

Лабораторная работа №3

Языки визуального моделирования в аналитических платформах

Цель работы: изучить процесс моделирования и возможности использования языков визуального моделирования.

Задачи работы:

- изучить основы языка UML;
- изучить приведенные в лабораторной работе примеры.

1. Краткая теория

Визуальное моделирование – процесс проектирование структуры базы данных с помощью модель «сущность-связь».

Сущность – это «предмет» рассматриваемой предметной области, который может быть идентифицирован некоторым способом, отличающим его от других «предметов». Конкретные человек, компания или событие являются примерами сущности.

Связь – это некоторое отношение между двумя и более сущностями, отражающее то, как они участвуют в общей деятельности, взаимодействуют друг с другом, совместно используются некоторой другой сущностью и т. д.

Сущности могут быть связаны различными отношениями. Например, две сущности «студент» и «кафедра» связаны отношениям «принадлежит», сущности «студент» и «преподаватель» связаны равноправными отношениями (рисунок 1).

В одном отношении могут участвовать более чем две сущности.

У связи сущность-тип появляются атрибуты, которые аналогичны атрибутам классов в UML, а сами типы сущностей во многом похожи на классы. В примерах, которые будут рассмотрены ниже, используется модель «сущность-связь», представленная в терминах диаграмм классов UML. При этом классы соответствуют типам сущностей, их атрибуты - атрибутам типов, а ассоциации - связям.

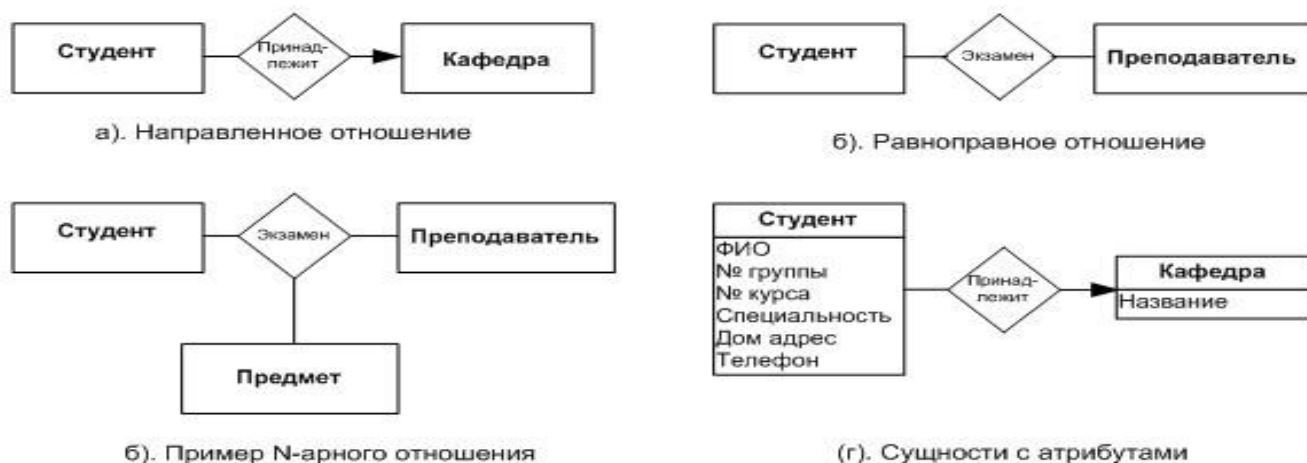


Рисунок 1 – Отношение «принадлежит»

На настоящий момент почти все СУБД поддерживают разработку физической модели схем баз данных с автоматической генерацией конечного кода - Microsoft Visual Studio, Oracle и т. д. Имеются также специальные модельные средства, поддерживающие кроме физической модели также и логическую. Концептуальные модели схем баз данных часто создаются в общих, универсальных UML-средах типа IBM Rational Rose, существует бесплатный аналог Dia, позволяющий импортировать UML-диаграммы в SQL/DDI.

Языки визуального моделирования

Существует множество языков визуального моделирования, такие как UML и BPMN, OCL.

BPMN — система условных обозначений (нотация) для моделирования бизнес-процессов.

BPMN поддерживает лишь набор концепций, необходимых для моделирования бизнес процессов. Моделирование иных аспектов, помимо бизнес процессов, находится вне зоны внимания BPMN. Например, моделирование следующих аспектов не описывается в BPMN:

модель данных;

организационная структура.

