Оглавление

Лабораторная работа №1. Создание бизнес-модели вариантов использования, бизнес-объектной модели.

Концепции моделирования бизнес-процессов

UML (Unified Modeling Language) – унифицированный язык моделирования.

Объектом моделирования бизнес-процессов является деятельность организации как системы. Это изучение деятельности организации. При моделировании исследуется структура организации, роли сотрудников в этой организации и взаимосвязи между сотрудниками. Кроме того, анализируются рабочие потоки в организации – главные процессы в ее деятельности.

Рассмотрим основные понятия и концепции моделирования бизнес-процессов. Такие понятия, как бизнес-роль, сотрудник или диаграмма Деятельности, позволят хорошо понять саму организацию. Мы обсудим следующие концепции:

* Бизнес-роль (Business actor)
* Сотрудник/исполнитель (Business worker)
* Бизнес-вариант использования (Business use case)
* Диаграмма Бизнес-вариантов использования (Business Use Case diagram) (Отношения между бизнес-ролями и бизнес-вариантами использования)

- Бизнес-сущность (Business entity)
- Диаграмма Деятельности (Activity diagram)

Еще раз отметим, что моделирование бизнес-процессов не акцентирует то, что будет или не будет автоматизировано (хотя информацию такого рода можно получить из рабочих потоков). Вместо этого акцентируются две области. Во-первых, границы влияния организации и круг людей, с которыми должен взаимодействовать сотрудник. Во-вторых, исследуются рабочие потоки внутри организации и возможности по их оптимизации.

Бизнес-роль (Business Actor) исполняет любой человек или подразделение, которые являются внешними по отношению к организации, но взаимодействуют с ней. Например, бизнес-роли по отношению к организации играют ее клиенты, кредиторы, инвесторы и поставщики. Все эти бизнес-роли интересны с точки зрения деятельности организации.

В UML бизнес-роль в модели показана следующим значком с названием роли (внизу пишется):

Клиент

Хотя значок изображает человека, бизнес-роль необязательно является индивидуальной. Это может быть группа людей или компания. Бизнес-роль моделируется для того, чтобы понять, кто и что взаимодействует с бизнесом и как это воздействует на бизнес. При обратном проектировании или построении новой системы необходимо всегда помнить о том, что организация обязана соответствовать требованиям внешних сущностей.

Сотрудник (Business Worker) — это роль внутри организации (не должность). Один человек может играть несколько ролей, занимая одну должность. Анализ роли вместо должности связан с тем, что должности время от времени меняются, хотя роли сотрудников

более постоянны.

В UML сотрудник показан в модели следующим значком с названием роли (внизу пишется):

Торговый представитель

Мы моделируем сотрудников, чтобы понять их роли в бизнесе и соотношение между такими ролями. Выявление всех сотрудников поможет выяснить ответственности, возлагаемые на каждую из ролей, необходимый уровень квалификации для исполнения роли и другие детали.

Как минимум необходимо рассмотреть следующие вопросы:

- Каковы ответственности сотрудника?
- Какую квалификацию должен иметь сотрудник для исполнения своей роли?
- Как он взаимосвязан с другими сотрудниками?
- В каких рабочих потоках он участвует?
- Каковы обязанности сотрудника в каждом из рабочих потоков?

Бизнес-вариант использования

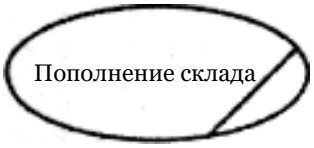
Бизнес-вариант использования (Business Use Case) — это группа связанных рабочих потоков внутри организации, предоставляющая данные для бизнес-роли. Другими словами, бизнес-вариант использования описывает деятельность организации, точнее то, что делается в организации для ведения бизнеса и тех, кто ведет этот бизнес. Набор всех бизнес-вариантов использования для организации должен полностью описывать бизнес этой организации.

Примерами бизнес-вариантов использования для магазина розничной торговли могут служить "Пополнить склад", "Назначить цену", "Продать товар", "Возвратить деньги" или "Доставить товар". Примеры для электронного бизнеса: "Зарегистрировать нового пользователя", "Создать/Изменить заказ", "Заполнить заказ", "Пополнить склад" или "Отменить заказ". Для инвестиционной компании дополнительно появятся "Купить акции" и "Продать акции".

Компании не всегда необходима полная автоматизация для работы с бизнес-вариантами использования. Для владельца ранчо вполне допустимы бизнес-варианты использования "Купить скот", "Продать скот", "Разлить молоко по бутылкам" или "Пополнить запас кормов".

В UML для бизнес-вариантов использования применяется следующий значок с названием деятельности (пишется внизу или внутри):

Пополнение склада



Обычно названия бизнес-вариантов использования имеют формат <глагол> <существительное>, например "Назначить цену" — "Price Products". Такой стандарт используется по нескольким причинам. Названия становятся унифицированными даже в проектах, где заняты множество аналитиков. Кроме того, названия проще понять, и, что важнее всего, название сконцентрировано на бизнесе, т.е. на исполняемой операции, а не сущности.

Разумеется, название "Назначить цену" не дает никаких сведений о деталях проведения такой операции. Для любого бизнес-варианта использования требуется сформировать некоторое описание, позволяющее людям понять то, что выполняется в таком варианте. Должен ли клерк использовать предыдущие цены при назначении Новой цены? Нужно ли обратиться к отчету, чтобы определить возможности клиентов по оплате товара? Потребуется ли провести подробное исследование о текущем уровне цен в Египте и Тунисе, чтобы вычислить среднее значение? Либо следует просто исходить из закупочной цены товара? Ответы на эти вопросы не известны, пока не документирован определенный рабочий поток в организации.

Документировать рабочий поток можно двумя способами. В некоторых случаях проще всего создать нумерованный пошаговый список того, что происходит во время прохода через вариант использования:

1. Клерк запрашивает у менеджера список всех новых товаров, для которых требуется назначить цены.

2. Клерк изучает закупочную ведомость, чтобы определить закупочную стоимость каждого товара.

3. Клерк начисляет 10% к закупочной цене, чтобы определить цену товара.

4. Клерк отправляет список новых цен на утверждение менеджеру.

5. Если менеджер не утверждает цену, то совместно с клерком он должен изменить предложенную цену товара.

6. Клерк формирует ведомость цен для всех новых товаров.

7. Клерк проставляет в ведомости цены на все новые товары.

Проблемы в этом случае возникают, когда присутствует большой объем логических условий, которые будут сбивать с толку аналитиков. В приведенном выше простом примере последовательность операций очевидна и понятна, но, к сожалению, реальные бизнес-процессы гораздо сложнее. Сотрудник обычно предпринимает некоторые действия при возникновении варианта А, но совсем иные действия в варианте В, причем может существовать еще и вариант С. В этих случаях лучше использовать диаграммы Деятельности.

Диаграмма Бизнес-вариантов использования

На диаграммах этого типа показаны бизнес-варианты использования, бизнес-роли и сотрудники организации, а также отношения между ними. Это дает полную модель хозяйственной деятельности компании и описание внешних по отношению к ней сущностей. Диаграмма Бизнес-вариантов использования описывает зону действия организации, т.е. ее окружение и границы. Пример ---------->

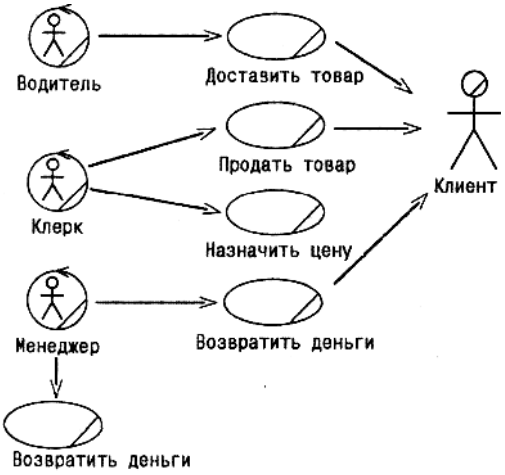


Диаграмма проста. Она предназначена для быстрого изучения высокоуровневых отношений в бизнесе без показа всех подробностей и информации, которая может затруднить изучение взаимосвязей между бизнес-ролями. Для большого количества бизнес-вариантов использования можно создать несколько диаграмм, каждая из которых содержит собственное подмножество полного набора бизнес-вариантов использования.

Стрелки от бизнес-ролей или сотрудников к вариантам использования показывают, кто инициирует данный вариант использования. В примере клерк (сотрудник организации) начинает процесс назначения цены товара. Стрелка от варианта использования к бизнес-роли показывает, что организация инициирует коммуникацию с данной ролью. В данном примере во время рабочего потока "Доставка товара" организация (в данном случае водитель) взаимодействует с клиентом.

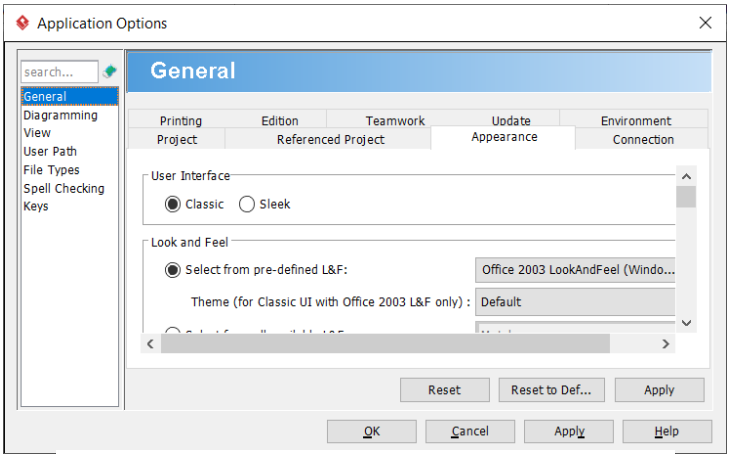


Рис. 1. Изменение режима в Visual Paradigm

Перед созданием диаграмм в Visual Paradigm лучше переключить программу в режим классического вида, если текущий режим другой: пункт меню Window –> Application Optioins –> General –> Appearance –> User Interface –> Classic (рис. 1). Изменения вступят в силу после перезапуска программы.

