Министерство образования и науки

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет инфокоммуникационных технологий

Кафедра программных систем

Отчет

по лабораторной работе № 5

# «Создание функциональной модели с помощью CASE-средств»

**Система управления проектами**

по дисциплине «Проектирование инфокоммуникационных систем»

**Выполнил: студент группы K4220**

**Проверил: к.т.н., доцент Н.А. Осипов**

Санкт-Петербург

2017

## Цель работы.

Создание функциональной модели с помощью CASE-средств.

## Задачи, решаемые при выполнении работы.

### Разработать основные UML-диаграммы, описывающие функциональность.

### Определить элементы, отображаемые на UML-схемах.

## Объект исследования.

Средства Visual Studio Ultimate для построения основных диаграмм UML.

## Метод экспериментального исследования.

Имитационное визуально ориентированное моделирование.

## Рабочие формулы и исходные данные.

Visual Studio Ultimate предоставляет шаблоны для UML-диаграмм, описывающее функциональность: диаграмма активности и диаграмма последовательности.

## Схема.

### Создание диаграмм в проекте моделирования.

UML-схемы моделирования и схемы слоев могут существовать только внутри проекта моделирования.

Все проекты моделирования содержат общую UML-модель и несколько UML-диаграмм. Каждая диаграмма является представлением части модели. UML-модель содержит все элементы, отображаемые на UML-схемах, и может просматриваться с помощью обозревателя моделей UML.

## Окончательные результаты.

### Диаграмма последовательности

В Visual Studio Ultimate *схема последовательностей* показывает взаимодействие, которое представляет последовательность сообщений между экземплярами классов, компонентами, подсистемами и субъектами. Время увеличивается вниз по диаграмме, на которой показывается переход управления от одного участника к другому.

На рисунке ниже показан пример экземпляров и событий вместо классов и методов. На рисунке могут появляться несколько экземпляров одного и того же типа, а также несколько вхождений одного сообщения.

Существует два вида схем последовательностей:

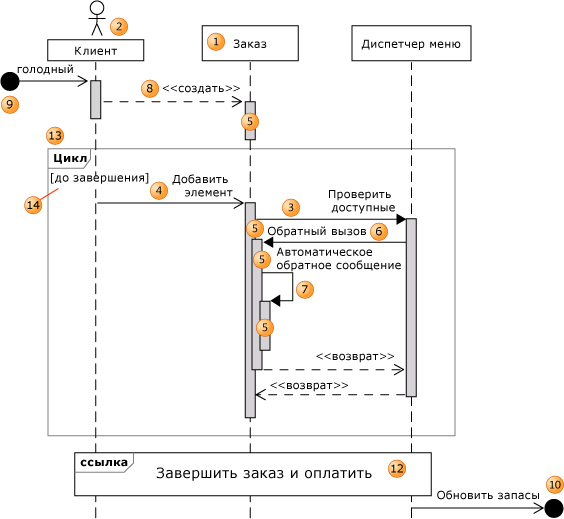
* Схемы последовательностей, основанные на коде могут быть созданы из кода программы .NET и помещены в любой проект.
* Схемы последовательностей UML образуют часть модели UML и существуют только в пределах UML-проекта моделирования.

Два вида схем последовательностей похожи, хотя некоторые из свойств их элементов различаются.

#### Чтение схем последовательностей



В следующей таблице описаны элементы, которые можно видеть на схеме последовательностей.