Несмотря на то, что BPMN позволяет моделировать потоки данных и потоки сообщений, а также ассоциировать данные с действиями, она не является схемой информационных потоков.

UML — язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

Преимущества UML:

UML объектно-ориентирован, в результате чего методы описания результатов анализа и проектирования семантически близки к методам программирования на современных объектно-ориентированных языках;

UML позволяет описать систему практически со всех возможных точек зрения и разные аспекты поведения системы;

Диаграммы UML сравнительно просты для чтения после достаточно быстрого ознакомления с его синтаксисом;

UML расширяет и позволяет вводить собственные текстовые и графические стереотипы, что способствует его применению не только в сфере программной инженерии;

UML получил широкое распространение и динамично развивается.

Пример.

1. Диаграммы вариантов использования

1.1. Моделирование контекста системы

Моделирование контекста подразумевает выявление актеров, которые находятся вне системы и взаимодействуют с ней. Диаграммы прецедентов нужны на этом этапе для идентификации актеров и семантики их ролей.

Любая система содержит внутри себя какие-либо сущности, в то время как другие сущности остаются за ее пределами. Например, в системе проверки кредитных карточек имеются счета, транзакции и механизмы проверки подлинности. В то же время обладатели кредитных карточек и торговые предприятия находятся вне системы. Сущности внутри системы отвечают за реализацию поведения, которого ожидают сущности, находящиеся снаружи. Сущности, находящиеся вне системы и взаимодействующие с ней, составляют ее контекст. Таким образом, контекстом называется окружение системы.

UML позволяет моделировать контекст с помощью диаграмм прецедентов, в которых внимание акцентируется на окружающих систему актерах. Важно правильно определить актеров, поскольку это позволяет описать класс сущностей, взаимодействующих с системой. Еще важнее определить, что не является актером, так как при этом ограничивается окружение системы: в нем остаются только те элементы, которые участвуют в ее работе.

Моделирование контекста системы состоит из следующих шагов:

Идентифицируйте окружающих систему актеров. Для этого нужно найти группы, которым участие системы требуется для выполнения их задач; группы, которые необходимы для осуществления системой своих функций; группы, взаимодействующие с внешними программными и аппаратными средствами, а также группы, выполняющие вспомогательные функции администрирования и поддержки.

Организуйте похожих актеров с помощью отношений обобщения/специализации.

Например, на рисунке 2 показан контекст системы, работающей с кредитными карточками, где основное внимание уделяется окружающим ее актерам. В первую очередь это Клиенты двух типов (Индивидуальный клиент и Корпоративный клиент), соответствующие ролям, которые играют люди при взаимодействии с системой. В этом контексте показаны и актеры, представляющие другие организации, такие как Торговые предприятия (с ними покупатели совершают карточные транзакции, приобретая вещи или услуги) и Субсидирующие финансовые институты (выполняющие роль клиринговой палаты для карточных счетов). В реальном мире последние два актера, скорее всего, сами будут программными системами.

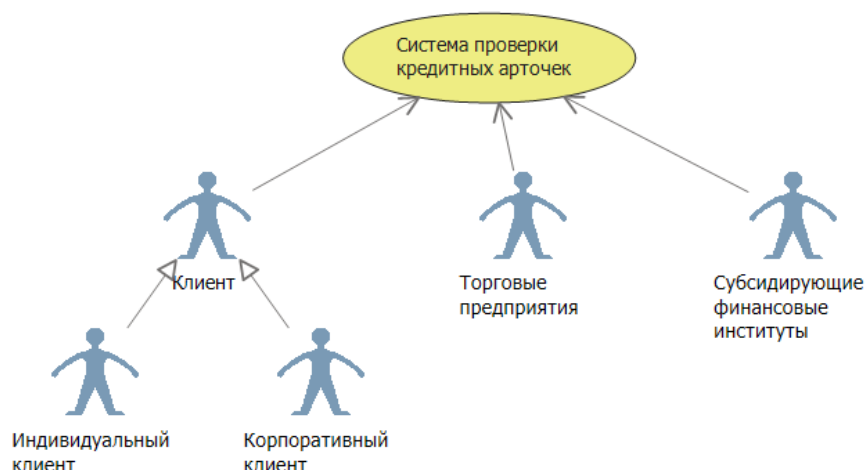


Рисунок 2 – Диаграмма вариантов использования

Аналогично можно моделировать контекст подсистемы. Элемент, который на одном уровне абстракции выглядит как система, часто становится подсистемой на другом, более высоком уровне абстракции. Моделирование контекста подсистемы может пригодиться при построении системы из нескольких взаимосвязанных частей.