Для создания диаграммы Бизнес-вариантов использования в Visual Paradigm в левой панели Diagram Navigator вызываем контекстное меню пункта Use Case Diagram и в нем New Use Case Diagram (рис. 2).

Для создания бизнес-роли используем значок Actor в панели инструментов, вытаскиваем его на рабочее поле, меняем подпись на нужную, в контекстном меню объекта выбираем Open Specification, в появившемся окне ставим галочку в поле Business Mode.

Для создания Сотрудника делаем то же самое, но еще добавляем к объекту овал (на панели ниже сиреневый объект в виде овала), меняем его размер по размеру ранее созданного объекта Actor, в контекстном меню овала выбираем Order –> Send to Back, выделяем оба объекта и группируем их (контекстное меню после выделения обоих объектов, Grouping –> Group). Теперь это один объект.

Для добавления Бизнес-варианта использования выбираем в панели инструментов Use Case и переносим на рабочее поле. В контекстном меню объекта выбираем Open Specification, в появившемся окне ставим галочку в поле Business Mode.

После этого можно соединять объекты линиями: можно выбрать объект Association в панели инструментов и провести между нужными объектами, или можно с зажатой правой кнопкой мыши провести от одного объекта до другого. Также можно воспользоваться специальным меню объекта (рис. 3, в зеленой рамке). Такое меню есть у всех объектов. Нужно за него потянуть до нужного объекта и отпустить, в появившемся окошке выбираем обычную линию. Чтобы линия стала стрелкой, нужно у одного из ее концов в контекстном меню в пункте Navigable выбрать Unspecified.

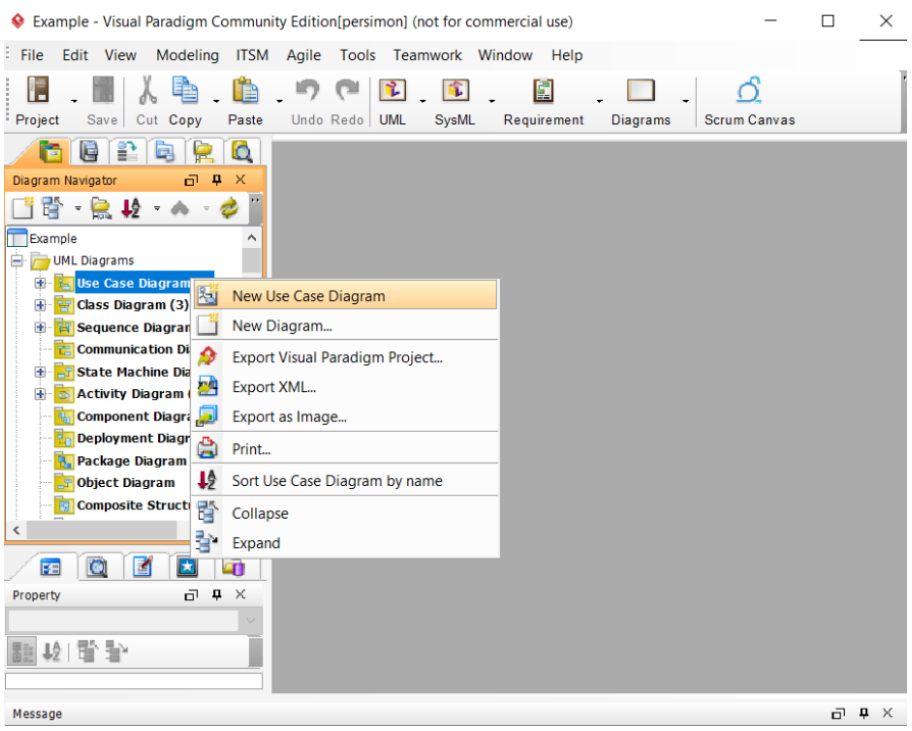


Рис. 2. Создание диаграммы Бизнес-вариантов использования.

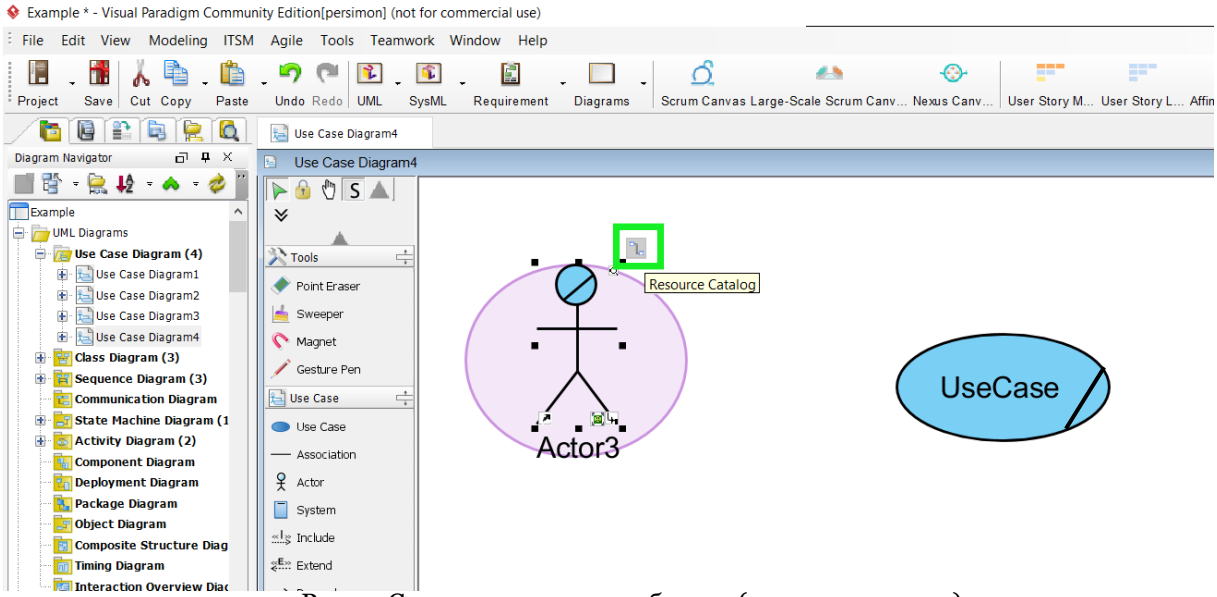


Рис. 3. Специальное меню объекта (внутри зеленого).

Диаграмма Деятельности показывает в графической форме этапы рабочего процесса, последовательность операций и тех, кто отвечает за их выполнение.

Диаграмма Деятельности — это способ графического изображения модели рабочих потоков для вариантов использования. Диаграмма показывает шаги (этапы) рабочего потока, точки принятия решений в этом потоке, ответственности для каждого этапа и объекты, влияющие на рабочий поток.

Основными элементами такой диаграммы являются:

- Дорожки, показывающие ответственности за изображенные на диаграмме задачи
- Деятельности, демонстрирующие этапы рабочих потоков
- Действия, показывающие шаги в пределах деятельности. Действие может происходить

при входе в деятельность, во время ее исполнения, при выходе из деятельности или при возникновении определенного события

• Бизнес-объекты, являющиеся сущностями, воздействующими на рабочий поток

• Переходы, показывающие движение рабочего потока между деятельностями

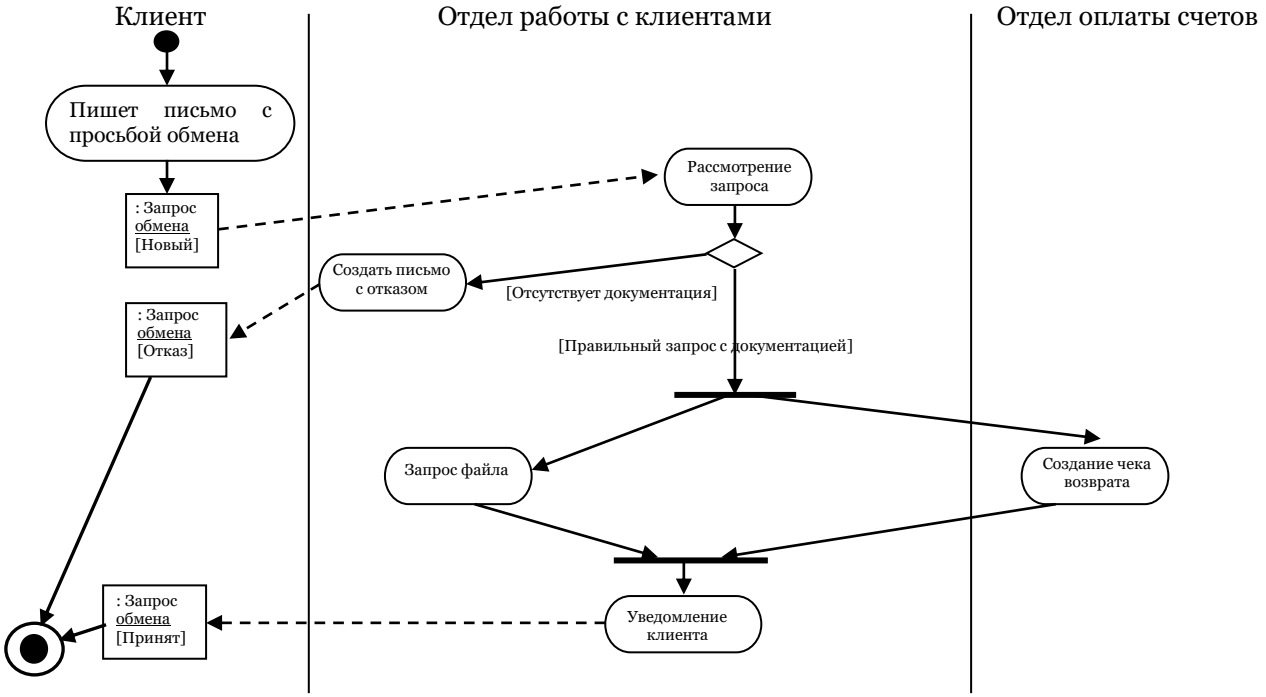
• Точки принятия решений, демонстрирующие решения, которые необходимо принять в пределах рабочего потока

• Синхронизации, иллюстрирующие два или более одновременных этапа в пределах рабочего потока

• Исходное состояние, определяющее начало рабочего потока

• Конечное состояние, определяющее завершение рабочего потока

Пример диаграммы Деятельности. В данном случае, клиент получает дефектный товар и просит его обменять.