| **Фигура** | **Элемент** | **Описание** |
| --- | --- | --- |
| 1 | **Линия жизни** | Вертикальная линия, которая представляет последовательность событий, происходящих в участнике во время взаимодействия, когда время направлено вниз по этой линии. |
| 2 | **Субъект** | Участник, являющийся внешним по отношению к разрабатываемой системе. |
| 3 | **Синхронное сообщение** | Отправитель ожидает ответа на синхронное сообщение перед тем, как продолжить. На рисунке показан вызов и возврат. Синхронные сообщения используются для представления обычных вызовов функций внутри программы, а также других видов сообщений, которые применяются аналогичным образом. |
| 4 | **Асинхронное сообщение** | Сообщение, не требующее ответа перед продолжением работы отправителя. Асинхронное сообщение показывает только вызов от отправителя. Используется для представления связи между отдельными потоками или создания нового потока. |
| 5 | **Вхождение выполнения** | Вертикальный затененный прямоугольник, который появляется на линии жизни участника и представляет период, когда участник выполняет операцию.  Выполнение начинается, когда участник получает сообщение. Если инициируемое сообщение было синхронным сообщением, выполнение заканчивается стрелкой возврата к отправителю. |
| 6 | **Сообщение обратного вызова** | Сообщение, возвращающееся обратно участнику, который ожидает возврата из предыдущего вызова. Результирующее вхождение выполнения отображается поверх существующего. |
| 7 | **Исходное сообщение** | Сообщение от участника самому себе. Результирующее вхождение выполнения отображается поверх отправляющего выполнения. |
| 8 | **Создайте сообщение** | Сообщение, создающее участника. Если участник получает сообщение о создании, он должен быть первым, кто его получает. |
| 9 | **Найти сообщение** | Асинхронное сообщение от неизвестного или не указанного участника. |
| 10 | **Потерянное сообщение** | Асинхронное сообщение неизвестному или не указанному участнику. |
| 11 | **Комментарий** | Примечание можно подключить к любой точке линии жизни. |
| 12 | **Использование взаимодействия** | Заключает последовательность сообщений, которые определены в другой схеме.  Чтобы создать **использование взаимодействия**, щелкните инструмент и выполните перетаскивание поверх линий жизни, которые требуется включить. |
| 13 | **Объединенный фрагмент** | Коллекция фрагментов. Каждый фрагмент может включать одно или несколько сообщений. Существует несколько видов объединенных фрагментов.  Чтобы создать фрагмент, щелкните сообщение правой кнопкой мыши, наведите указатель на пункт **Разместить во фрагменте**, после чего выберите тип фрагмента. |
| 14 | **Фрагмент условия** | Может использоваться для установки условия, зависящего от того, будет ли найден фрагмент.  Чтобы задать условие, выберите фрагмент, выберите условие и введите значение. |

### Диаграмма деятельности

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.
* Шаги в тестовом случае.
* Программный протокол, т. е. разрешенная последовательность взаимодействий между компонентами.
* Программный алгоритм.

#### Чтение схем активности

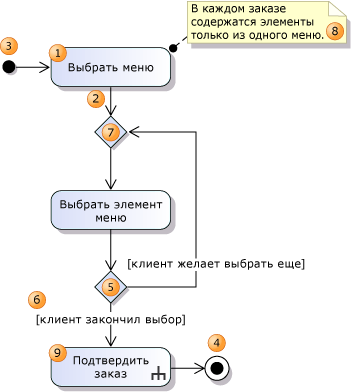


Действия и другие элементы, отображаемые на схеме активности, представляют собой одно действие. Эти действия можно просмотреть в обозревателе моделей UML. Он создается при добавлении первого элемента в схему.

Чтобы прочитать схему, представьте, что поток управления проходит вдоль соединителей от одного действия к другому.

#### Простые потоки управления

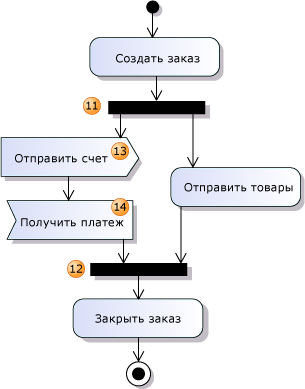
Последовательность действий можно показать с помощью ветвей и циклов.



| **Фигура** | **Элемент** | **Описание и основные свойства** |
| --- | --- | --- |
| 1 | **Действие** | Шаг в действии, в котором пользователи программы выполняют какие-либо задачи. |
| 2 | **Поток управления** | Соединитель, который показывает поток управления между действиями.  Чтобы создать поток управления, используйте средство **Соединитель**. |
| 3 | **Начальный узел** | Указывает первый шаг или шаги в действии. |
| 4 | **Конечный узел действия** | Окончание действия. |
| 5 | **Узел решений** | Условная ветвь в потоке. Имеет один вход и два или более выходов. |
| 6 | **Условие** | Условие, которое задает, может ли токен проходить вдоль соединителя. Чаще всего используются на исходящих потоках узла решений.  Чтобы задать условие, щелкните поток правой кнопкой мыши, выберите **Свойства** и задайте свойство **Условие**. |
| 7 | **Узел слияния** | Требуется для слияния потоков, разделенных узлом решений. Имеет два или более входов и один выход. Токен на любом входе отображается на выходе. |
| 8 | **Комментарий** | Предоставляет дополнительные сведения об элементах, с которыми связан. |
| 9 | **Действие вызова поведения** | Действие, которое определяется более подробно на другой схеме активности. |
| (не показана) | **Действие вызова операции** | Действие, которое вызывает операцию для экземпляра класса. |
|  | **Действия** | Поток работ, описываемый схемой активности. Чтобы просмотреть свойства действия, необходимо выбрать его в **Обозревателе моделей UML**. |
|  | **UML-схема активности** | Эта схема отображает действие. Чтобы просмотреть ее свойства, щелкните пустую область схемы. |

#### Параллельные потоки

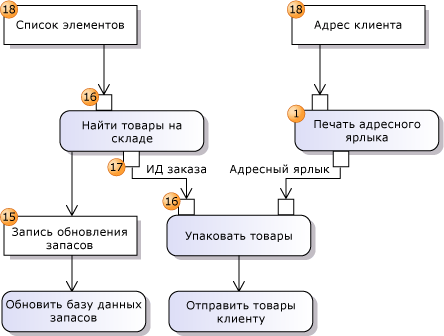
Можно описать последовательности действий, выполняемых одновременно.



| **Фигура** | **Элемент** | **Описание** |
| --- | --- | --- |
| 11 | **Вилочный узел** | Разделяет единый поток на параллельные потоки. Каждый входящий токен создает токен на каждом исходящем соединителе. |
| 12 | **Узел присоединения** | Объединяет параллельные потоки в один поток. Если каждый входящий поток имеет ожидающий токен, создается токен на выходе. |
| 13 | **Действие отправки сигнала** | Действие, которое отправляет сообщение или сигнал другому действию или параллельному потоку того же действия. Тип и содержимое сообщения видны из названия действия или задаются в дополнительных комментариях. |
| 14 | **Действие события принятия** | Действие, которое ожидает сообщения или сигнала, чтобы продолжиться. Тип сообщения, которое может быть получено действием, виден из названия или задается в дополнительных комментариях.  Если действие не имеет входящего потока управления, оно создает токен всякий раз при получении сообщения. |

#### Потоки данных

Можно описать поток данных из одного действия в другое.



| **Фигура** | **Элемент** | **Описание** |
| --- | --- | --- |
| 15 | **Узел объекта** | Представляет данные, передаваемые в потоке.   * **Ordering** — способ хранения нескольких токенов. * **Selection** — вызывает процесс фильтрации данных, который можно определить на другой схеме. * **Upper Bound** — 0 означает, что данные должны передаваться в потоке напрямую; \* означает, что данные можно хранить в потоке. * **Type** — тип хранимых и передаваемых объектов. |
| 16 | **Закрепление ввода** | Представляет данные, которые действие может получать при выполнении. |
| 17 | **Закрепление вывода** | Представляет данные, которые действие создает при выполнении. |
| 18 | **Узел параметра действия** | Узел объекта, через который действие может получать или создавать данные.  Используется, если представленное схемой действие вызывается из другого действия, либо если схема описывает операцию или функцию. |

## Выводы и анализ результатов работы.

В результате работы было осуществлено ознакомление с нотацией UML для функционирования систем и исследование диаграмм с применением среды Visual Studio Ultimate на примере моделирования системы.

UML-схемы моделирования и схемы слоев могут существовать только внутри проекта моделирования.

Цель и сценарии включенного варианта использования должны иметь смысл независимо друг от друга, чтобы их можно было включать в варианты использования, создаваемые позже.

Разделение вариантов использования на включающие и включенные части позволяет достичь следующих целей:

• Структурировать описания вариантов использования по уровню детализации.

• Избежать дублирования общих сценариев в разных вариантах использования.