Задание. Выбрать тему и закрепить ее у преподавателя. Для выбранной темы построить диаграмму вариантов использования.

1.2. Моделирование требований к системе

Моделирование требований к системе предполагает указание на то, что система должна делать (с точки зрения внешнего наблюдателя), независимо от того, как она должна это делать. Диаграммы прецедентов нужны здесь для специфицирования желаемого поведения системы.

Требование (Requirement) - это особенность проекта, свойство или поведение системы. Приступая к сбору требований, вы как бы описываете условия контракта, заключаемого между системой и сущностями вне нее, в котором декларируется, что система должна делать. При этом, как правило, вас заботит не то, как именно система будет выполнять поставленные задачи, а только то, что она будет делать. Хорошо спроектированная система должна полностью выполнять все требования, причем делать это предсказуемо и надежно. Ее создание начинается с соглашения о том, каково ее назначение, хотя в ходе разработки понимание требований будет постепенно изменяться.

Требования можно выразить по-разному, от неструктурированного текста до выражений на формальном языке (или, например, с помощью примечаний, user story). Большая часть функциональных требований к системе - или даже все они - может быть выражена в виде вариантов использования, в чем помогают диаграммы прецедентов UML.

Моделирование требований к системе производится следующим образом:

Установите контекст системы, идентифицировав окружающих ее актеров.

Для каждого актера рассмотрите поведение, которого он ожидает или требует от системы.

Поименуйте эти общие варианты поведения как прецеденты.

Выделите общее поведение в новые прецеденты, которые будут использоваться другими; выделите вариации поведения в новые прецеденты, расширяющие основные потоки событий.

Смоделируйте эти прецеденты, актеры и отношения между ними на диаграмме прецедентов.

Дополните прецеденты примечаниями, описывающими нефункциональные требования; некоторые из таких примечаний можно присоединить к системе в целом.