Начнем чтение диаграммы: клиент (customer) начинает процесс с написания письма с запросом на обмен товара. Представитель отдела работы с клиентами (customer service representative) получает письмо. Если отсутствует необходимая документация, представитель отдела работы с клиентами пишет и посылает клиенту уведомление об отказе (rejection notice) на запрос. Если же на данный товар есть документация, то представитель отдела работы с клиентами регистрирует запрос, причем одновременно клерк по счетам к оплате (accounts payable) пишет соответствующий счет. Когда обе эти операции завершены, представитель отдела работы с клиентами уведомляет клиента о принятии к исполнению его запроса.

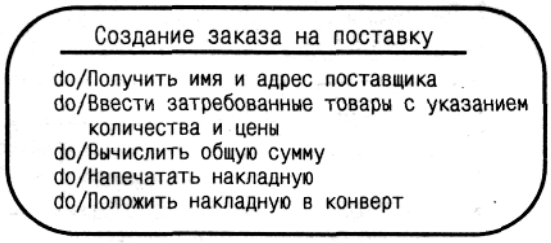
Обсудим нотацию на диаграмме. В начале (в левой верхней части диаграммы) находится закрашенная точка, отмечающая начальное состояние. Этот символ показывает начало процесса.

Скругленные прямоугольники на диаграмме отмечают действия, т.е. единичные этапы рабочего потока. Это задачи, исполняемые сотрудниками. Диаграмма разделена на три вертикальные дорожки, в верхних частях которых указаны роли, связанные со всеми действиями в пределах каждой из них.

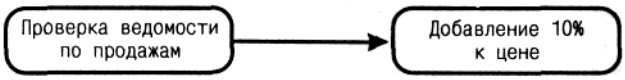
Внутри действия можно перечислить отдельные операции, т.е. этапы выполнения действия. Например, для действия "создать заказ на поставку" операциями могут быть: "получить имя и адрес поставщика", "ввести наименование и цену", "подсчитать итог" и "вывести на печать". Эти этапы слишком малы, чтобы попасть на общую диаграмму Деятельности высокого уровня, но могут предоставить дополнительную информацию о процессе.

Существует четыре типа действий:

- Возникающие во время входа в действие. Маркируются словом entry (вход).
- Возникающие в процессе деятельности. Являются этапами внутри активности. Маркируются словом do (сделать).
- Возникающие при выходе из деятельности. Маркируются словом exit (выход).
- Возникающие при определенном событии. Маркируются словом event (событие).



Стрелки, соединяющие деятельности, называются переходами (transition). Переход показывает, какая деятельность выполняется первой после завершения текущей деятельности.



В данном примере, завершив проверку цены товаров, клерк начинает начисление 10% на эти цены.

Можно установить для перехода ограждающие условия, определяющие выполнение перехода. Эти условия отмечены квадратными скобками. Деятельность "создать письмо с отказом с отказом" выполняется только тогда, когда истинно граничное условие "отсутствует документация". Горизонтальные полосы называются синхронизациями. Они позволяют показать, что две или более деятельностей выполняются одновременно. Верхняя синхронизация показывает управляемое разделение рабочего потока на две ветви. По завершении последующих деятельностей возникает следующая синхронизация, называемая схождением (join). После схождения рабочий поток снова становится единым управляемым потоком. Полосы синхронизации могут быть горизонтальными или вертикальными. В примере на рисунке представитель отдела работы с клиентами регистрировал запрос одновременно с выпиской счета клерком по счетам к оплате. Только по завершении обеих этих деятельностей клиенту отправляется уведомление.

Квадраты на диаграмме показывают объекты. На эти объекты воздействует рабочий поток, и объекты изменяют свое состояние. В данном примере запрос может быть новым, отвергнутым или утвержденным. Пунктирными линиями показаны деятельности, влияющие на состояние объекта, например, создание уведомления об отказе устанавливает для запроса состояние "отвергнут".

Бизнес-сущности (понятия) (Business entity) — это объекты, используемые организацией для проведения своего бизнеса. К таким сущностям относятся понятия, с которыми ежедневно имеют дело сотрудники организации, например, заказы, счета, отгрузочная тара, контракты, а также скрепки и кнопки — все, что имеет отношение к бизнесу.

Еще раз вернемся к последнему определению. Необходимо перечислить основные элементы, с которыми связан бизнес и без которых он невозможен. В бизнесе по производству скрепок, скрепка становится важным элементом бизнеса, но в других случаях можно не обращать на нее внимания и не учитывать этот элемент. При идентификации бизнес-сущностей ищут ответы на следующие вопросы:

Какие товары производит компания?

Какие услуги предоставляет компания?

Какие элементы компания приобретает для своей хозяйственной деятельности?

Какие элементы компания получает и отправляет своим клиентам?

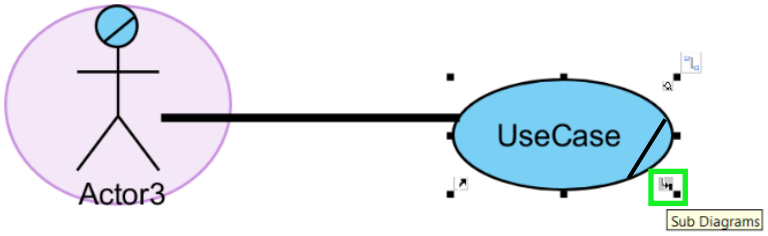
Какие элементы передаются между сотрудниками компании?

Можно проанализировать существительные в названиях уже описанных бизнес-вариантов использования. По большей части они соответствуют бизнес-сущностям. Для бизнес-сущностей используется следующий значок:

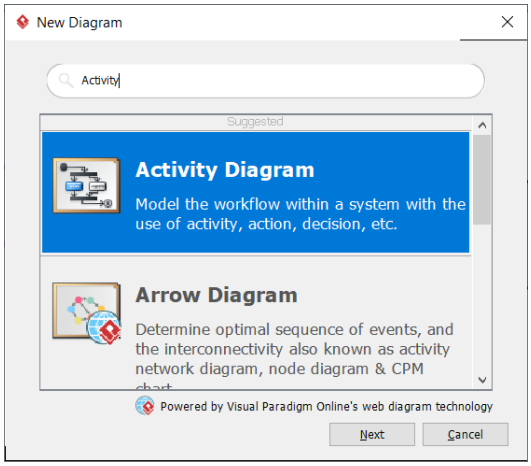
: Счет
[Пополнение склада]

Возле двоеточия – название объекта, в квадратных скобках на следующей строке – состояние объекта.

Для создания диаграммы деятельности в Visual Paradigm для конкретного бизнес-варианта использования, можно использовать нижнее специальное меню объекта (рис. 4а в зеленом квадрате). В меню выбираем пункт New Diagram, в появившемся окне в строке поиска пишем Activity и выбираем Activity Diagram (рис. 4б), в следующем окне жмем Next, в следующем можно поменять название диаграммы и жмем Ok. Появится рабочее поле и панель инструментов для создания диаграммы деятельности.



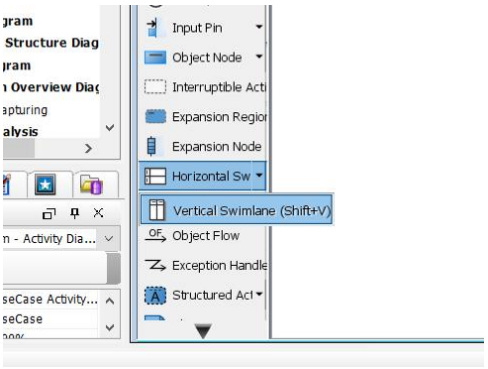
а) выбор меню



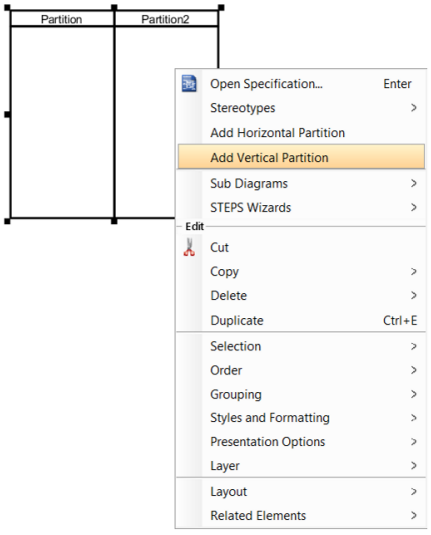
б) выбор диаграммы

Рис. 4. Специальное меню для создания связанной диаграммы.

Для начала нужно создать дорожки для потоков деятельности. В панели инструментов выбираем Vertical Swimlane (в одной вкладке с Horizontal Swimlane – рис. 5а). Помещаем на поле и добавляем еще одну дорожку (рис. 5б), меняем названия дорожек и их размеры.



а) выбор вида дорожек



б) добавление дорожки

Рис. 5. Создание дорожек для потоков деятельности.

Итоговый вид дорожек должен быть примерно как на рис. 6. Далее можно добавлять объекты на дорожки. Начинается все с объекта Initial Node – точка инициализации – начало диаграммы. Перетаскиваем ее на дорожку клиента.

