**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ  
К ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ   
ПО ДИСЦИПЛИНЕ**

**"ПРОГРАММНАЯ ИНЖЕНЕРИЯ"**

**ОГЛАВЛЕНИЕ**

[Выделение классов 2](#_Toc128239953)

[Внутренняя структура, размещение 6](#_Toc128239954)

[Варианты использования 12](#_Toc128239955)

[Взаимодействия 13](#_Toc128239956)

[Состояния 14](#_Toc128239957)

### Выделение классов

**Метод Аббота** заключается в изучении текстовых описаний проблемы или постановки задач с целью выделения классов-кандидатов. Метод определяет набор правил извлечения существительных из текста, объединения синонимов и исключения неподходящих существительных, таких как числительные, н е имеющие отношения к изучаемой проблеме понятия и другие.

**Метод именных групп** является модификацией оригинального метода Аббота, изначально сформулированного для извлечения сущностей в структурном проектировании. Метод именных групп также основан на выделении существительных, но оставляет на усмотрение аналитика критерии отнесения класса-кандидата к релевантным, нерелевантным или нечетких классам.

**Метод карточек класс-контракт-коллеги** (CRC – class – responsibilities – coolloborators) (класс – обязанности - участники) направлен на анализ взаимодействия объектов для реализации некоторой функции системы. При рассмотрении сценариев взаимодействий участники формулируют обязанности владения информацией, выполнения действия или организации взаимодействия, назначаемые предполагаемым объектам. Метод применяется итеравтивно, последовательно рассматривая ключевые функции системы. Результатом метода являются описания обязанностей объектов и их взаимодействия, обычно представляемые в виде карточек формата 90 на 140 мм.

**Метод шаблонных классов** состоит в поиске в системе классов, обычно выделяемых в сходные по назначению системах. Для этого применяются перечни категорий классов, таких как внешние сущности, вещи, события, организационные единицы, места, структуры классов их других категорий.

Ассоциация

Ассоциация показывает, что объекты одной сущности (класса) связаны с объектами другой сущности.

Существует пять различных типов ассоциации. Наиболее распространёнными являются двунаправленная и однонаправленная. Например, классы «рейс» и «самолёт» связаны двунаправленной ассоциацией, а классы «человек» и «кофейный автомат» связаны однонаправленной.

Двойные ассоциации (с двумя концами) представляются линией, соединяющей два классовых блока. Ассоциации более высокой степени имеют более двух концов и представляются линиями, один конец которых идет к классовому блоку, а другой к общему ромбику. В представлении однонаправленной ассоциации добавляется стрелка, указывающая на направление ассоциации.

Ассоциация может быть именованной, и тогда на концах представляющей её линии будут подписаны роли, принадлежности, индикаторы, мультипликаторы, видимости или другие свойства.

Агрегация

[http://upload.wikimedia.org/wikipedia/commons/thumb/2/2a/KP-UML-Aggregation-20060420.svg/300px-KP-UML-Aggregation-20060420.svg.png](http://commons.wikimedia.org/wiki/File:KP-UML-Aggregation-20060420.svg?uselang=ru)

Диаграмма классов, показывающая Агрегацию между двумя классами

Агрегация — это разновидность ассоциации при отношении между целым и его частями. Как тип ассоциации агрегация может быть именованной. Одно отношение агрегации не может включать более двух классов (контейнер и содержимое).

Агрегация встречается, когда один класс является коллекцией или контейнером других. Причём по умолчанию, агрегацией называют агрегацию по ссылке, то есть когда время существования содержащихся классов не зависит от времени существования содержащего их класса. Если контейнер будет уничтожен, то его содержимое — нет.

Графически агрегация представляется пустым ромбиком на блоке класса и линией, идущей от этого ромбика к содержащемуся классу.

Рассмотрим в качестве примера задание 1.

В класс Folder добавить следующие операции:

+CreateSubFolder (path:String),

+CreateFile (name:String),

+DeleteSubFolder (path:String),

+DeleteFile (name:String).

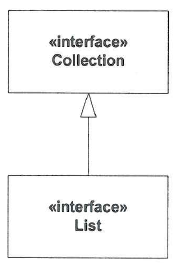
Добавляем в модель вспомогательный класс Buffer, в класс File добавляем следующие операции:

+ReadToBuffer (buf:Buffer, pos:Integer),

+WriteToFile (buf:Buffer, pos:Integer).