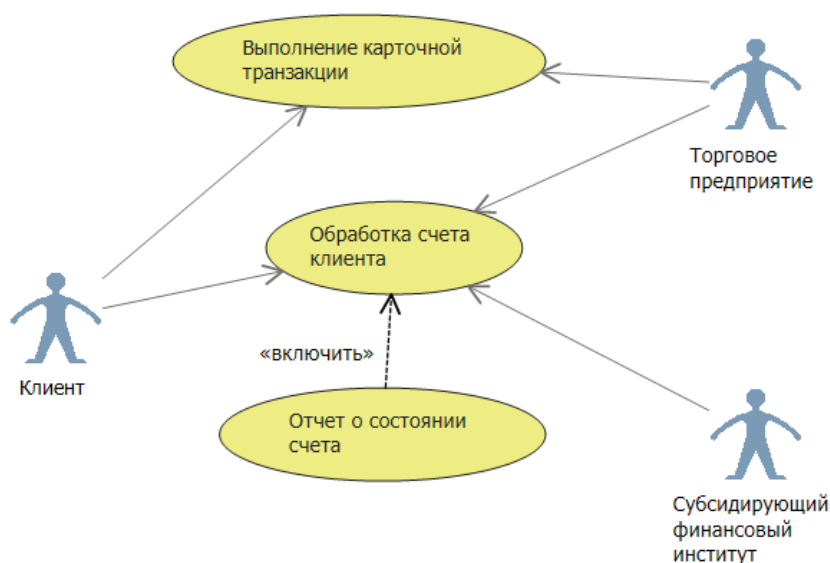


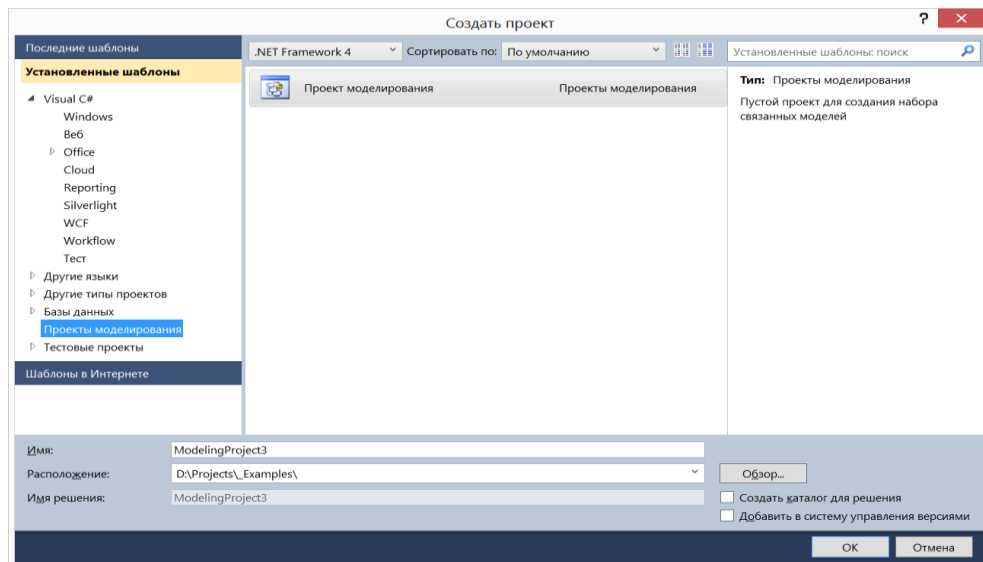
Рисунок 3 - Пример диаграммы вариантов использования для выявления требований

Диаграмму можно расширять, вводя не только функциональные требования, но и нефункциональные. Для повышения наглядности в диаграмму можно добавлять примечания.

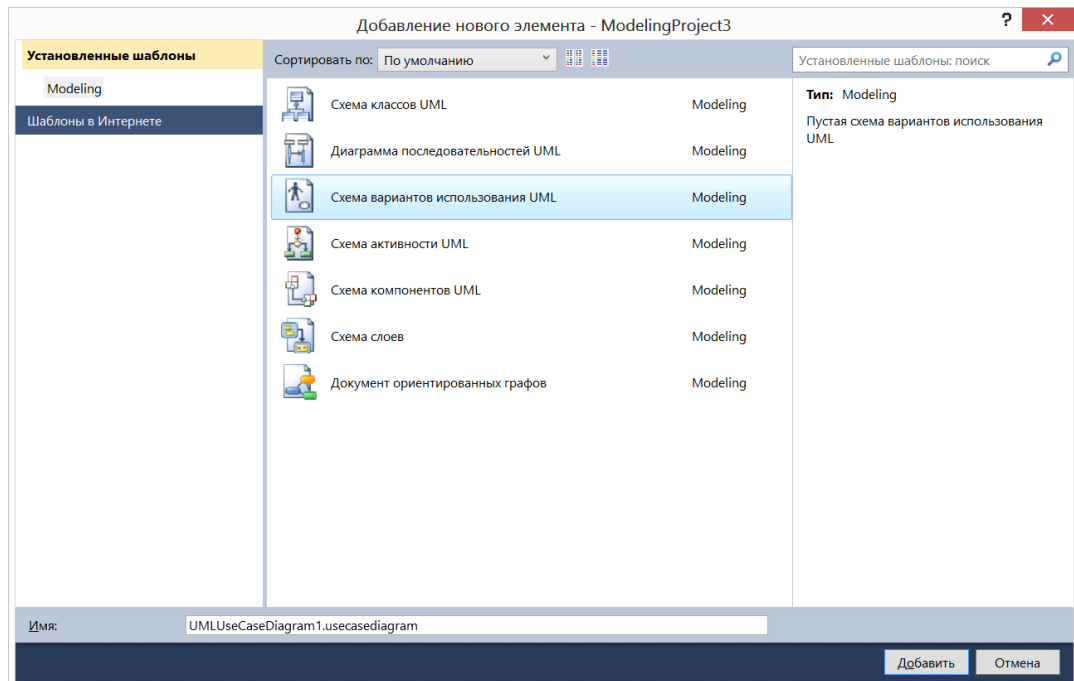
Такая же методика применима и при моделировании требований к подсистеме.

Диаграммы вариантов использования в Visual Studio 2010.

Создать новый проект, используя шаблон «Проекты моделирования»:



Добавить новую диаграмму в проект через Проект - Добавить новый элемент:



Панель инструментов для построения диаграммы использования (обычно слева от рабочей области):

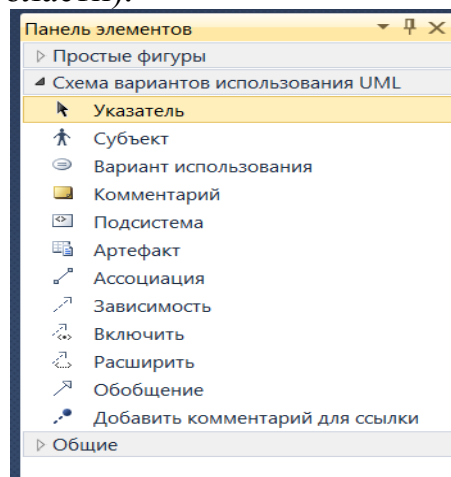


Диаграмма деятельности.

