Лабораторная работа №4.1

Цель: научиться создавать диаграмму последовательности.

В ходе проектирования ИС аналитик поэтапно спускается от общей концепции, через понимание ее логической структуры к наиболее детальным моделям, описывающим физическую реализацию.

С помощь диаграммы прецедентов (вариантов использования) выявляются основные пользователи системы и задачи, которые данная система должна решать. С помощью диаграммы деятельности мы описываем последовательность действий для каждого прецедента, необходимая для достижения поставленной цели.

Далее проектируется логическая структура системы с помощью диаграммы классов. На данном этапе выделяются классы, формирующие структуру БД Системы, а также классы реализующие некий набор операций, способствующий достижению целей в рамках выбранного прецедента. Для описания сложного поведения некоторых объектов (экземпляров класса) составляется диаграмма состояний.

Таким образом, аналитиками фиксируются такие поведенческие аспекты как алгоритм действий в рамках одного или нескольких прецедентов, необходимый для достижения определённого результата, а также изменение состояния объектов в ходе выполнения приведенных действий.

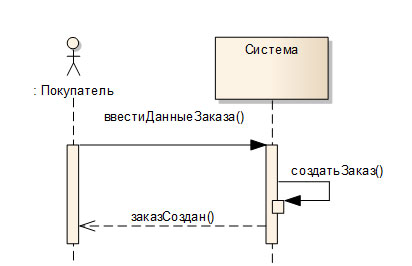
Зачастую на этапе спецификации требований необходимо показать не только алгоритм действий или изменение состояния объекта, но и обмен сообщениями между отдельными объектами Системы. Данную задачу решает диаграмма взаимодействия.

Диаграмма взаимодействия предназначена для моделирования отношений между объектами (ролями, классами, компонентами) Системы в рамках одного прецедента.   
Данный вид диаграмм отражает следующие аспекты проектируемой Системы:

* обмен сообщениями между объектами (в том числе в рамках обмена сообщениями со сторонними Системами)
* ограничения, накладываемые на взаимодействие объектов
* события, инициирующие взаимодействия объектов.

В отличие от диаграммы деятельности, которая показывает только последовательность (алгоритм) работы Системы, диаграммы взаимодействия акцентируют внимание разработчиков на сообщениях, инициирующих вызов определенных операций объекта (класса) или являющихся результатом выполнения операции.

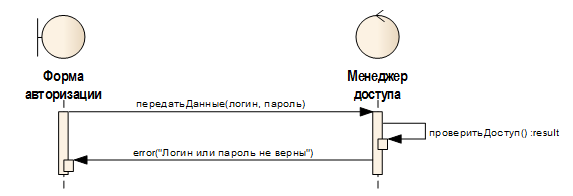
Диаграмма последовательности является одной из разновидности диаграмм взаимодействия и предназначена для моделирования взаимодействия объектов Системы во времени, а также обмена сообщениями между ними.

Одним из основных принципов ООП является способ информационного обмена между элементами Системы, выражающийся в отправке и получении сообщений друг от друга. Таким образом, основные понятия диаграммы последовательности связаны с понятием Объект и Сообщение.   
   
На диаграмме последовательности объекты в основном представляю экземпляры класса или сущности, обладающие поведением. В качестве объектов могут выступать пользователи, инициирующие взаимодействие, классы, обладающие поведением в Системе или программные компоненты, а иногда и Системы в целом.

Объекты располагаются с лева на права таким образом, чтобы крайним с лева был тот объект, который инициирует взаимодействие.   
Неотъемлемой частью объекта на диаграмме последовательности является линия жизни объекта. Линия жизни показывает время, в течение которого объект существует в Системе. Периоды активности объекта в момент взаимодействия показываются с помощью фокуса управления. Временная шкала на диаграмме направлена сверху вниз.