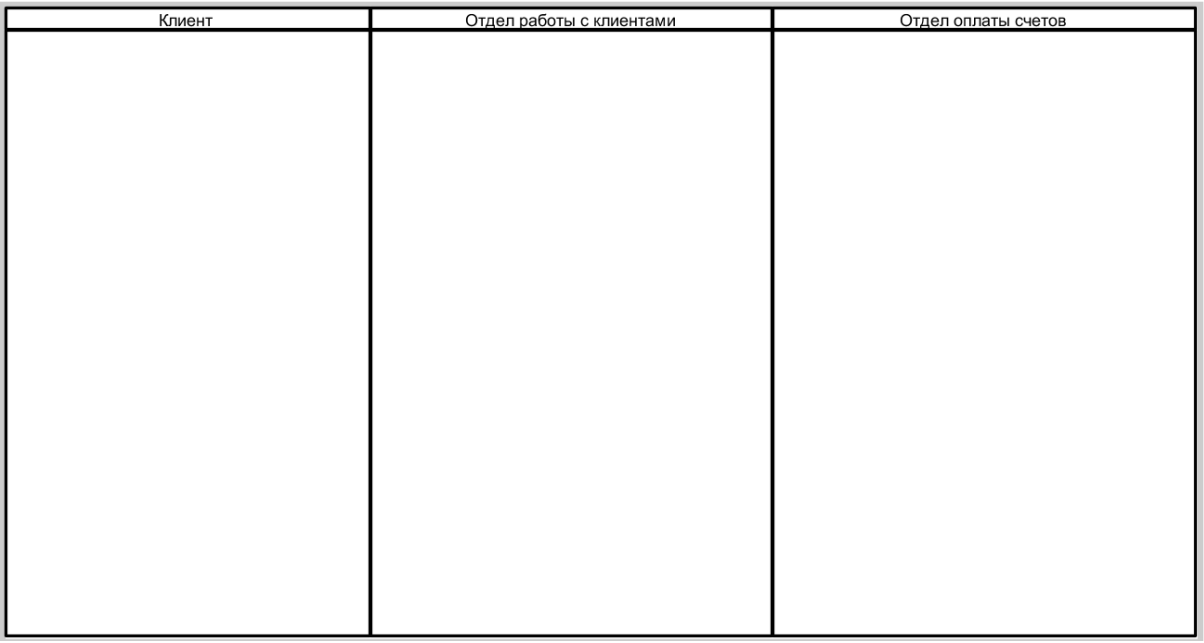


Рис. 6. Итоговый вид пустых дорожек.

Далее можно из верхнего специального меню точки инициализации вытащить нужный объект (рис. 7). Можно выбрать Activity или Action. И меняем надпись на нужную.

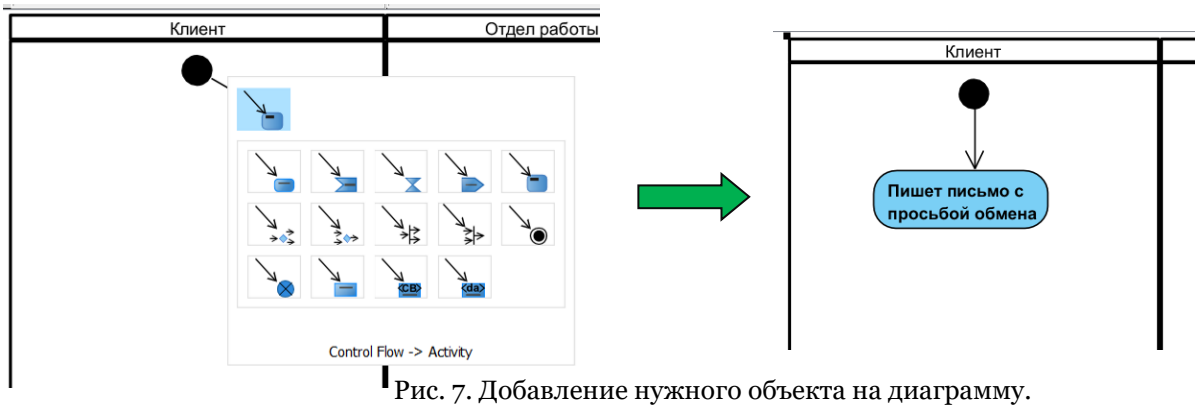


Рис. 7. Добавление нужного объекта на диаграмму.

Тот же объект можно вытащить и из панели инструментов. Остальные объекты согласно рисунку на странице 5 добавляются аналогично на нужную дорожку (учитывайте, что на панели инструментов несколько объектов могут скрываться в одной вкладке со стрелкой ▾).

Есть один нюанс с объектами Fork и Join для разделения и соединения параллельных действий (они на одной вкладке с объектом Decision Node): они изначально вставляются в вертикальном виде, вид можно поменять в контекстном меню объекта Orientation –> Horizontal.

Задание: 1. Создать бизнес-модель (диаграмму бизнес-вариантов использования) по своему варианту.

2. Создать диаграммы деятельности для всех бизнес-вариантов использования.

Лабораторная работа №2. Разработка требований. Диаграмма прецедентов

Анализ требований похож на бизнес-моделирование, но есть и отличия. Если бизнес-моделирование делает акцент на саму организацию, то в системном моделировании основное внимание нацелено на разрабатываемую систему.

	Бизнес-моделирование	Анализ требований
Вариант использования Use Case	описывает особенности бизнеса	описывает действия системы в контексте бизнеса
Действующее лицо Actor	внешнее по отношению к организации	внешнее по отношению к системе (может принадлежать организации)
Сотрудник	принадлежит организации	не используется

Диаграмма прецедентов (Use case diagram) Этот вид диаграмм позволяет создать список операций, которые выполняет система. Часто этот вид диаграмм называют диаграммой функций, потому что на основе набора таких диаграмм создается список требований к системе и определяется множество выполняемых системой функций.

Данный тип диаграмм используется при определении требований к будущей программной системе. Отражает объекты как системы, так и предметной области, и задачи, ими выполняемые.

Действующие лица – это все, кто взаимодействует с разрабатываемой системой. Варианты использования описывают все, что попадает в зону действия системы, а действующие лица – все, что находится вне границ действия системы.

Существуют три основных типа действующих лиц: пользователи системы, другие системы, взаимодействующие с данной, и время.

Первый тип – это физическое лицо или пользователь. Присваивая ему имя нужно отражать роль, а не должность.

Прецедент или вариант использования (use case) – это описание последовательности выполняемых системой действий, которая производит наблюдаемый результат, значимый для какого-то определенного актера (actor). Каждый Use case – это описание сценария поведения, которому следуют действующие лица (Actors).

Анализ вариантов использования позволит клиенту увидеть то, что он получит от системы, и согласовать границы действия системы в самом начале разработки проекта.

Чтобы выявить варианты использования лучше начать с анализа документации, предоставляемой заказчиком. Следует обратить внимание на участников проекта и постараться выяснить, что каждый из них ожидает от готового продукта. Для каждого участника проекта нужно выяснить:

– как он будет пользоваться системой?
– будет ли он работать с информацией (создавать, читать, обновлять, удалять)?
– будет ли он информировать систему о внешних событиях?
– должна ли система уведомлять его о каких-либо изменениях или событиях?

Варианты использования не зависят от реализации. Они заостряют внимание на том, что должна делать система, а не на том, как она будет это делать. Набор вариантов использования должен ясно и просто показывать пользователю всю систему. В типичных системах выявляется от 20 до 70 вариантов использования.

Каждый вариант использования должен представлять собой законченную транзакцию между пользователем и системой, причем результат транзакции должен иметь важное значение для пользователей.

Получив окончательный список вариантов использования, следует задуматься о его полноте и попытаться ответить на следующие вопросы:

– Существует ли для любого функционального требования хотя бы один вариант использования? Если для требования нет варианта использования, то такое требование не

будет реализовано в системе.

- Анализировалось ли использование системы каждым участником?
- Какую информацию предоставляет системе каждый из участников?
- Какую информацию получит от системы каждый из участников?
- Обсуждались ли вопросы обслуживания системы? Должны быть люди, которые запускают и останавливают систему.
- Учтены ли все внешние системы, с которыми взаимодействует разрабатываемая система?
- Какую информацию от внешних систем будет получать или отправлять разрабатываемая система?

Отношения

Ассоциативные отношения служат для показа взаимосвязей между вариантами использования и действующим лицом.

Между вариантами использования возможны отношения трех типов: **включающие, расширяющие и обобщенные.**

Между действующими лицами возможны только **обобщенные** отношения.

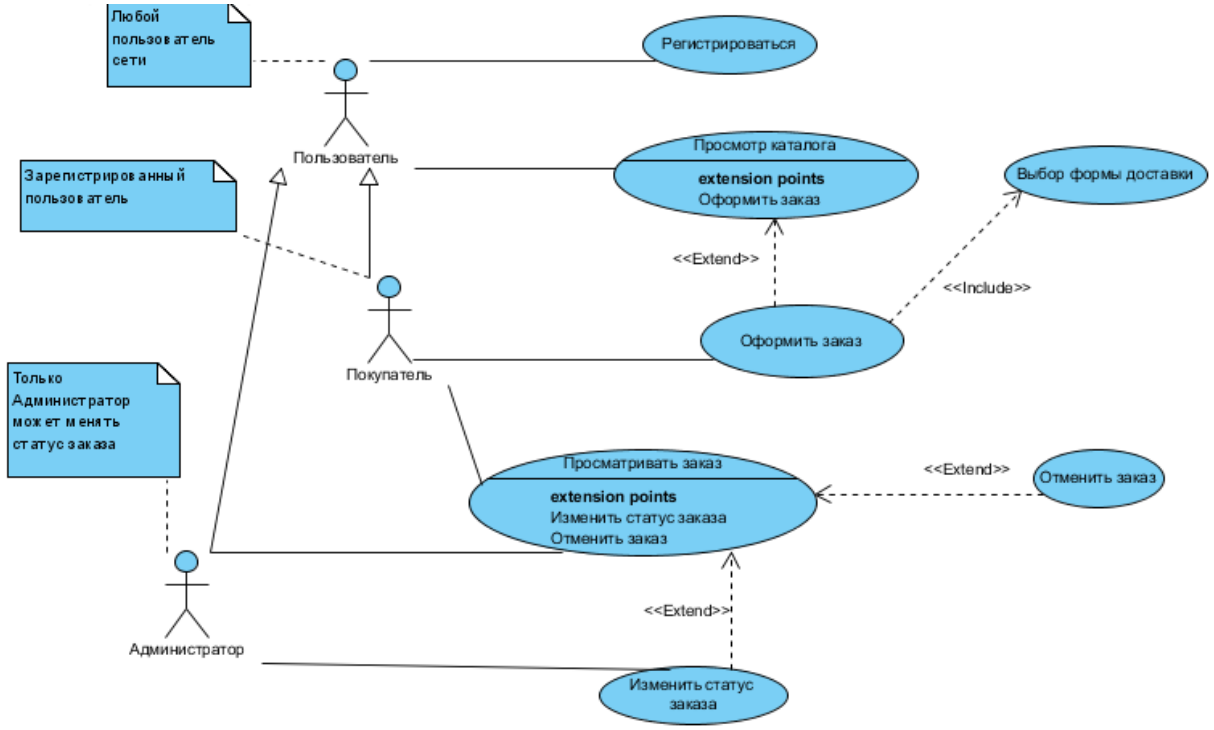
Включающие отношения позволяют одному прецеденту предоставлять функции другим прецедентам. Такие отношения необходимы в двух случаях.

Во-первых, если несколько вариантов использования имеют во многом идентичную функциональность, ее можно разделить на отдельные варианты использования, которые включаются в общий вариант использования.

Во-вторых, включающие отношения помогут в ситуации, когда один из вариантов использования обладает необычно широкой функциональностью.

Показываются пунктирной стрелкой со словом ***include*** *(от главного к подчиненному)*. Они выполняются всегда!!!

Расширяющие отношения позволяют одному варианту использования дополнить свою функциональность за счет другого. Показываются пунктирной стрелкой со словом **extend** (от подчиненного к главному). Выполняются в каких-то исключительных случаях (обработка ошибок, отмена заказа, внештатная ситуация).



Дополнительные варианты использования (extend и include) удобнее всего вытаскивать из специального верхнего меню объекта, потом можно поменять стрелки, если что.

Задание: 1. Создать диаграмму прецедентов по своему варианту.

2. Создать текстовые документы с описанием прецедентов по шаблону (см. файлы на сервере).

Лабораторная работа №3. Диаграммы взаимодействия

На диаграммах взаимодействия отображают один из процессов обработки информации в варианте использования: какие объекты нужны потоку, какими сообщениями обмениваются объекты, какие действующие лица инициируют поток и в какой последовательности отправляются сообщения.

Существует два типа диаграмм взаимодействия – диаграммы последовательности (sequence) и кооперативные диаграммы (collaboration). Диаграммы последовательности заостряют внимание на управлении и организованы по времени, а кооперативные диаграммы отображают потоки данных.

Этапы создания диаграмм: поиск объектов, поиск действующих лиц, добавление сообщений в диаграмму.

Поиск объектов: проанализировать текст задачи, выписать все существительные, убрать незначащие из них.

Объекты-сущности. Хранят информацию и часто отображаются в таблицах или полях базы данных. Большая часть существительных в потоке событий станут объектами-сущностями (рейс, пассажир или билет)

Граничные объекты. Находятся на границе между системой и внешним миром, т.е. формируют интерфейс приложения (формы, окна и т.д.). Формы могут проявиться в потоке событий, но интерфейсы обычно не входят в поток.

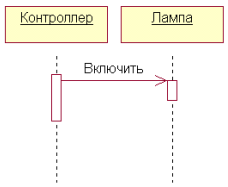
Управляющие объекты. Не несут в себе бизнес-функциональности, но координируют и управляют другими объектами в общей логике потока.

На диаграмме для каждого объекта можно задать его устойчивость (persistence).

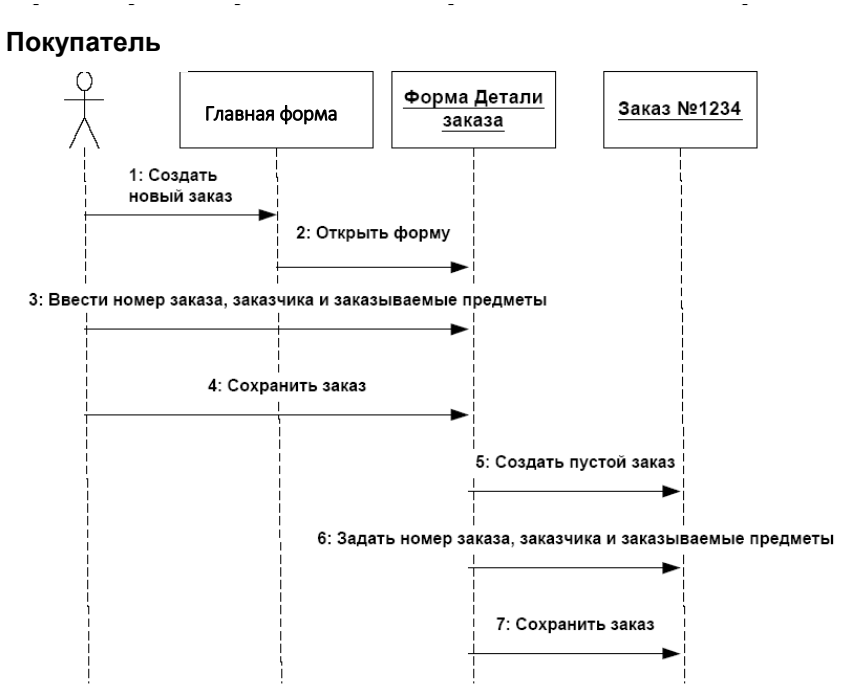
Persistent (Устойчивый) Устойчивый объект сохраняется в базе данных или другим способом, обеспечивающим постоянное хранение. Такой объект будет существовать даже после прекращения работы программы.

Static (Статичный) Статичный объект сохраняется в памяти компьютера в течении всего времени работы программы. Он, в частности, продолжает существование после выполнения отраженных на диаграмме последовательности действий, но не сохраняется после работы программы.

Transient (Временный) Временный объект сохраняется в памяти в течение очень короткого времени (например, пока не закончится выполнение процессов, определенных в диаграмме последовательности)

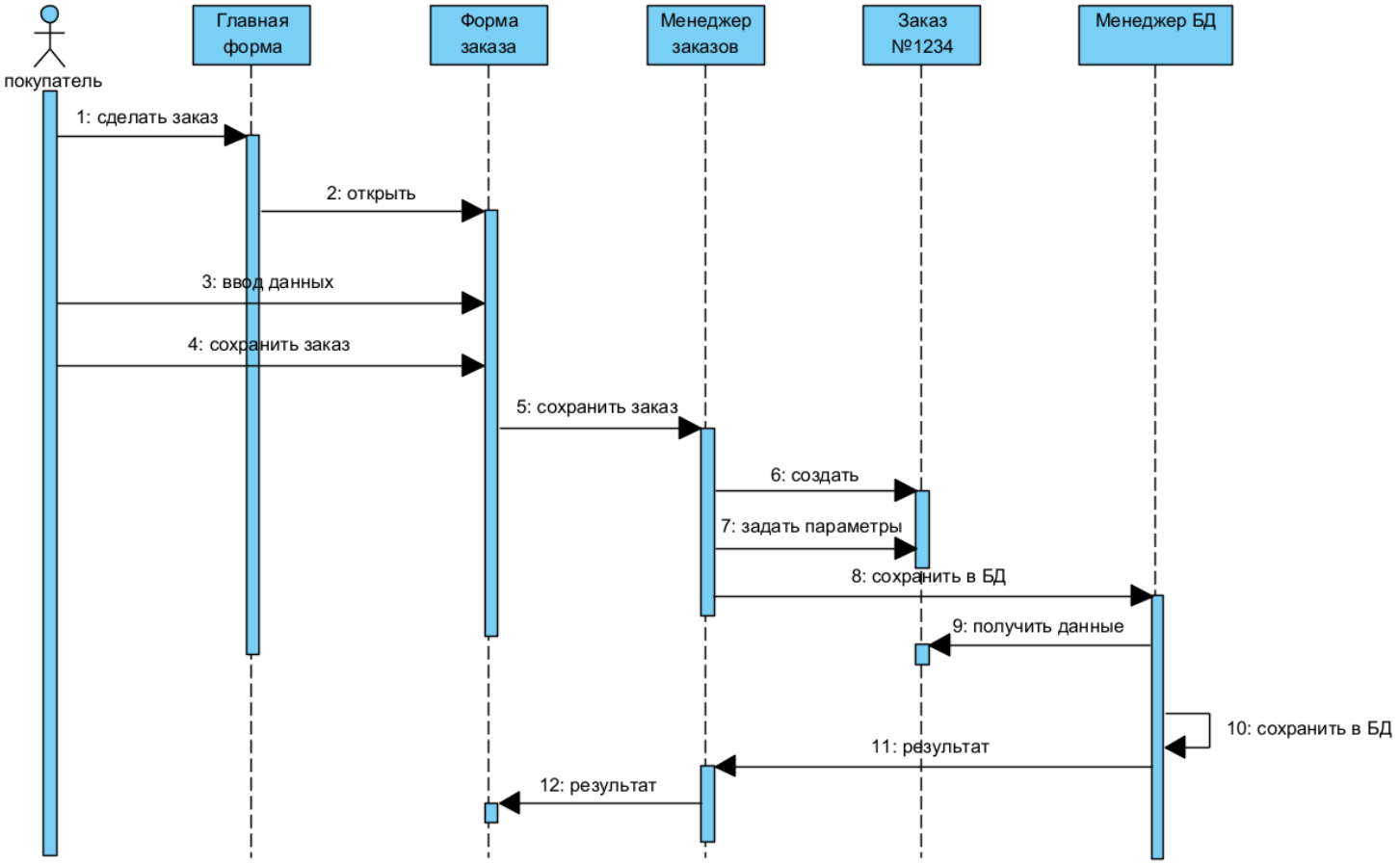


Пример диаграммы последовательностей (Sequence Diagram) прецедента «Сделать заказ» в любом электронном магазине.



Уточнение этой диаграммы

Теперь надо позаботиться об управляющих объектах и взаимодействии с базой данных. Как видно из диаграммы, объект Детали заказа имеет множество ответственностей, с которыми лучше всего мог бы справиться управляющий объект. Кроме того, новый заказ должен сохранять себя в базе данных сам. Вероятно, эту обязанность лучше было бы переложить на другой объект. Также, необходимо как-то информировать пользователя о результате выполненных действий.