2.1 Диаграмма деятельности (activity diagram) — диаграмма, на которой показано разложение некоторой деятельности на её составные части.

Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграммы деятельности = диаграммы активности

Каждое состояние на диаграмме деятельности соответствует выполнению некой операции, а переход в следующее состояние происходит только после завершения выполнения этой операции.

Диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия или деятельности, а дугами - переходы от одного состояния действия к другому.

Пример построения диаграммы деятельности:

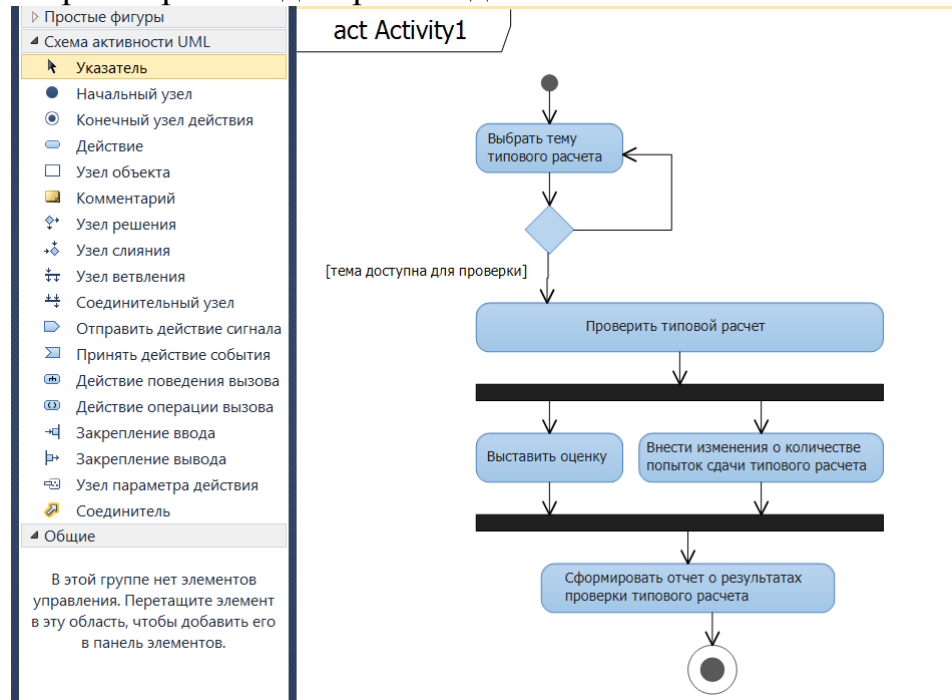


Диаграмма деятельности - это, по существу, блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Результат может привести к изменению состояния системы или возвращению некоторого значения.

Диаграмма деятельности отличается от традиционной блок-схемы более высоким уровнем абстракции;

возможностью представления с помощью диаграмм деятельности управления параллельными потоками наряду с последовательным управлением.

Основными направлениями использования диаграмм деятельности являются

- визуализация особенностей реализации операций классов;
- отображение внутрисистемной точки зрения на прецедент.

