Министерство образования и науки

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

Факультет инфокоммуникационных технологий

Отчет

по лабораторной работе № 2

# «Создание диаграмм в Visual Studio»

**Программное обеспечение Мини-АТС**

по дисциплине «Проектирование инфокоммуникационных систем»

**Выполнил: Кислюк И. В. K4220**

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

Санкт-Петербург

2018

## Цель работы.

Создание диаграмм моделей средствами Visual Studio.

## Задачи, решаемые при выполнении работы.

### Разработать основные UML-диаграммы.

### 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 пяти часто используемых UML-диаграмм:

* вариантов использования,
* активности,
* классов,
* компонентов,
* последовательностей.

Кроме того, можно создавать схемы слоев, которые помогают определить структуру системы.

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.

## 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.

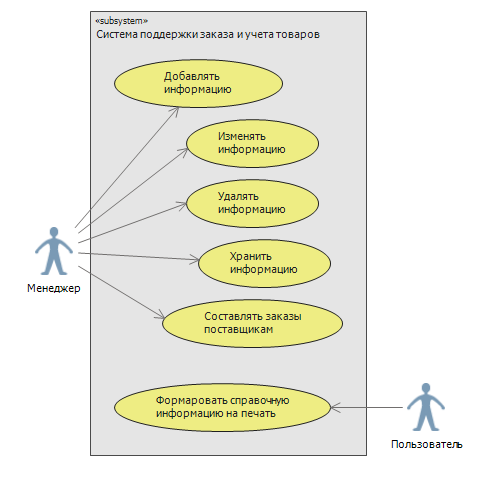
### Разработка схемы вариантов использования.

Схема вариантов использования является основным инструментом, используемым для описания пользовательских требований. Она описывает отношения между требованиями, пользователями и основными компонентами. Однако схема вариантов использования не описывает требования подробно, их можно представить на отдельных схемах или в документах, которые можно связать с каждым вариантом использования.

#### Чтение схем вариантов использования



В следующих таблицах описаны элементы, которые можно использовать на схеме вариантов использования, и их основные свойства.



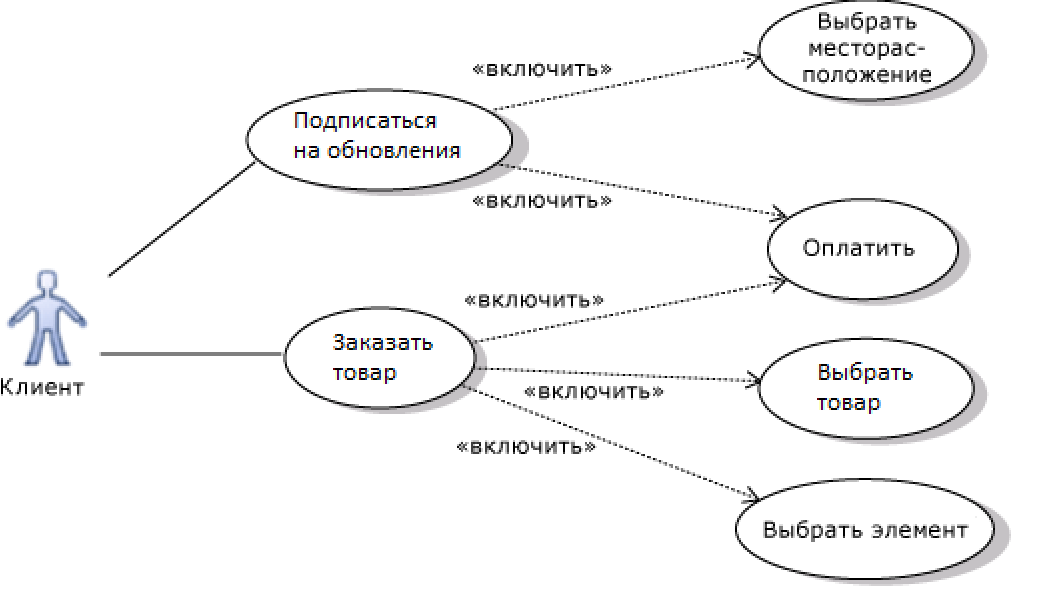
* *Актер* (1) — в терминах VS называется *субъектом*, это класс лиц, организаций, устройств или внешних программных компонентов. взаимодействующих с системой. Примерами субъектов являются следующие: Клиент, Ресторан, Датчик температуры, Устройство авторизации кредитных карт.
* *Вариант использования* (2) представляет действия, совершаемые одним или несколькими субъектами для достижения определенной цели. Примерами вариантов использования являются следующие: Заказ еды, Обновление меню, Обработка платежа.