Прежде всего, необходимо прочитать задачу и представить описанную в ней модель на диаграмме классов. Рассмотрим задание 5 в качестве примера. В условии сказано, что в модели присутствует интерфейс доступа к элементам коллекции, который называется Collections. Далее говорится, что интерфейс работы со списками с названием List уточняет интерфейс Collection и, наоборот, интерфейс Collection обобщает интерфейс List. Значит, в модели также присутствует отношение обобщения между интерфейсами List и Collection.

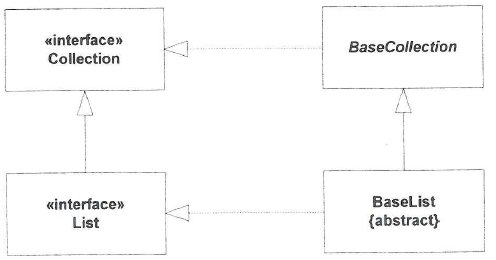
Представим имеющиеся три элемента на диаграмме.



Интерфейс является особым видом классификатора, изображается в виде прямоугольника с ключевым словом interface в двойных угловых кавычках. Оба интерфейса не имеют операций или атрибутов, поэтому соответствующие секции на диаграмме не показаны. Отношение обобщения указывается треугольной стрелкой, направленной от уточняющего к более общему элементу.

Далее в условии сказано, что в модели определен класс с названием BaseCollection, который является абстрактным. Класс реализует интерфейс Collection, значит в модели определено отношение реализации интерфейса Collection классом BaseCollection, и отношение реализации интерфейса List классом BaseList.

Модель, описанная в условии задачи, представлена на рис.

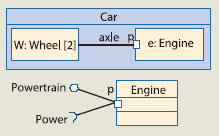


Класс является классификатором, изображается в виде прямоугольника, назнвание абстрактного класса указывается курсивом, как для класса BaseCollection или ключевым словом abstract, следующим после названия класса.

Чтобы определить, каким образом писать ключевые слова: в кавычках или фигурных скобках, можно использовать простое правило. Ключевое слово “ubterface” пишется в двойных кавычках потому, что указывает на метакласс Interface, определенный в UML, в то время как {abstract} указывает на значение свойства isAbstract метакласса Class.

### Внутренняя структура, размещение

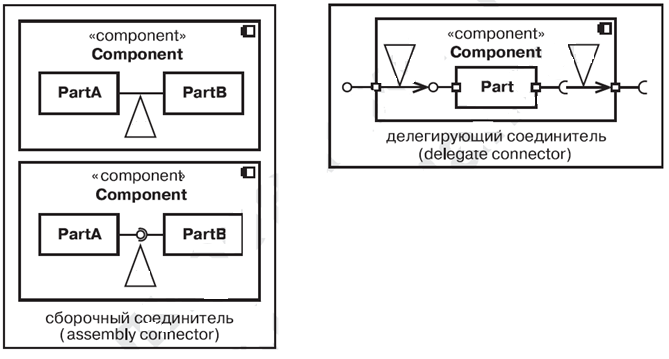
Для улучшения поддержки моделирования программной архитектуры и многократного использования классификаторов в нескольких средах в UML 2.0 введены понятия структурированного классификатора и порта. Структурированный классификатор имеет внутреннюю структуру, описываемую структурой его составных частей.

  
Пример структурированных классов UML 2.0, взятый из спецификации Суперструктуры UML 2.0. Структурированный класс Car («автомобиль») состоит из двух частей: W:Wheel («колесо») и e:Engine («двигатель»). Часть W состоит из двух «колес», а «двигатель» e является экземпляром структурированного класса Engine. Последний имеет порт p, реализующий интерфейс Powertrain («силовая передача»)

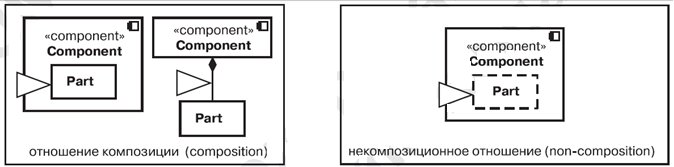
На рис. показан пример структурированных классов. Так, класс Car («автомобиль») состоит из двух частей: W:Wheel («колесо») и e:Engine («двигатель»). Часть W состоит из двух «колес», а «двигатель» e является экземпляром структурированного класса Engine («мотор»). Эти части связывает соединитель axle («вал»). Класс Engine имеет порт с именем p, реализующий предоставляемый интерфейс Powertrain («силовая передача») и требующий операций, описанных в требуемом интерфейсе Power («мощность»).

**Структурированным классификатором** называется классификатор, который может включать соединители, связывающие содержащиеся в классификаторе свойства. Структурированные классификаторы определяют контекст для соединителей и позволяют описать связи между экземплярами, возникающие во время выполнения системы.