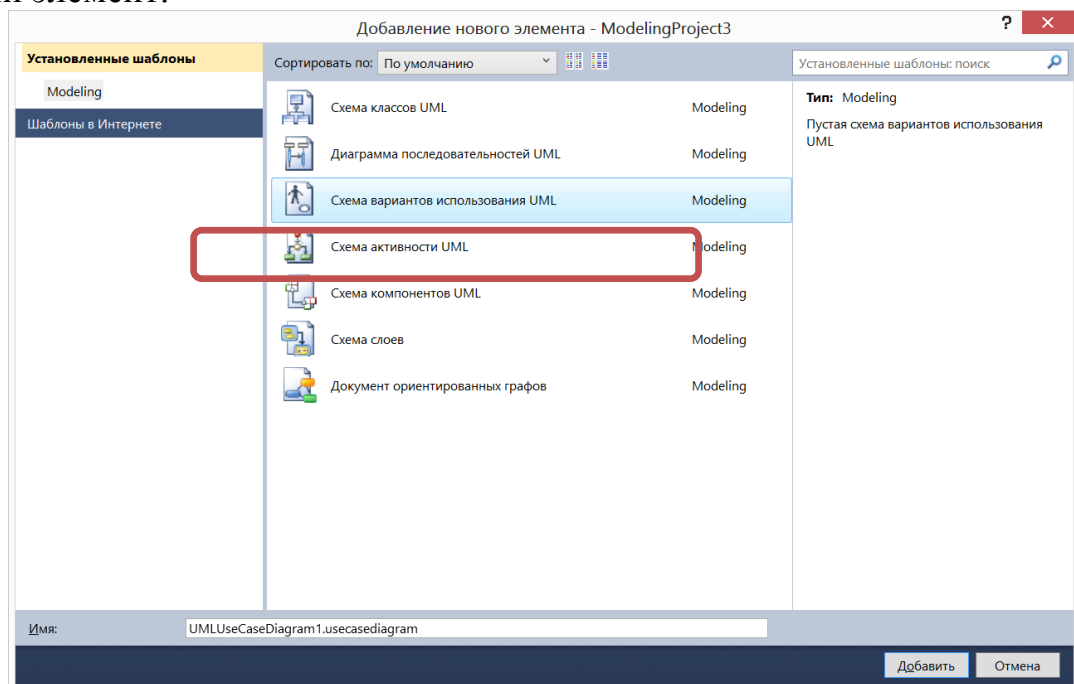
В последнем случае диаграммы деятельности применяют для описания шагов, которые должна предпринять система после того, как инициирован прецедент

Разработка диаграммы деятельности преследует цели:

- детализировать особенности алгоритмической и логической реализации выполняемых системой операций и прецедентов;
- выделить последовательные и параллельные потоки управления;
- подготовить детальную документацию для взаимодействия разработчиков системы с ее заказчиками и проектировщиками.

2.2. Построение диаграммы деятельности

Добавить новую диаграмму в свой проект через Проект - Добавить новый элемент:



Информация о правилах построения схем активности в Visual Studio.
<http://msdn.microsoft.com/ru-ru/library/dd409360.aspx>

Управления жизненным циклом приложений с помощью Visual Studio и Team Foundation Server

Моделирование приложения

Разработка моделей для программного проектирования

Практическое руководство.

Создание проектов и схем для UML-моделирования

Практическое руководство.

Изменение моделей и схем UML

Практическое руководство.

Импорт элементов UML-моделей из XMI-файлов

Схемы слоев: справочные материалы

Схемы слоев: рекомендации

UML-схемы деятельности: справочные материалы

Свойства элементов на схемах деятельности

UML-схемы деятельности: рекомендации

Схемы компонентов UML: справочные материалы

UML-схемы деятельности: справочные материалы

Visual Studio 2012 | Другие версии | Эта тема еще не получила оценку - Оценить эту тему

На *схеме активности* бизнес-процесс или программный процесс показан как рабочий процесс, состоящий из ряда действий. Эти действия могут выполняться людьми, программными компонентами или компьютерами.

Схему активности можно использовать для описания процессов нескольких типов, таких как в следующих примерах.

- Бизнес-процесс или рабочий процесс, в котором участвуют пользователи и система. Дополнительные сведения см. в разделе [Моделирование требований пользователей](#).
- Шаги в тестовом случае. Дополнительные сведения см. в разделе [UML-схемы вариантов использования: правила работы](#).
- Программный протокол, т. е. разрешенная последовательность взаимодействий между компонентами.
- Программный алгоритм.

В этом разделе описаны элементы, которые можно использовать в схемах активности. Более подробные сведения о создании схем активности см. в разделе [UML-схемы деятельности: рекомендации](#). Чтобы создать UML-схему деятельности, в меню **Архитектура** щелкните **Создать схему**. Дополнительные сведения об общих принципах создания схем моделирования см. в разделе [Практическое руководство. Изменение моделей и схем UML](#).

Чтение схем активности

В таблицах в следующих разделах описаны элементы, которые можно использовать на схеме активности, и их основные свойства. Полный список свойств элементов см. в разделе [Свойства элементов на схемах деятельности](#).

Действия и другие элементы, отображаемые на схеме активности, представляют собой одно действие. Эти действия можно просмотреть в обозревателе

Задания для самостоятельной работы

Выбрать и закрепить у преподавателя объект исследования. В качестве объекта исследования должна выступать любая информационная система.

Для выбранной системы построить диаграммы вариантов использования и диаграммы деятельности.