Для создания диаграммы последовательностей в Visual Paradigm удобнее всего использовать нижнее правое меню нужного прецедента (рис. 8). Выбираем New Diagram и в появившемся окне начинаем печатать Seq и программа предложит нужный вид диаграмм (Sequence Diagram). В следующем окне выбираем Blank и жмем Next. В следующем окне можно поменять название диаграммы и нажать Ok.

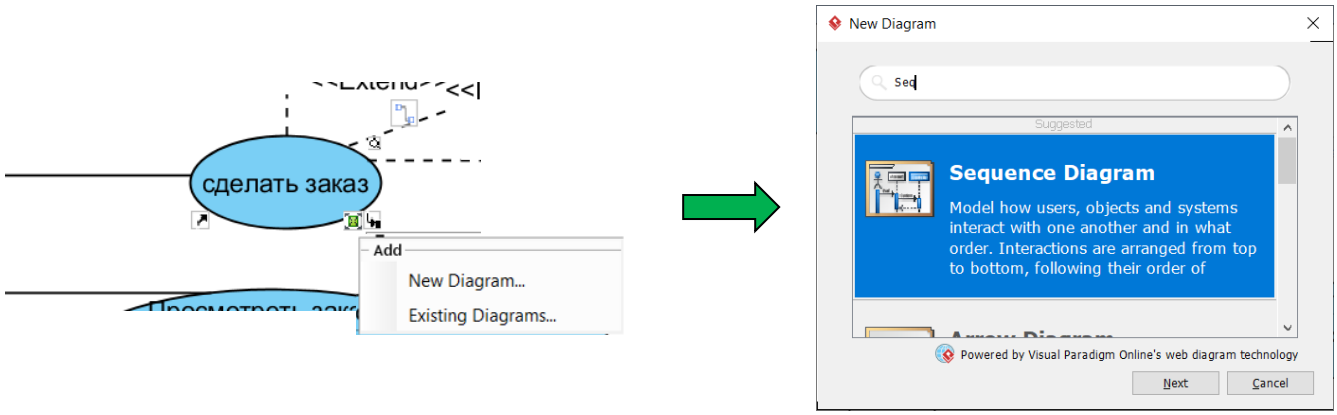


Рис. 8. Создание диаграммы последовательности (Sequence Diagram)

Появится поле для создания диаграммы. Начать следует с добавления объекта Actor и дать ему соответствующее имя (рис. 9). После этого можно из объекта «вытащить» другие объекты (LifeLine) и тоже переименовать. При этом появляется стрелка с номером, и нужно прописать название действия для стрелки (сообщения).

Таким же образом добавляются остальные объекты на диаграмму. Для изменения параметров нумерации сообщений на стрелках (по умолчанию они имеют вложенную иерархическую нумерацию 1.1, 2.1 и т.д.) нужно вызвать контекстное меню диаграммы (в любом свободном месте диаграммы) и в пункте Sequence Number выбрать Single Level, тогда нумерация будет простой, сквозной (1, 2, 3 и т.д.).

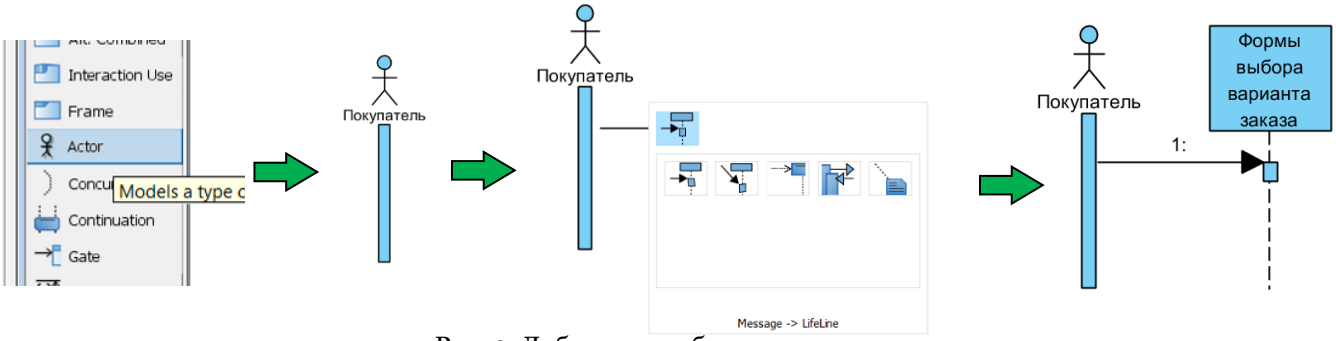


Рис. 9. Добавление объектов на диаграмму

Задание: создать диаграммы последовательностей для всех прецедентов из диаграммы прецедентов (из лаб №2).

Лабораторная работа №4. Построение логической модели

Логическая модель строится по результатам двух этапов – анализа и проектирования. На этапе анализа выявляются классы, определяются их атрибуты и операции, отношения между классами. На этапе проектирования данная модель адаптируется к определенной среде программирования.

Класс – это некоторая сущность, инкапсулирующая данные и поведение. В соответствии с традиционным подходом данные располагаются в базе данных, а поведением занимается собственно приложение. ООП предполагает объединение некоторого количества данных и поведения, обрабатывающего эти данные.

Класс в UML изображается с помощью прямоугольника, в верхней части которого содержится имя и стереотип класса. Средний раздел – атрибуты класса, т.е. его данные. В нижней секции описываются операции, т.е. поведение класса.

Выявление классов.

Выявляются классы с помощью анализа диаграмм последовательности. Нужно проанализировать все объекты этих диаграмм, выбрать объекты со схожими свойствами и действиями и создать класс, экземплярами которого будут данные объекты.

В итоге ***все*** объекты на диаграммах последовательности должны быть соотнесены с соответствующими классами. Чтобы соотнести объект и класс (уже созданным в логическом представлении), нужно в спецификации объекта выбрать из списка нужный класс. Один класс может проявляться на нескольких диаграммах последовательности или несколько раз на одной и той же диаграмме в виде разных объектов этого класса.

Каждому классу можно дать имя, определить стереотип, а также задать несколько других параметров.

Стереотип – это механизм, позволяющий категоризировать классы. Например, чтобы найти все формы в модели можно создать стереотип Form. Стереотипы помогают и в процессе генерации кода, они помогают определить тип создаваемого класса в целевом языке программирования.

При выявлении классов рассматриваются три возможных стереотипа: сущность (entity), граница (boundary) и управление (control). Не все они содержатся в потоке событий и на диаграммах взаимодействия. Классы-сущности часто необходимы для создания таблиц БД. Каждый атрибут класса-сущности становится полем в базе данных (такой подход уменьшает вероятность сбора ненужной информации). Управляющие классы называют менеджерами. Остальные классы посылают ему обычно мало сообщений, но сам он посылает много сообщений. Например, менеджер безопасности может отвечать за контроль событий, связанных с безопасностью, а класс менеджер транзакций – заниматься координацией сообщений, относящихся к транзакциям с базой данных. Могут быть и другие менеджеры для работы с такими элементами функционирования системы, как разделение ресурсов, распределенная обработка данных и обработка ошибок.

Создание класса:

Сначала создается пакет, к которому данный класс будет принадлежать (т.е. Entity, Boundary, Control). Вкладка Class Repository (левая панель), контекстное меню, пункт Model Element – Package. В открывшемся окне задается имя пакета.

Далее контекстное меню нужного пакета – Model Element – Class. В появившемся окне вводится имя класса, на вкладке Attributes – атрибуты класса (если есть), вкладка Operations – все операции класса (вызываемые функции и процедуры). Операциями класса становятся все сообщения из диаграмм последовательности, направленные к объекту, являющегося представителем создаваемого класса. Атрибутами класса становятся все значения, которые передаются объекту класса и которые он хранит.

Абстрактный класс – это класс, который не наполняется конкретным содержанием. Обычно их применяют при работе с наследованием, когда надо описать данные и поведение, общее для нескольких классов (в закладке General поставить галочку у поля Abstract. Имя класса в пакете станет написано курсивом).

Соотнесение объекта с нужным классом: в диаграмме последовательности выбрать объект, в контекстном меню Select class – Select class и выбрать нужный (если он уже был создан ранее),

либо создать новый, но он создастся вне пакетов, потом его можно перетащить в нужный пакет.

Соотнесение операции с операцией класса: двойной щелчок на названии операции –и выбрать из списка нужную (если они были созданы в классе), если нужной нет, то сначала создать операцию в классе, потом соотнести с этой.

Задание: Создать классы по всем объектам, создать пакеты (entity, boundary, control), распределить классы по пакетам, прописать все операции и атрибуты классов, распределить все объекты по классам.

Лабораторная работа №5. Отношения классов. Построение диаграммы классов. Диаграмма состояний.

Диаграммы классов.

На диаграммах классов отображаются некоторые классы и пакеты системы. Это статические картины фрагментов системы и связей между ними. Обычно для описания системы создают несколько диаграмм классов. На одних показывают некоторое подмножество классов и отношения между классами подмножества. На других отображают то же подмножество, но вместе с атрибутами и операциями классов. Третьи соответствуют только пакетам классов и отношениям между ними.

По умолчанию существует одна диаграмма классов, называемая главной. На этой диаграмме показывают пакеты классов модели. Внутри каждого пакета также должна быть главная диаграмма, включающая в себя все классы этого пакета. Сущ-ет два подхода к группировке классов: или по стереотипу, или по назначению. Иногда совмещают – разбивают по назначению, а внутри сортируют по стереотипу.

Сущ-ет 5 типов отношений, которые могут быть установлены между классами: ассоциации, зависимости, агрегации, обобщения и реализации.

Ассоциация – это семантическая связь между классами. Ее рисуют обыкновенной линией. После того, как классы связали ассоциацией, они могут передавать друг другу сообщения на диаграмме последовательности. Могут быть однонаправленными или двунаправленными. Например, если между классом Рейс и Пассажир установлено отношение ассоциации, класс Рейс получит атрибут Пассажир, чтобы знать кто является пассажиром, а класс Пассажир – атрибут Рейс, что позволит пассажиру определить, каким рейсом он полетит. Если ассоциация однонаправлена, сообщения могут отправляться только одним из классов и соответственно приниматься другим классом и не наоборот.

Зависимость – Показывает, что один класс ссылается на другой. Т. о., изменения во втором классе повлияют на первый. Она всегда однонаправлена. Показывает, что когда один класс не участвует в создании экземпляров другого класса, ему необходимо посылать сообщения этому второму классу. При этом не генерируется никаких специальных атрибутов. Зависимости применяют и в том случае, когда класс необходим в качестве параметра или возвращаемого типа в операциях другого класса. Изображаются в виде пунктирной стрелки.

Агрегация – отношение между целым и частями. Изображается в виде линии с ромбиком у класса, являющегося целым. М.б. по значению (объекты создаются и разрушаются в одно и то же время – кнопка и окно) и по ссылке (объекты создаются и разрушаются в разное время – список сотрудников и сотрудник).

Реализация – служит для показа связи между классом и его интерфейсом, компонентом и его интерфейсом и т.д. Такое отношение соединяет открытый и видимый всем интерфейс (интерфейсный класс или вариант использования) с подробной реализацией этого интерфейса (класс, пакет или реализация варианта использования).

Обобщение – для отображения отношений наследования между двумя классами.

Для выявления отношений нужно:

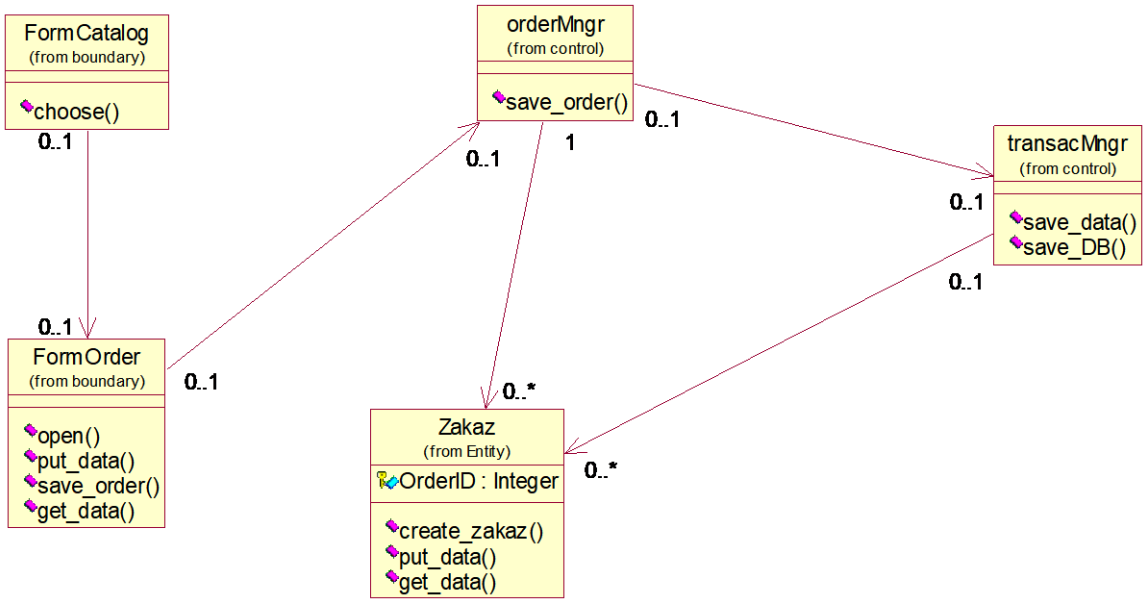
1. Изучить диаграммы последовательности. Если класс А посылает сообщение классу В, между этими классами должно быть установлено отношение. Как правило, обнаруживаемые таким образом отношения – это ассоциации или зависимости.
2. Исследовать классы на предмет наличия отношений целое – часть. Любой класс, состоящий из других классов, может принимать участие в отношениях агрегации.
3. Исследовать классы на предмет отношений обобщения. Найти все классы, у которых есть несколько типов или вариантов. Например, класс Сотрудник. В фирме могут быть два типа сотрудников – получающих почасовую оплату и оклад. Для них нужно создать классы. Общие для всех сотрудников атрибуты, операции и отношения следует поместить в класс Сотрудник, а уникальные – в соответствующие классы.
4. Изучить классы еще раз, чтобы найти остальные отношения обобщения. Чтобы обнаружить классы, которые имеют много общего (например, счет до востребования и сберегательный счет – их данные и поведение сходно, для общих элементов двух классов

можно создать третий класс – Счет).

Задание множественности

Множественность показывает, сколько экземпляров одного класса взаимодействуют с помощью отношения с одним экземпляром другого класса в данный момент. Отношение множественности позволяет понять, является ли данное отношение обязательным (студент и курс).

Отношения можно уточнять с помощью имен или ролевых имен. Имя отношения – это обычно глагол или глагольная фраза, описывающая, зачем нужна связь (компания нанимает сотрудников).



Для создания диаграммы классов в Visual Paradigm удобнее всего вызвать контекстное меню пакета во вкладке Class Repository и в разделе Sub Diagrams выбрать New Diagram (рис. 10). В появившемся окне пишем Class и выбираем Class Diagram, жмем Next. В следующем окне выбираем Blank и жмем Next. В последнем окне можно поменять название диаграммы и нажать Ok.

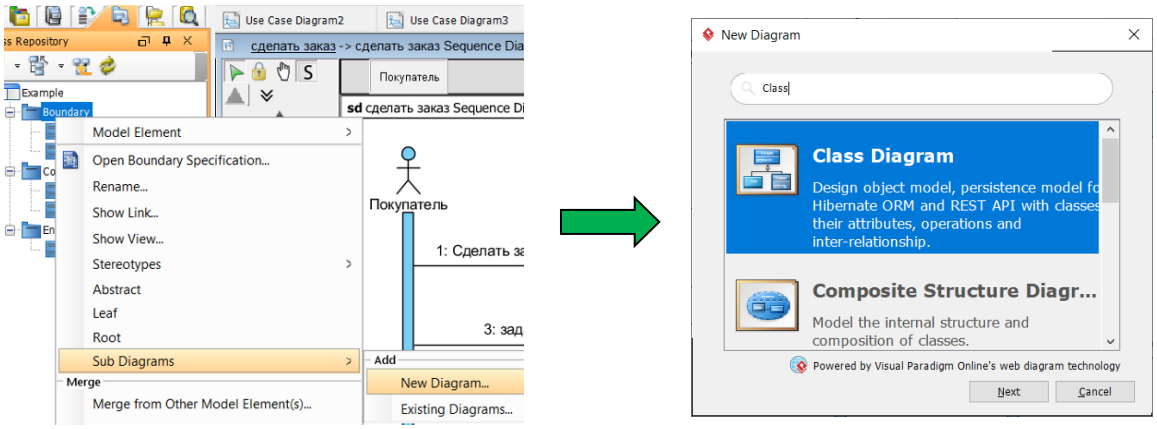


Рис. 10. Добавление объектов на диаграмму

На появившееся поле можно перетащить нужные классы и соединить соответствующими отношениями. Множественность задается через контекстное меню возле нужного класса у линии, соединяющей классы, выбираем пункт Multiplicity и там ставим нужный параметр (рис. 11).

Классы, которые соединены отношениями на одной диаграмме, будут связаны и если их вытащить на другую диаграмму.

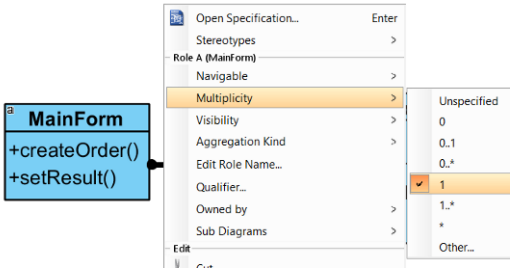
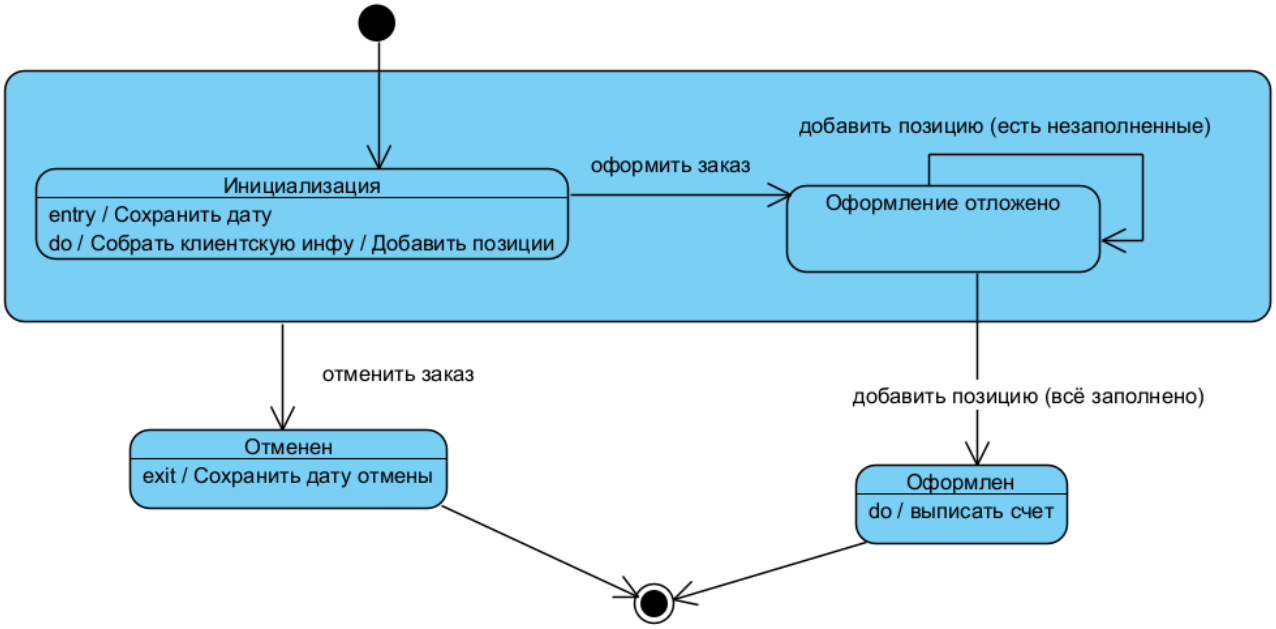


Рис. 11. Задание множественности

Для объектов пакета Entity обычно создаются диаграммы состояний. Диаграмма состояний показывает все возможные состояния, в которых может находиться объект, а также процесс смены состояний в результате внешнего влияния.



Для создания диаграммы состояний (State Machine Diagram) вызываем контекстное меню объекта в Class Repository, в пункте Sub Diagrams выбираем New Diagram, в появившемся окне печатаем State, выбираем State Machine Diagram и нажимаем Next (рис. 12).

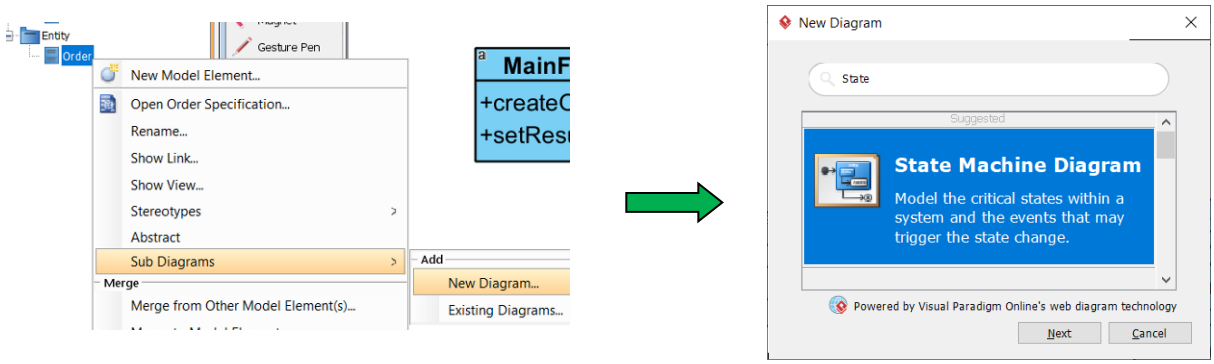


Рис. 12. Создание диаграммы состояний

В следующем окне выбираем Blank и жмем Next. В последнем окне можно поменять название диаграммы и нажать Ok. В появившемся рабочем окне уже есть точка начала диаграммы. Можно вытащить состояние из точки начала (рис. 13) из верхнего правого меню. Тянем его вниз, отпускаем и появляется выбор нужного объекта, выбираем Transition –> State.

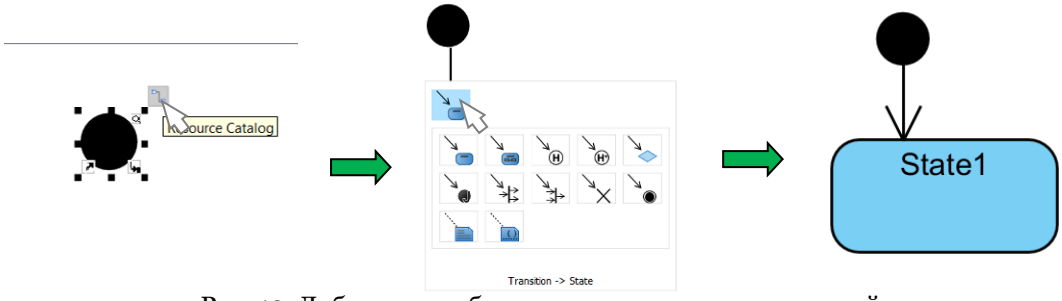


Рис. 13. Добавление объектов на диаграмму состояний

Появится объект, обозначающий состояние. Меняем его название на нужное и далее вытаскиваем другие состояния из него. Все переходы в диаграмме должны в итоге приводить к конечному состоянию (Final State).

Задание. Создать диаграммы классов в пакетах Boundary и Control, 1 общую диаграмму с отношениями между классами, родительские классы (в свойствах выбирается Abstract, стрелка от потомков к родителям, прописать общие операции и атрибуты). Для классов пакета Entity делается диаграмма состояний.

Лабораторная работа №6. Диаграмма размещения.

Диаграмма размещения (Deployment Diagram) отражает физическое распределение готового приложения, включая размещение и топологию сети, а также размещение в ней компонентов системы. Рассматриваются также проблемы определения требуемой полосы пропускания сети, предполагаемое кол-во параллельно работающих пользователей, действия при неполадках на сервере и т.п.

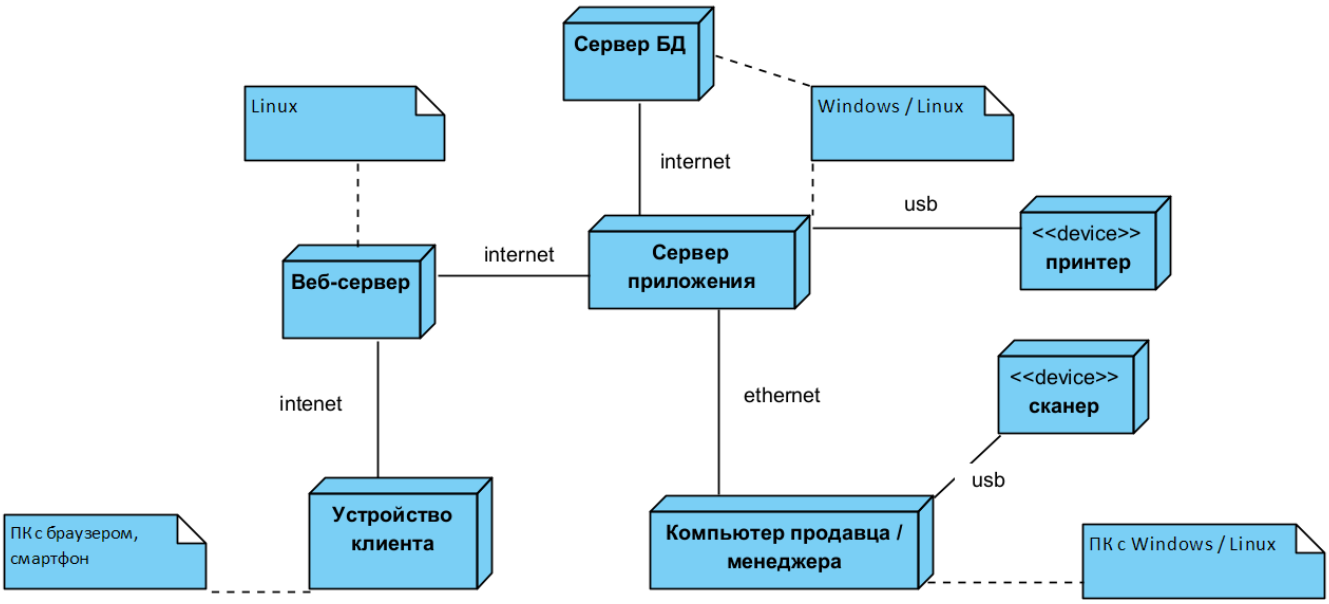
Для модели может быть создана только одна диаграмма размещения.

Вычислительное устройство (Node) – любая машина, способная производить обработку данных. К ним относятся: серверы, рабочие станции, устройства, содержащие физические процессоры.

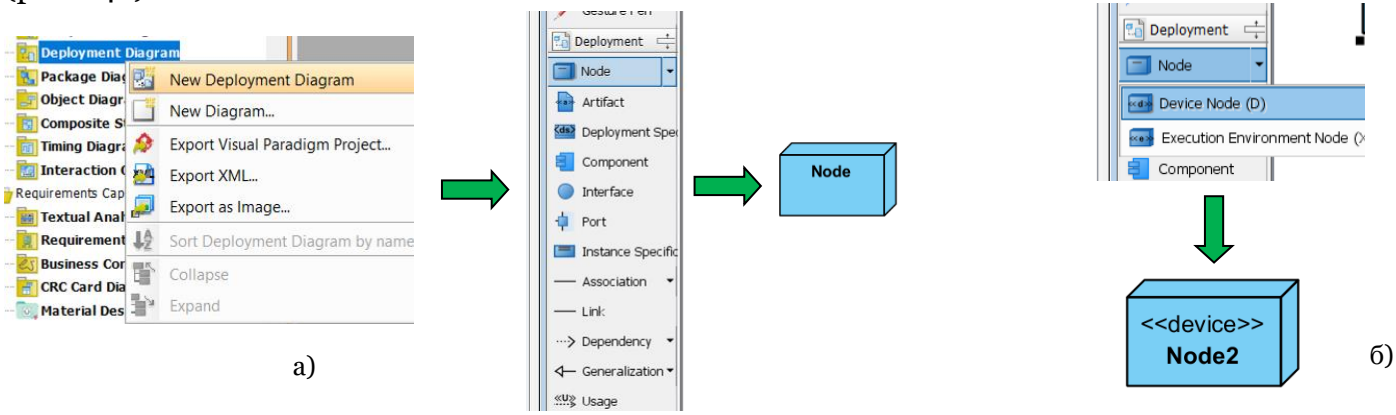
Для их классификации применяются стереотипы (UNIX, Windows, Linux).

Характеристики процессора – это его физическое описание (скорость процессора, объем памяти и др.) В каждом процессоре можно описать процессы, на нем выполняемые, например любой исполняемый файл (в спецификации, закладка Detail).

Устройством (device) называется аппаратура, применяющаяся для решения определенной задачи или ограниченного круга задач. Это – терминалы, принтеры и сканеры.



Для создания диаграммы размещения на вкладке Diagram Navigator вызываем контекстное меню раздела и выбираем New Deployment Diagram (рис. 14). Сразу появится рабочая область диаграммы с панелью инструментов. Выбираем объект Node и помещаем его на поле (рис. 14а). Меняем название на нужное и вытаскиваем из него следующие объекты для получения диаграммы, изображенной выше. Для добавления устройств (Device Node) раскрываем список с объектом Node, выбираем объект Device Node и помещаем на рабочее поле (рис. 14б).



а) б)

Рис. 14. Добавление объектов на диаграмму

Создать Диаграмму размещения для своего проекта.