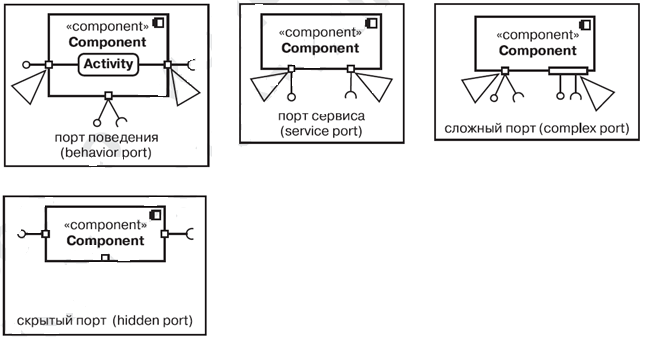
**Соединители** – это черты структурированного классификатора, имеющие тип и связывающие два или более свойств классификатора. В то время как связи (links) являются экземплярами ассоциаций, соединители ограничивают возможные связи между экземплярами, в зависимости от классификатора, которому принадлежат эти экземпляры.Соединители бывают **сборочные (assembly connector)** и **делегирующие (delegate connector)**. Сборочный соединитель соединяет части структурированного классификатора между собой, а делегирующий связывает часть с портом.



**Частью классификатора** называется свойство, с которым классификатор связан отношением композиции (операция или атрибут).



**Портом** называется часть классификатора, через которую осуществляется взаимодействие с другими внешними классификаторами. Порт может предоставлять и требовать наличия нескольких интерфейсов. Делегирующий соединитель связывает порт с одной из частей классификатора, если данная часть реализует предоставляемый порт интерфейс или нуждается в требуемом интерфейсе.



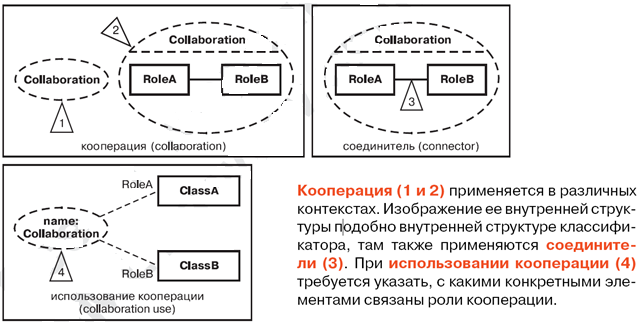
**Сборочный соединитель** связывает части классификатора или разных классификаторов между собой и указывает возможные пути взаимодействия частей.

**Компонент** является модулем системы, который скрывает свою внутреннюю структуру, взаимодействует с другими компонентами через определенные интерфейсы, и материализация (manifestation) которого в системе может быть заменена при сохранении окружения (environment).Таким образом, поведение компонента полностью определяется набором предоставляемых и требуемых интерфейсов. Компоненты используются для определения повторно используемых частей системы и для описания модульной структуры системы.

**Кооперация** является структурированным классификатором, обладающих поведением, и определяет роли составляющих ее частей и взаимодействия между ними в контексте кооперации. Кооперации используются для определения взаимодействий, обеспечивающих достижение какой-либо цели или реализации функциональности.

Вхождение кооперации связывает элементы модели с ролями, определенными в кооперации.

**Ролями** в кооперации называют части кооперации, параметры поведения или локальные переменные в поведении, структуру которого описывает кооперация.



**Артефакт** описывает файл, документ или другой объект, который используется или создается в процессе разработки программы или в результате ее установки и использования. Артефакт в модель является классификатором и может вступать в ассоциации с другими артефактами.

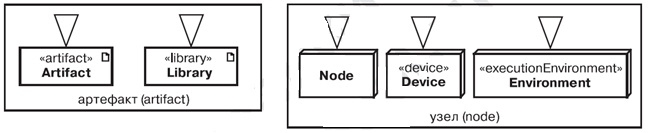
Артефакты являются манифестацией (материализуют) других элементов модели, таких как компоненты.