На диаграммах последовательности допустимо использование стандартных стереотипов класса:

|  |  |
| --- | --- |
| http://www.it-gost.ru/images/articles/uml/seq_2.gif | **Actor** – экземпляр участника процесса (роль на диаграмме прецедентов) |
| http://www.it-gost.ru/images/articles/uml/seq_3.gif | **Boundary** – Класс-Разграничитель -  используется для классов, отделяющих внутреннюю структуру системы от внешней среды (экранная форма, пользовательский интерфейс, устройство ввода-вывода). Объект со стереотипом <<bondary>> отличается от, привычного нам, класса <<Интерфейс>>, который по большей части предназначен для вызова методов класса, с которым он связан. Объект boundary показывает именно экранную форму, которая принимает и передает данные обработчику.</bondary>> |
| http://www.it-gost.ru/images/articles/uml/seq_4.gif | **Contro**l – Класс-контроллер - активный элемент, который используются для выполнения некоторых операций над объектами (программный компонент, модуль, обработчик) |
| http://www.it-gost.ru/images/articles/uml/seq_5.gif | **Entity** – Класс-сущность - обычно применяется для обозначения классов, которые хранят некую информацию о бизнес-объектах (соответствует таблице или элементу БД) |

Также одним из основных понятий, связанных с диаграммой последовательности, является *Сообщение*.   


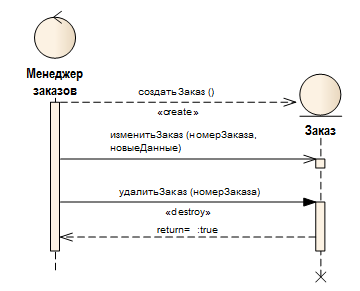
На диаграмме деятельности выделяются сообщения, инициирующие ту или иную деятельность или являющиеся ее следствием. На диаграмме состояний частично показан обмен сообщениями в рамках сообщений инициирующих изменение состояния объекта.

Диаграмма последовательности объединяет диаграмму деятельности, диаграмму состояний и диаграмму классов.

Таким образом, на диаграмме последовательности мы можем увидеть следующие аспекты:

* Сообщения, побуждающие объект к действию
* Действия, которые вызываются сообщениями (методы) – зачастую это передача сообщения следующему объекты или возвращение определенных данных объекта
* Последовательность обмена сообщениями между объектами

 Итак, прием сообщения инициирует выполнение определенных действий, направленных на решение отдельной задачи тем объектом, которому это сообщение отправлено. Сообщение в большинстве случаев (за исключением диаграмм, описывающих концептуальный уровень Системы) это вызов методов отдельных объектов, поэтому для корректного исполнения метода в сообщении необходимо передать какие-то данные и определить, что мы хотим видеть в ответ. При именовании сообщения на уровне проектирования реализации системы в качестве имени сообщения следует использовать имя метода.



В UML различают следующие виды сообщений:

* синхронное сообщение (**synchCall**) - соответствует синхронному вызову операции и подразумевает ожидание ответа от объекта получателя. Пока ответ не поступит, никаких действий в Системе не производится.
* асинхронное сообщение (**asynchCall**) - которое соответствует асинхронному вызову операции и подразумевает, что объект может продолжать работу, не ожидая ответа.
* ответное сообщение (**reply**) – ответное сообщение от вызванного метода. Данный вид сообщения показывается на диаграмме по мере необходимости или, когда возвращаемые им данные несут смысловую нагрузку.
* потерянное сообщение (**lost**) – сообщение, не имеющее адресата сообщения, т.е. для него существует событие передачи и отсутствует событие приема
* найденное сообщение (**found**) – сообщение, не имеющее инициатора сообщения, т.е. для него  существует событие приема и отсутствует событие передачи

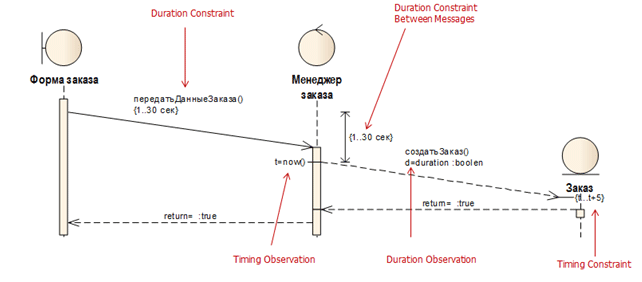
Для сообщений также доступен ряд предопределенных стереотипов. Наиболее часто используемые стереотипы это **create**и **destroy.**

Сообщение со стереотипом **create**, вызывает в классе метод, которые создает экземпляр класса. На диаграмме последовательности не обязательно показывать с самого начала все объекты, участвующие во взаимодействии. При использовании сообщения со стереотипом  **create**, создаваемый объект отображается на уровне конца сообщения.

Для уничтожения экземпляра класса используется сообщение со стереотипом **destroy**, при этом в конце линии жизни объекта отображаются две перекрещенные линии.

При отображении работы с сообщениями иногда возникает необходимость указать некоторые временные ограничения. Например, длительность передачи сообщения или ожидание ответа от объекта не должно превышать определенный временной интервал. Можно указать следующие временные параметры:

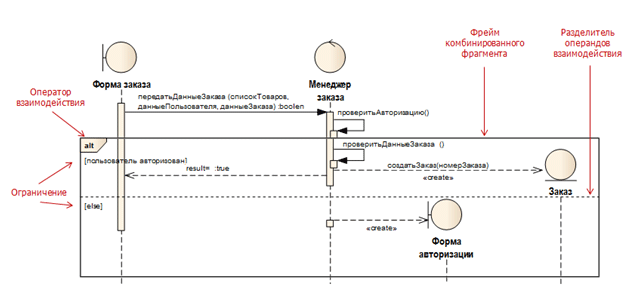
* ограничение продолжительности (Duration Constraint) – минимальное и максимальное значение продолжительности передачи сообщения
* ограничение продолжительности ожидания между передачей и получением сообщения (Duration Constraint Between Messages)
* перехват продолжительности сообщения (Duration Observation)
* временное ограничение (Timing Constraint) – временной интервал, в течение которого сообщение должно прийти к цели (устанавливается на стороне получателя)
* перехват времени, когда сообщение было отправлено (Timing Observation)



Форма заказа передает данные Менеджеру заказа, при этом передача данных не должна длиться больше 30 сек. – данное ограничение может понадобиться при выявлении требований к быстродействию Системы. Далее получение данных с формы инициирует запуск метода для создания экземпляра класса Заказ. Между получением данных от Формы заказа и инициализацией создания объекта должно пройти не более 30 сек., в противном случае пользователю может быть предоставлено сообщение об ошибке или недоступности сервера. Длительность передачи сообщения о создании объекта может быть зафиксирована в переменной d.

Данное значение может понадобиться при установке временного ограничения на получение ответного сообщения клиентом. В момент передачи сообщения фиксируется временное значение и заносится в переменную t. Таким образом, можно установить ограничение на стороне приемника, указав переменную t в качестве минимального значения и t+<допустимый интервал> в качестве максимального значения.

До появления UML 2.0 диаграмма последовательности рассматривалась только в рамках моделирования последовательности обмена сообщениями. Расширение сценария отображались с помощью ветвления линий сообщений, что не давало полной картины взаимодействия объектов. Таким образом, для целей моделирования расширения сценария, параллельности процессов или цикличности использовались диаграммы деятельности. Для решения данных задач в UML 2.0 было введено понятие фрейма взаимодействия и операторов взаимодействия.



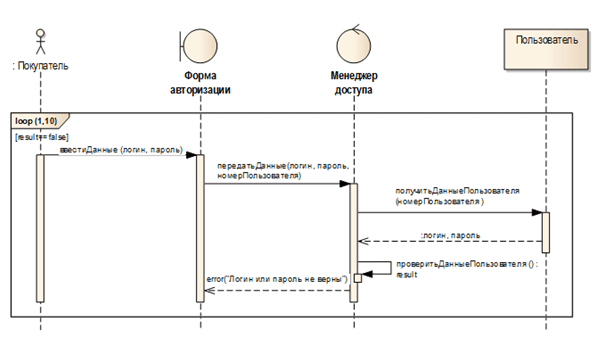
Отдельные фрагменты диаграммы взаимодействия можно выделить с помощью фрейма. Фрейм должен содержать метку оператора взаимодействия. UML содержит следующие операнды:

* **Alt -**Несколько альтернативных фрагментов (alternative); выполняется только тот фрагмент, условие которого истинно
* **Opt -**Необязательный (optional) фрагмент; выполняется, только если условие истинно. Эквивалентно **alt**с одной веткой
* **Par -**Параллельный (parallel); все фрагменты выполняются параллельно
* **loop -**Цикл (loop); фрагмент может выполняться несколько раз, а защита обозначает тело итерации
* **region -**Критическая область (critical region); фрагмент может иметь только один поток, выполняющийся за один прием
* **Neg -**Отрицательный (negative) фрагмент; обозначает неверное взаимодействие
* **ref -**Ссылка (reference); ссылается на взаимодействие, определенное на другой диаграмме. Фрейм рисуется, чтобы охватить линии жизни, вовлеченные во взаимодействие. Можно определять параметры и возвращать значение
* **Sd -**Диаграмма последовательности (sequence diagram); используется для очерчивания всей диаграммы последовательности, если это необходимо.

При использовании фрагмента условного операнда фрейм должен содержать условие для ограничения взаимодействия. При использовании условного или параллельного операнда фрейм делится на регионы взаимодействия с помощью разделителя операторов взаимодействия.

К условным операндам относятся alt и opt. Операнд alt используется при моделировании расширения сценария, т.е. при наличии альтернативного потока взаимодействия. Оператор opt используется, если сообщение должно быть передано, только при истинности какого-то условия. Данный фрейм используется без разделения на регионы.

Параллельность потоков взаимодействия можно изобразить с помощью операнда par. Внутри фрейма моделируются потоки взаимодействия в отдельных регионах.

Цикличность потока взаимодействия может быть представлена на диаграмме последовательности с помощью операнда loop. При  использовании оператора цикла можно указать минимальное и максимальное число итераций. Также фрейм должен содержать условие, при наступлении которого взаимодействие повторяется.   
   
Диаграммы последовательности предназначены для моделирования взаимодействия между несколькими объектами.  Зачастую диаграммы последовательности создаются для моделирования взаимодействия в рамках одного прецедента.

На концептуальном уровне можно использовать диаграммы последовательности для моделирования взаимодействия между Бизнес-актерами, но зачастую подобные диаграммы обрастают лишними подробностями и плохо читаются. На данном уровне лучше подойдут диаграммы деятельности, исключение составляют случае, когда необходимо смоделировать обмен сообщениями между двумя независимыми Системами.

Также диаграммы последовательности подойдут для моделирования взаимодействия пользователя и Системы в целом.

На уровне детальной спецификации требований диаграммы последовательности используются для моделирования взаимодействия компонентов Системы и пользовательских классов в рамках выбранного прецедента.

Задание на лабораторную работу:

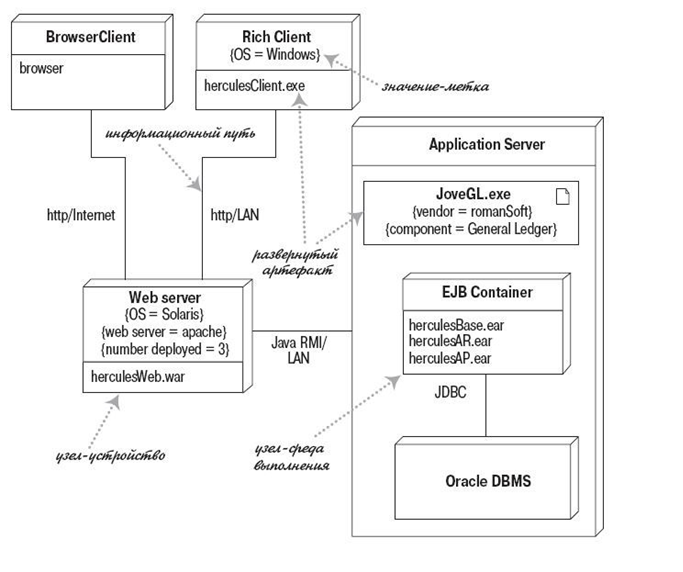
1. Разработать диаграмму последовательности.
2. Написать отчет о проделанной работе.
3. Пролистать дальше и найти вторую часть лабораторной работы.

Лабораторная работа №4.2

Цель: научиться создавать диаграмму развертывания.

**Диаграмма развертывания** представляет физическое расположение системы, показывая, на каком физическом оборудовании запускается та или иная составляющая программного обеспечения. Диаграмма развертывания очень проста, поэтому будем кратки.  
  
На рисунке показан пример простой**диаграммы развертывания**. Главными элементами диаграммы являются узлы, связанные информационными путями. Узел (*node*) – это то, что может содержать программное обеспечение. Узлы бывают двух типов. Устройство (*device*) – это физическое оборудование: компьютер или устройство, связанное с системой. Среда выполнения (*execution environment*) – это программное обеспечение, которое само может включать другое программное обеспечение, например операционную систему или процессконтейнер.  
  
Узлы могут содержать артефакты (*artifacts*), которые являются физическим олицетворением программного обеспечения; обычно это файлы.

Такими файлами могут быть исполняемые файлы (такие как файлы .exe, двоичные файлы, файлы DLL, файлы JAR, сборки или сценарии) или файлы данных, конфигурационные файлы, HTML-документы и т. д. Перечень артефактов внутри узла указывает на то, что на данном узле артефакт разворачивается в запускаемую систему.  
  
Артефакты можно изображать в виде прямоугольников классов или перечислять их имена внутри узла. Если вы показываете эти элементы в виде прямоугольников классов, то можете добавить значок документа или ключевое слово «*artifact*». Можно сопровождать узлы или артефакты значениями в виде меток, чтобы указать различную интересную информацию об узле, например поставщика, операционную систему, местоположение – в общем, все, что придет вам в голову.  
  
Часто у вас будет множество физических узлов для решения одной и той же логической задачи. Можно отобразить этот факт, нарисовав множество прямоугольников узлов или поставив число в виде значения-метки.



На рисунке обозначены три физических веб-сервера с помощью метки number deployed (количество развернутых), но это не стандартная метка.

Артефакты часто являются реализацией компонентов. Это можно показать, задав значения метки внутри прямоугольников артефактов.

Информационные пути между узлами представляют обмен информацией в системе. Можно сопровождать эти пути информацией об используемых информационных протоколах.

Здесь представлен основной набор символов**диаграммы развертывания**, необходимый для того, чтобы суметь прочитать диаграмму и создать свою.

|  |  |  |
| --- | --- | --- |
| **Термин** | **Изображение** | **Описание** |
| Узел (node) | Узел диаграммы развертывания UML | Узел (*node*) – это то, что может содержать программное обеспечение. Узлы бывают двух типов. Устройство (device) – это физическое оборудование: компьютер или устройство, связанное с системой. Среда выполнения (*executionenvironment*) – это программное обеспечение, которое само может включать другое программное обеспечение, например операционную систему или процесс контейнер. |
|  |  |  |
| Устройство (device) | Устройство (device) диаграммы развертывания UML | Устройство (*device*) – это физическое оборудование: компьютер или устройство, связанное с системой. |
|  |  |  |
| Среда выполнения (execution environment) | Среда выполнения (execution environment) диаграммы развертывания UML | Среда выполнения (*executionenvironment*) – это программное обеспечение, которое само может включать другое программное обеспечение, например операционную систему или процесс контейнер. |
|  |  |  |
| Артефакты (artifacts) | Артефакты (artifacts) диаграммы развертывания UML | Являются физическим олицетворением программного обеспечения; обычно это файлы. Такими файлами могут быть исполняемые файлы (такие как файлы .exe, двоичные файлы, файлы DLL, файлы JAR, сборки или сценарии) или файлы данных, конфигурационные файлы, HTML документы и т. д. |
| Информационный путь | Информационный путь диаграммы развертывания UML | Информационные пути между узлами представляют обмен информацией в системе. Можно сопровождать эти пути информацией об используемых информационных протоколах. |
|  |  |  |
| Значение-метка | Значение-метка диаграммы развертывания UML | При помощи значений-меток можно показать реализацию компонентов. |

Задание на лабораторную работу №4.2:

1. Разработать диаграмму развертывания.
2. Написать отчет о проделанной работе.