На схеме вариантов использования они ассоциированы (3) с субъектами, выполняющими их.

* *Система (4)* — это любой объект в разработке. Системой может быть небольшой программный компонент, субъектами которой являются другие программные компоненты, полное приложение или крупный распределенный набор приложений, развернутых на нескольких компьютерах и устройствах. Примерами подсистем являются следующие: "Веб-сайт для заказа еды", "Бизнес по доставке еды", "Веб-сайт, версия 2".

Схема вариантов использования может показывать, какие варианты использования поддерживаются системой или ее подсистемами.

#### Структурирование вариантов использования



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.

### Диаграмма последовательностей

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:

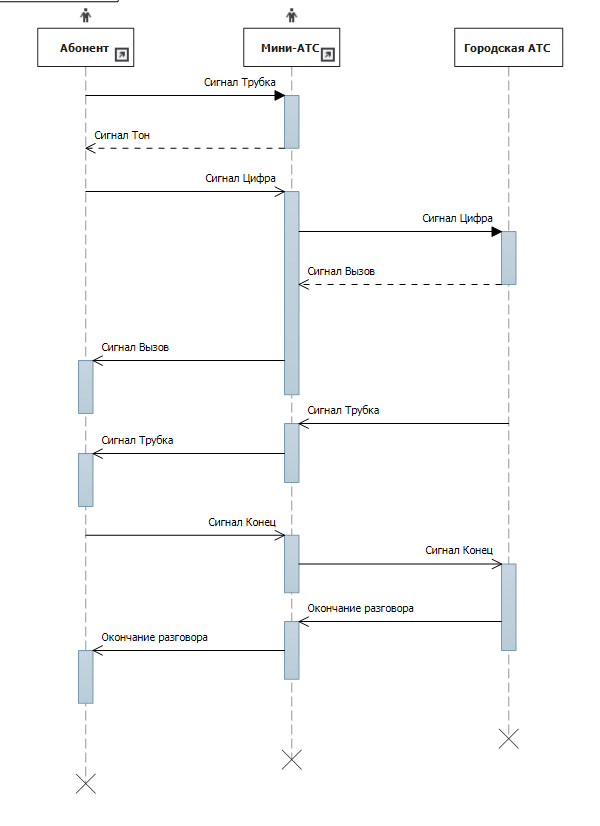
* Схемы последовательностей, основанные на коде могут быть созданы из кода программы .NET и помещены в любой проект.
* Схемы последовательностей UML образуют часть модели UML и существуют только в пределах UML-проекта моделирования.

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. |

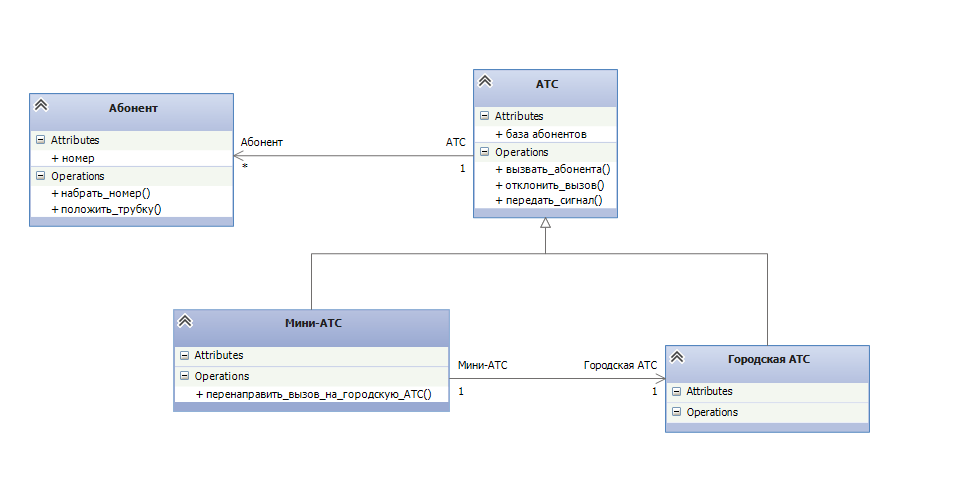
### Диаграмма классов

UML-схема классов описывает структуры объектов и сведений, используемые для внутренней организации приложения и для взаимодействия с пользователями. Кроме того, схема предоставляет сведения об этих структурах безотносительно какой-либо конкретной реализации. Ее классы и отношения могут реализовываться несколькими способами, например, в таблицах базы данных, XML-узлах или сочетаниях программных объектов.