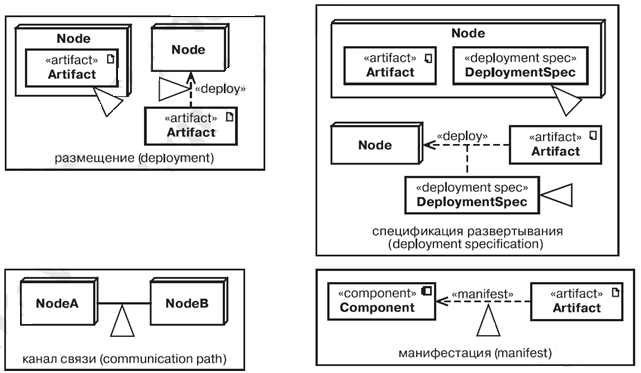
**Узел** описывает вычислительный ресурс, на котором могут быть развернуты для выполнения артефакты. Узлы взаимодействуют через каналы связи (communication path). В метамодель узел является специальным видом класса, а каналы связи – ассоциациями между уздами. Вычислительные элементы оборудования моделируются устройствами, а программы платформы – средами исполнения.

Диаграмма размещения используется для описания структуры системы с точки зрения артефактов, размещенных на вычислительных узлах.

Основные сущности, используемые на диаграмме: артефакты (1 и 2) и узлы (3, 4 и 5). Размещение артефактов на узлах задается с помощью отношения размещения (6 и 7), которое может быть дополнено спецификацией развертывания (8 и 9). Узлы связываются между собой отношением, которое называется канал связи (10).

Если требуется показать, какие компоненты физически реализует тот или иной артефакт, то для этого применяется отношение манифестации (11).





Прежде всего, необходимо прочитать задачу и представить описанную в ней модель на диаграмме классов. Рассмотрим задание 9 в качестве примера. В условии задачи сказано, что в модели определена кооперация Sales, в которую роли Salesman и Customer входят как составные части. Представим модель на диаграмме внутренней структуры.

Кооперация изображается эллипсом с пунктирной линией, роли Salesman и Customer представлены прямоугольниками. Согласно спецификации роли в кооперации colloborationRole являются подмножеством ролей в структурной классификаторе, метаклассе, который уточняет кооперация. В структурной классификаторе прямоугольником со сплошной линией обозначаются только части, т.к. роли, которые классификатор включает через композицию. Остальные роли изображаются прямоугольниками с пунктирной линией. Тем не менее в спецификации роли в кооперации также изображаются прямоугольниками со сплошной линией.

В первом вопросе необходимо показать, что роли взаимодействуют между собой в рамках данной кооперации. Для этого нужно добавить соединитель между Salesman и Customer, так как по определению, соединитель указывает возможные пути взаимодействия. В условии не говорится о типе соединителя и ролей кооперации, поэтому ответом по первому вопросу будет диаграмме, представленная на рисунке.

Соединитель между ними изображен сплошной линией.

Перейдем ко второму вопросу. В условии сказано, что необходимо создать новую кооперацию BrokeredSale, в которой осуществляется продажа с посредником, и уточняется, что посредник представляется покупателю продавцом, а продавцу – покупателем. В первом вопросе была построена кооперация взаимодействия покупателя с продавцом, поэтому, следуя указанию использовать вхождение кооперации, определим кооперацию BrokeredSale так, чтобы она включала два вхождения кооперации Sale: одно для взаимодействия продавца с посредником, а второе – для взаимодействия покупателя с посредником. Полученная кооперация подставлена на диаграмме.

Роль customer при вхождении кооперации Sales связана с ролью Broker кооперации BrokeredSale, роль salesman – с ролью Salesman. Роль salesman при вхождении кооперации Customers связана с ролью Broker кооперации BrokeredSale, а роль customer – с ролью Customer.

### Варианты использования

Разработка диаграммы вариантов использования преследует цели:

1. Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы.
2. Сформулировать общие требования к функциональному поведению проектируемой системы.
3. Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.
4. Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования (use case) служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой.

В самом общем случае, диаграмма вариантов использования представляет собой граф специального вида, который является графической нотацией для представления конкретных вариантов использования, актеров, возможно некоторых интерфейсов, и отношений между этими элементами. При этом отдельные компоненты диаграммы могут быть заключены в прямоугольник, который обозначает проектируемую систему в целом. Следует отметить, что отношениями данного графа могут быть только некоторые фиксированные типы взаимосвязей между актерами и вариантами использования, которые в совокупности описывают сервисы или функциональные требования к моделируемой системе.

### Взаимодействия

Рассмотрим в качестве примера задание 19. Условие задачи описывает взаимодействие и начинается с представления роли автора, который отправляет сообщение другому участнику – редактору. Указано, что автор ожидает подтверждения о получении, значит, сообщение является синхронным и подтверждение следует моделировать ответным сообщением. После отправки подтверждения автору, редактор отправляет сообщение рецензенту. Здесь не указано, что редактор ожидает результата или подтверждения, поэтому сообщение является асинхронным. После получения сообщения от редактора рецензент направляет редактору сообщение с результатами рецензирования, которое также является асинхронным.

Обратите внимание на порядок возникновения событий в задаче. При изложении условия задач подразумевается, что возникновение событий, описываемых в следующем предложении, происходит позднее всех событий для упоминаемых ранее участников.

Далее, редактор после получения сообщения с результатом от рецензента направляет сообщение автору и, после этого, сообщение с благодарностью рецензенту. Диаграмма последовательности построенной модели приведена на рисунке.

Укажет общее правило, согласно которому следует добавлять в модель спецификации исполнения. Если в результате получения сообщения предполагается обработка или исполнение поведения участником взаимодействия, то следует добавить спецификацию исполнения. Если же такой обработки не предполагается или она несущественна для целей моделирования, спецификацию исполнения на диаграмме можно опустить.

Также обратите внимание, что начало спецификации исполнения совпадает с получением обрабатываемого сообщения, и отправка ответного сообщения совпадает с завершением обработки. Поэтому во многих случаях ответные сообщения могут быть на диаграмме не показаны.

### Состояния

Рассмотрим в качестве примера задание 6. В условии указан классификатор TrafficLights – светофор, поведение которого описывается конечным автоматом. После создания экземпляра светофора, автомат переходит из начального псевдосостояния в состояние выключен Offline. Переход выполянется безусловно, эффект при переходе отсутствует. Далее упоминается композитное состояние Online, в которое вложено состояние Green. При получении события On светофор должен перейти в состояние Green. Поэтому добавляем переход из состояния Offline в Green. О дополнительных условиях перехода ничего не сказано, поэтому сторожевое условие не указываем. Заметим, что вид события On не указан, поэтому можно полагать его событием получения сигнала.

Далее в условии сказано, что по истечении 50 секунд автомат переходит в состояние Yellow. Значит, в модель нужно добавить переход из состояния Green в состояние Yellow, также вложенное в Online, и указать триггер с событием времени. Последнее возникает по истечении интервала времени равного 50 секундам от момента, когда состояние Green стало активным в последний раз. Аналогично, добавляет переход из Yellow в состояние Red и указывает событие времени с интервалом в три секунды, и переход из Red в Green с интервалом в 50 секунд. Диаграмма построенной схемы состояний приведена на рисунке.

Нужно добавить возможность выключить светофор. По аналогии с включением добавляем переход из состояния включено Online в состояние выключено Offline, указываем приведенное в условии событие Off, которое можно полагать событием получения сигнала. Ответом по первому вопросу будет модель, представленная на рисунке.

Необходимо обеспечить возможность настройки интервалов между сигналами светофора, в то время как в текущей модели интервалы заданы константами. Для этого в контекстный классификатор TrafficLights добавим три атрибута для хранения продолжительности сигналов светофора: green для состояния Green, yellow и red для состояний Yellow и Red соответственно. Заменим в триггерах переходов между вложенными состояниями в Online константы на выражения времени, извлекающие значения из введенных ранее атрибутов.

По условию, длительности интервалов сигналов светофора, а значит, добавленных атрибутов, следует задавать, когда светофор находится в отключенном состоянии Offline. Как отразить это в конечном автомате? Вспомним, что автомат описывает поведение классификатора и вызываемое в нем поведение в ответ на =события. Поэтому автомат описывает, в том числе, какое поведение будет вызвано при получении сигнала или вызове операции (если явно не указано в поведении, что оно реализует данную операцию вне зависимости от состояния классификатора). Добавим в состояние Offline внутренние переходы в себя с триггером по событию вызова операций setGreen, setYellow и setRed с параметром t. Каждому из переходов укажем поведение в качестве эффекта – присваивание соответствующему операции атрибуту значения t. Построенная модель представлена на рисунке.

Следует обратить внимание на следующие моменты. Первый: вызов операций setGreen, setYellow и setRed будет игнорирован, если светофор не выключен, так как поведение, вызываемое при приеме события вызова этих операций, указано только в состоянии Offline. Второй: для обработки вызовов операций в модели использованы внутренние переходы, а не внешние переходы в себя, что можно считать хорошим тоном в моделировании. Действительно, выполнение внешнего перехода, помимо выполнения эффекта, приводит к прерыванию действия, выполняемого в состоянии, выполнению действия при выходе, выполнению действия при входе в состояние, обновлении времени входа в состояние, что влияет на возникновение событий времени для триггеров, определенных в данном состоянии. Все эти моменты необходимо учитывать при использовании внешних переходов в себя для указания поведения, выполняемого при вызове операций.