используется для указания того, что существует отдельная диаграмма, описывающая внутреннее устройство этого состояния.

К специальным состояниям относятся:

– начальное и конечное состояния;

– точки соединения;

– точки выбора;

– историческое и глубокое историческое состояние;

– синхронизация;

– прекращение выполнения.

Начальное и конечное состояние могут отсутствовать на диаграмме, если у объекта нет выделенного начала (например, нет «начального» цвета светофора) или окончания времени жизни.

Несколько переходов из разных состояний, ведущих в одно состояние, можно объединить с помощью точки соединения (*рис. 14.5*).

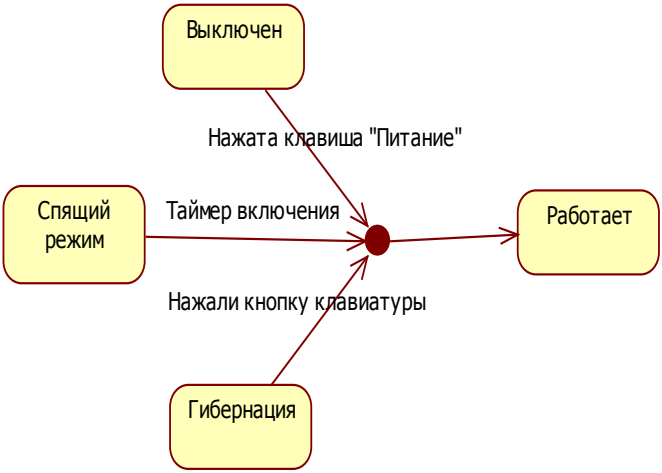

Рис. 14.5. Пример использования точки соединения

Несколько переходов, исходящих из данного состояния и имеющих общее событие перехода, можно объединить в дерево сегментированных переходов с помощью точки выбора (*рис. 14.6*).

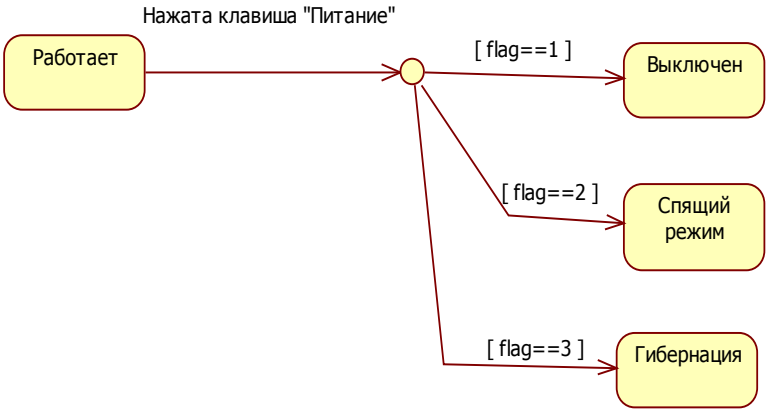

Рис. 14.6. Пример использования точки выбора

Исторические состояния позволяют запомнить, в каком именно вложенном состоянии объект находился перед тем, как он покинул составное состояние. На *рис. 14.7* показано, что электронные часы после включения показывают тот же экран, который они показывали перед выключением.

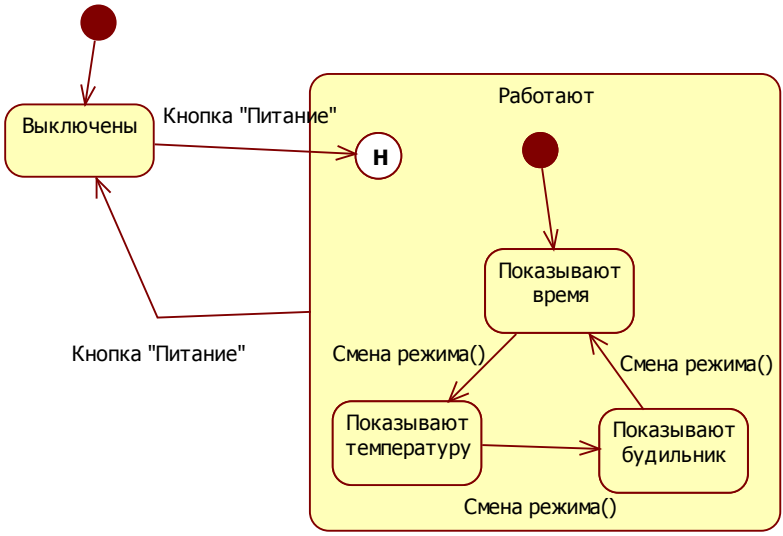

Рис. 14.7. Пример использования исторического состояния

Синхронизация используется для изображения параллельных подавтоматов. Узел «прекращение выполнения» используется для изображения прекращения выполнения одного из параллельных подавтоматов.