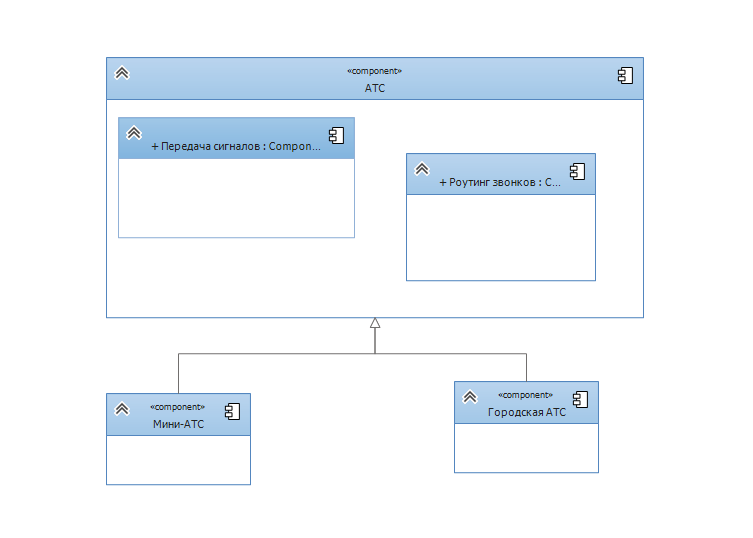
#### Чтение схем классов



В этом разделе в таблице описаны элементы, которые можно увидеть на UML-схеме классов.



| **Фигура** | **Элемент** | **Описание** |
| --- | --- | --- |
| 1 | **Класс** | Определение объектов, совместно обладающих данными характеристиками структуры и поведения. |
| 1 | **Классификатор** | Общее имя для класса, интерфейса или перечисления. Компоненты, варианты использования и субъекты также являются классификаторами. |
| 2 | **Элемент управления "свернуть/развернуть"** | Используется для просмотра подробностей классификатора |
| 3 | **Атрибут** | Типизированное значение, прикрепленное к каждому экземпляру классификатора. |
| 4 | **Операция** | Метод или функция, которую можно выполнить с помощью экземпляров классификатора. Чтобы добавить операцию, щелкните раздел **Операции** и нажмите **ВВОД**. Введите сигнатуру операции. |
| 5 | **Ассоциация** | Отношение между членами двух классификаторов. |
| 5a | **Агрегат** | Ассоциация, представляющая отношение совместного владения. Свойству **Агрегат** роли-владельца присвоено значение **Сделано общим**. |
| 5б | **Композиция** | Ассоциация, представляющая отношение целого и части. Свойству **Агрегат** роли-владельца присвоено значение **Составной**. |
| 6 | **Имя ассоциации** | Имя ассоциации. Имя может оставаться пустым. |
| 7 | **Имя роли** | Имя роли, т. е. одного из окончаний ассоциации. Может использоваться для ссылки на связанный объект. |
| 8 | **Количество элементов** | Указывает, сколько объектов на этом окончании можно связать с объектами на другом окончании. |
| 9 | **Обобщение** | *Конкретный* классификатор наследует часть своего определения от *общего* классификатора. Общий классификатор находится на окончании соединителя с указателем стрелки. Атрибуты, ассоциации и операции наследуются конкретным классификатором. Используется инструмент **Наследование**, для создания обобщения между двумя классификаторами. |



| Фигура | Элемент | Описание |
| --- | --- | --- |
| 10 | **Интерфейс** | Определение части внешне видимого поведения объекта |
| 11 | **Перечисление** | Классификатор, состоящий из набора строковых литералов. |
| 12 | **Пакет** | Группа классификаторов, ассоциаций, действий, линий жизни, компонентов и пакетов. Логическая схема классов показывает, что членами данного пакета являются классификаторы и пакеты. |
| 13 | **Импорт** | Отношение между пакетами, указывающее, что один пакет включает все определения другого. |
| 14 | **Зависимость** | Определение или реализация зависимого классификатора может измениться, если изменяется классификатор на окончании с наконечником стрелки. |

|  |  |  |
| --- | --- | --- |
| Фигура | **Элемент** | Описание |
| 15 | **Реализация** | Класс реализует операции и атрибуты, определенные интерфейсом. Воспользуйтесь инструментом **Наследование**, чтобы создать реализацию между классом и интерфейсом. |
| 16 | **Реализация** | Альтернативное представление того же отношения. Метка на символе обозначения указывает на интерфейс.  Чтобы создать эту презентацию, выделите существующее отношение реализации. Рядом с ассоциацией появляется тег действия. Щелкните тег действия и выберите **Показывать без описания операций**. |

### 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

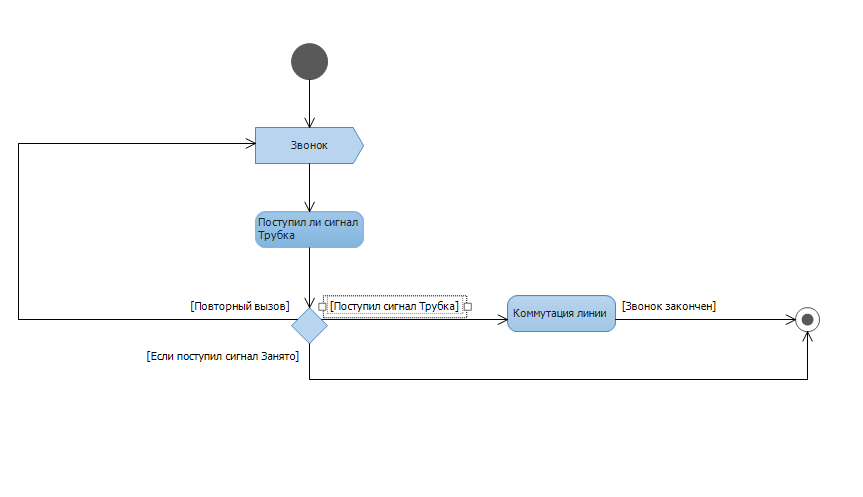


Действия и другие элементы, отображаемые на схеме активности, представляют собой одно действие. Эти действия можно просмотреть в обозревателе моделей UML. Он создается при добавлении первого элемента в схему.

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

#### Простые потоки управления

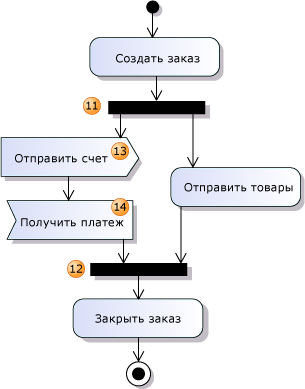
Последовательность действий можно показать с помощью ветвей и циклов.



| **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

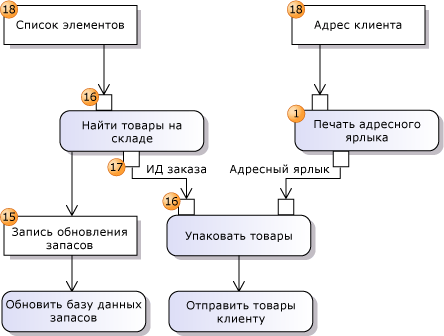
Можно описать последовательности действий, выполняемых одновременно.



| **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. |

### Диаграмма компонентов

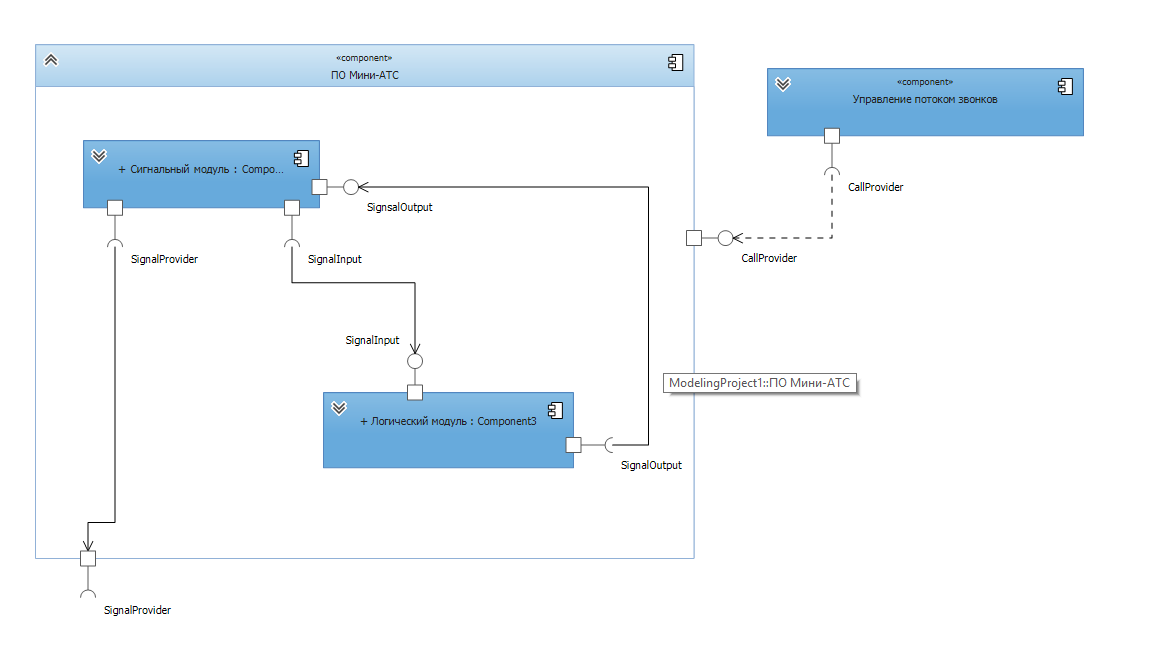
В Visual Studio Ultimate на *схеме компонентов* показаны части конструкции программной системы. Схема компонентов помогает визуализировать высокоуровневую структуру системы и поведение служб, предоставляемых и потребляемых этими элементами через интерфейсы.

Схему компонентов можно использовать, чтобы описать конструкцию системы, реализуемую на любом языке и в любом стиле. Нужно только определить части конструкции, взаимодействующие с другими частями через ограниченный набор входных и выходных каналов. Можно использовать компоненты любого масштаба, взаимосвязанные любым способом.

#### Чтение схем компонентов



Ниже в таблице описаны элементы, которые можно использовать на схеме компонентов, и их основные свойства.



| **Фигура** | **Элемент** | **Описание и основные свойства** |
| --- | --- | --- |
| 1 | **Компонент** | Допускающий повторное использование функциональный элемент системы. Компонент предоставляет и потребляет поведение через интерфейсы и может использовать другие компоненты.  Можно скрывать или отображать внутренние части компонента с помощью элемента управления "развернуть/свернуть" (9).  Компонент — это вид класса.   * **Является неявно создаваемым экземпляром**. Если значение true (по умолчанию), компонент существует только как артефакт конструкции. Во время выполнения существует только ее часть. |
| 2 | **Предоставленный порт интерфейса** | Представляет группу сообщений или вызовов, реализуемых компонентом и доступных для использования другими компонентами или внешними системами. Порт — это свойство компонента, имеющее в качестве типа интерфейс. |
| 3 | **Требуемый порт интерфейса** | Представляет группу сообщений или вызовов, отправляемых компонентом другим компонентам или внешним системам. Компонент предназначен для соединения с компонентами, которые предоставляют хотя бы эти операции. Порт имеет в качестве типа интерфейс. |
| 4 | **Зависимость** | Может использоваться для указания, что требуемый интерфейс одного компонента может соответствовать предоставленному интерфейсу другого компонента.  Зависимости также можно использовать в более общем случае при работе с элементами модели, чтобы показать, что конструкция одного зависит от конструкции другого. |
| 5 | **Часть** | Атрибут компонента, тип которого, как правило, является другим компонентом. Часть используется при внутреннем проектировании ее родительского компонента. Графически части изображаются вложенными в родительский компонент.  Чтобы создать часть существующего типа компонента, перетащите компонент из Проводника по моделям UML в компонент-владелец.  Чтобы создать часть нового типа, выберите инструмент **Компонент** и щелкните компонент-владелец. Например, компонент Car имеет части engine:CarEngine, backLeft:Wheel, frontRight:Wheel и т. д.  Несколько частей могут иметь один и тот же тип, и разные компоненты могут иметь части одного типа.   * **Тип**. Тип части, определяемый в другом месте модели. Как правило, типом является другой компонент. * **Количество элементов**. По умолчанию используется значение 1. Можно задать значение **0..1**, чтобы указать, что часть может иметь значение **null**, или задать значение **\***, чтобы указать, что часть является коллекцией экземпляров данного типа. Также в качестве значения можно задать любое выражение, которое можно оценить в числовом диапазоне. |
| 6 | **Сборка части** | Соединение между требуемыми портами интерфейса одной части и предоставленными портами интерфейса другой. |
| 7 | **Делегирование** | Связывает порт с интерфейсом одной из частей компонента. Указывает, что сообщения, отправленные компоненту, обрабатываются этой частью, или что сообщения, отправленные этой частью, отсылаются из родительского компонента. |
| 8 | **Обобщение** | Указывает, что один компонент наследуется от другого. Части и интерфейсы наследуются. |
| 9 | **Элемент управления "**развернуть/свернуть**"** | Позволяет скрывать или отображать внутренние части компонента. |

## 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.