Контрольные вопросы

1. Перечислите основные диаграммы UML.
2. Что такое StarUML?
3. Для чего применяются диаграммы состояний?
4. Перечислите основные компоненты диаграммы состояний.
5. В чем заключаются отличия статических и динамических UML диаграмм?

Список литературы

Основная литература

1. Бабич А. В. Введение в UML [Электронный ресурс] / А. В. Бабич — Электрон. текстовые данные. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 198 с. — Режим доступа: http://www.iprbookshop.ru/62809.html — ЭБС «IPRbooks».

2. Батоврин В. К. Системная и программная инженерия. Словарь-справочник [Электронный ресурс]: учебное пособие для вузов / В. К. Батоврин — Электрон. текстовые данные. — Саратов: Профобразование, 2017. — 280 с. — Режим доступа: http://www.iprbookshop.ru/63956.html — ЭБС «IPRbooks».

3. Влацкая И. В. Проектирование и реализация прикладного программного обеспечения [Электронный ресурс]: учебное пособие / Влацкая И. В., Заельская Н. А., Надточий Н. С. — Электрон. текстовые данные. — Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2015. — 119 с. — Режим доступа: http://www.iprbookshop.ru/54145.html — ЭБС «IPRbooks».

4. Грекул В. И. Проектирование информационных систем. Курс лекций [Электронный ресурс]: учебное пособие для студентов вузов, обучающихся по специальностям в области информационных технологий / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина — Электрон. текстовые данные. — Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 303 с. — Режим доступа: http://www.iprbookshop.ru/67376.html — ЭБС «IPRbooks».

5. Долженко А. И. Технологии командной разработки программного обеспечения информационных систем [Электронный ресурс] / А. И. Долженко — Электрон. текстовые данные. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 300 с. — Режим доступа: http://www.iprbookshop.ru/39569.html — ЭБС «IPRbooks».

Дополнительная литература

1. Дэвид Белладжио Стратегия управления конфигурацией программного обеспечения IBM Rational ClearCase [Электронный ресурс] / Дэвид Белладжио, Том Миллиган — Электрон. текстовые данные. — Саратов: Профобразование,

2017. — 382 с. — Режим доступа: http://www.iprbookshop.ru/63958.html — ЭБС «IPRbooks».

2. Кознов Д. В. Введение в программную инженерию [Электронный ресурс] / Д. В. Кознов — Электрон. текстовые данные. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 306 с. — Режим доступа: http://www.iprbookshop.ru/52146.html — ЭБС «IPRbooks».

Справочная литература

1. 12207/FPDAM 1.2 — Software Engineering — Life Cycle Processes // ISO/IEC JTC1/SC7 N2413, Software & System Engineering Secretariat, Canada. — 2001. — 62 p.

2. DTR 9126-2:2001. Software Engineering — Product Quality — Part 2: External Metrics// ISO/ГЕС JTC1/SC7 N2419, Software & System Engineering Secretariat, Canada. — 2001. — 111 p.

3. DTR 9126-3:2001. Software Engineering — Product Quality — Part 3: Internal Metrics //ISO/ffiC JTC1/SC7 N2416, Software & System Engineering Secretariat, Canada. — 2001. — 66 p.

4. DTR 9126-4:2001. Software Engineering — Software Product Quality — Part 4: Quality in Use Metrics // ISO/ГЕС JTC1/SC7 N2430, Software & System Engineering Secretariat, Canada. — 2001. — 70 p.

5. FDIS 9126-1:1999. Software Engineering — Product quality — Part 1: Quality model //ISO/ffiC JTC1/SC7 N2228, Software & System Engineering Secretariat, Canada. — 1999. — 35 p.

6. ISO/IEC 15939:2002. Software Engineering — Software Measurement Process.

7. ГОСТ 19.102-77. Единая система программной документации. Стадии разработки.

8. ГОСТ 19.201-78. Техническое задание. Требования к содержанию и оформлению.

9. ГОСТ 28195-99. Оценка качества программных средств. Общие положения.

10. ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств».