Методические указания к практическим занятиям

Оглавление

Введение

Темы практических работ

Методические рекомендации по выполнению практических занятий

Практические работы выполняются после освоения соответствующего лекционного материала. Работы выполняются индивидуально как на учебном занятии, так и во время самостоятельной работы. После выполнения работы, полученные результаты оформляются в виде отчета. Отчет должен включать в себя постановку задачи, описание хода работы, полученные результаты и выводы по работе. Написанный отчет должен быть сдан преподавателю и после проверки должен быть защищен. По результатам защиты отчета преподавать выставляет оценку по данной практической работе и допускает студента к следующей работе.

Критерии оценки защиты работы и участия в обсуждении других работ:

- культура речи, ясность и четкость изложения результатов работы;
- проявление коммуникативных навыков, умений коллективной работы, толерантности, нацеленности на сотрудничество;
- готовность воспринимать конструктивную критику, пересматривать свои установки, искать более эффективные подходы.

Критерии оценки:

Всего за практические (семинарские) занятия студент может набрать 30 баллов (6 х 5 = 30)

- 5 баллов выставляется студенту, если все требования, предъявляемые к заданию, выполнены, работа подготовлена и представлена в срок, студент продемонстрировал в процессе защиты работы и участия в обсуждении других работ требуемые качества;
- 4 балла выставляется студенту, если все требования, предъявляемые к заданию, выполнены, но есть существенные замечания по ряду характеристик выполнения и/или защиты работы;
- 3 балла выставляется студенту, если большинство требований, предъявляемых к заданию, выполнены, но студент не защитил работу в срок или не продемонстрировал в процессе защиты работы и участия в обсуждении других работ большинства требуемых качеств;
- работа не зачтена (0 баллов), если разработанное задание репродуктивного уровня, студент демонстрирует недостаточные знания по теоретическим аспектам работы, требования к работе выполнены частично. Небрежно оформленные иллюстрации, грамматические ошибки в отчете.

Практическая работа №1. Коэволюция информационных технологий и информационных систем (2 часа)

По выбранной индивидуально теме из представленного ниже списка подготовить отчет о предметной области проектируемой информационной системы, ее функциональных и нефункциональных требованиях. Проанализировать необходимые для разработки и эксплуатации данной системы информационных технологий.

1. Автоматизация HR-процессов при помощи искусственного интеллекта.
2. Автоматизированная система оценки информационной модели веб-сайтов
3. Автоматизированное «распределение» по Атласу новых профессий
4. Анализатор тональности сообщений постов пользователей в соц. сетях
5. Ваш виртуальный ассистент для выбора и покупки одежды
6. Веб-приложение для ведения базы данных соискателей на работу в ИТ-компании
7. Веб-приложение для ведения базы данных соискателей на работу в компании
8. Веб-приложение для чтения книг
9. Веб-система поддержки организации развивающего досуга
10. Виртуальная примерочная для онлайн ритейла на основе Computer Vision
11. Генеративная система бот-поэт
12. Генератор дизайна интерьера
13. Городской интернет-сервис туристических услуг
14. Двумерное игровое приложение для Android-устройств
15. Детектор конфликтогенов в личной коммуникации
16. Дизайнер UX для создания новых элементов сайта, дизайна соцсетей и фирменного стиля
17. Игровое приложение MindPlay с нейроинтерфейсом
18. Интеллектуальная система «Личный ассистент»
19. Интеллектуальная система распознавания пользовательской активности по данным сенсоров мобильных устройств.
20. Интеллектуальный ассистент (бот) риэлтерской компании
21. Интерактивные тренировки критического мышления
22. Интернет-магазин мебели с встроенным конструктором и кастомизатором
23. Интернет-магазин одежды с виртуальной примеркой
24. Интернет-магазин спортивной одежды
25. Интернет-магазин эко-продуктов питания
26. Информационная система автоматизированного анализа новостных интернет-ресурсов "Политинформатор"
27. Информационная система комплексного регрессионного анализа и класстеризации психодиагностических данных
28. Информационная система оценки технической эстетики

пользовательского интерфейса интернет-ресурсов
29. Информационная система поддержки изучения иностранного языка
30. Информационная система поддержки прикладных исследований визуальной привлекательности пользовательского интерфейса программ
31. Информационная система предприятия сервисного обслуживания и ремонта оргтехники
32. Информационная система управления ветеринарной лабораторией
33. Карьерный бот-наставник «Хай, про!»
34. Киберфизический спортивный тренажер
35. Конструктор цифровых двойников известных людей
36. Маркетплейс «FotoMagazin»
37. Маркетплейс по поиску и бронированию экскурсий туров и впечатлений.
38. Мобильное приложение виртуальных туров по Ростовской области
39. Мобильное приложение для керамистов
40. Мобильное приложение по подбору персонализированного крема для лица
41. Мобильное приложение по созданию меню для детей с аллергией
42. Мобильное приложение по составлению своего гардероба
43. Мобильное приложение, анализирующее состав продукта
44. Мобильное приложение-помощник в подготовке к экзамену в ГИБДД
45. Модульный программный комплекс психологического онлайн сопровождения клиентов
46. Нейроинтерфейс для ввода графических данных
47. Обучающее приложение NeuroEnglish с нейроинтерфейсом
48. Онлайн помощник для ускорения подбора персонала
49. Платформа для спортивных онлайн соревнований
50. Помощник абитуриента вуза
51. Приложение «Экопропаганда раздельного сбора отходов»
52. Приложение виртуальной реальности для развития профессиональных и междисциплинарных навыков
53. Приложение для обучения навыкам скорочтения
54. Приложение для проведения окулографического эксперимента
55. Приложение для учёта рабочего времени «Time Tracker»
56. Приложение по генерации интерактивных презентаций
57. Приложение-мотиватор по повышению качества жизни
58. Приложение-читалка для комиксов
59. Программа цифровой двойник Стива Джобса
60. Программа цифровой двойник Стивена Хокинга
61. Промо-сайт магистерской программы
62. Психоаналитик на основе ИИ
63. Разработка диалоговой платформы конструктора чат-ботов для мессенджеров и социальных сетей с распознаванием естественного языка (NLU)

64. Сайт Таганрогского художественного театра
65. Сайт школы успешного родителя
66. Сервис по подбору хобби
67. Сервис создания персонализированных маршрутов путешествия при помощи искусственного интеллекта.
68. Сервис-конструктор для сбора мнений жителей о городе с помощью опросов на основе карт и 3D моделей.
69. Симулятор профессий для профориентации молодежи на базе MMORPG.
70. Система HR с использованием алгоритмов машинного обучения.
71. Система бесконтактного управления компьютером для людей с ограниченными возможностями здоровья.
72. Система искусственного интеллекта, поддерживающая принятие кадровых решений на основе анализа данных из резюме, социальных сетей и открытых источников.
73. Система передачи мгновенных сообщений.
74. Система поддержки многофакторного анализа пользовательского интерфейса программных продуктов.
75. Система поддержки принятия решений в области психиатрии.
76. Система слежения за взглядом Eye Tracker специального назначения.
77. Система управления тестированием ПО.
78. Создание web-сервиса по автоматическому анализу и синтезу новостных текстов».
79. Туристическая интернет платформа «Маркетплейс туриста».
80. Цифровая личность исторической персоны.

Практическая работа №2. Методологии моделирования информационных процессов (6 часов)

- Rational Unified Process
- Разработка диаграммы вариантов использования и редактирование свойств ее элементов
- Разработка диаграммы классов и редактирование их свойств
- Разработка диаграммы кооперации и редактирование свойств ее элементов
- Разработка диаграммы последовательности и редактирование свойств ее элементов
- Разработка диаграммы состояний и редактирование свойств ее элементов
- Разработка диаграммы деятельности и редактирование свойств ее элементов
- Разработка диаграмм компонентов и развертывания, редактирование свойств их элементов, генерация программного кода проекта

В ходе работы необходимо выбрать, дать общую характеристику и рабочий интерфейс несколько популярных CASE- средств.

Rational Rose

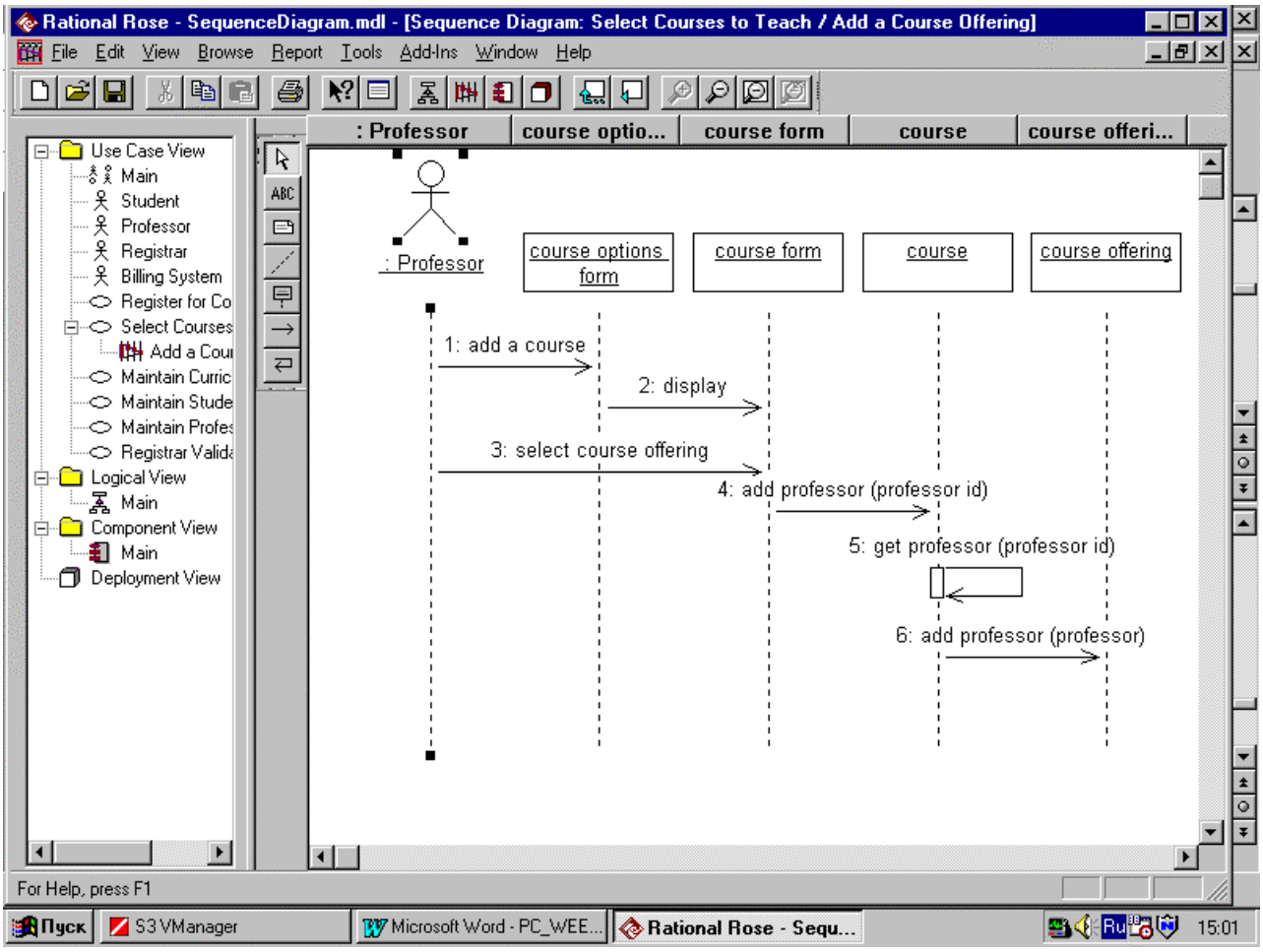

Рисунок 1. Пользовательский интерфейс Rational Rose

Rational Rose представляет собой CASE средство проектирования и

разработки информационных систем и программного обеспечения для управления предприятиями. Как и другие CASEсредства (ARIS, BPwin, ERwin) его можно применять для анализа и моделирования бизнес процессов. Первая версия этого продукта была выпущена компанией Rational Software. В дальнейшем Rational Rose былкуплен IBM.

Принципиальное отличие Rational Rose от других средств заключается в объектно-ориентированном подходе. Графические модели, создаваемые с помощью этого средства, основаны на объектно-ориентированных принципах и языке UML (Unified Modeling Language). Инструменты моделирования Rational Rose позволяют разработчикам создавать целостную архитектуру процессов предприятия, сохраняя все взаимосвязи и управляющие воздействия между различными уровнями иерархии.

Моделирование бизнес процессов в Rational Rose выполняется за счет применения различных аспектов. Каждый из этих аспектов концентрирует внимание на определенных характеристиках и возможностях процессов.

К таким аспектам относятся:

- *вариант использования* (use case). Этот аспект дает возможность понять, каким образом действуют участники процесса и за счет этого определить их взаимодействие и влияние на процесс. Для построения моделей процесса в рамках данного аспекта применяются Use-case диаграммы, диаграммы последовательностей, диаграммы совместной работы и диаграммы действий;
- *логический аспект*. С помощью этого аспекта можно определить функциональные требования процессов. Он задает логическую взаимосвязь между классами элементов процессов. Для построения моделей применяются диаграммы классов и диаграммы состояний;
- *составляющие элементы*. Этот аспект обращает внимание на состав элементов процесса и их распределение при создании информационной системы. Модели в этом аспекте строятся с помощью диаграммы компонентов. Она содержит информацию об элементах процесса и программном обеспечении;
- *ввод в действие*. Этот аспект показывает схему процесса в привязке к аппаратному обеспечению информационной системы. Для построения моделей применяется только одна диаграмма – диаграмма топологии;

За счет применения различных аспектов Rational Rose предоставляет пользователям (бизнес аналитикам, инженерам, техническим специалистам и руководителям) возможность создавать, анализировать, изменять и управлять моделями, используя единый объектно-ориентированный подход и единый язык моделирования.

Возможности Rational Rose

Последние версии Rational Rose содержат несколько программных продуктов, которые обеспечивают широкие возможности по моделированию бизнес процессов. Пользователи могут создавать графические модели процессов, приближенные к потребностям бизнеса.

Rational Rose обеспечивает следующие возможности моделированиябизнес процессов:

- *поддержка* объектного моделирования. Применение принципов объектного моделирования и языка UML позволяет приблизить модели процессов к требованиям бизнеса и упрощает вид моделей;
- *структурное представление элементов*. Модели процессов и их элементы могут быть представлены в виде графической структуры, наглядно отображающий их состав и взаимосвязи;
- *интеграция моделей*. За счет применения единого языка UML, Rational Rose позволяет объединить модели бизнес процесса, модели приложений и модели данных;
- *интеграция с программными продуктами*. Для расширения возможностей моделирования и анализа бизнес процессов в Rational Rose реализована возможность интеграции с другими программными продуктами, например, Microsoft Visual Studio;
- *открытая архитектура*. Она позволяет дополнять существующий инструментарий программы новыми функциями и возможностями;
- *обратное проектирование*. Эта возможность позволяет на основе имеющегося программного кода построить понятийную модель. Для целей моделирования бизнес процессов данная возможность может быть полезна, если моделируемый процесс автоматизирован.

Преимущества Rational Rose

Основное преимущество данного CASE средства связано с объектным принципом моделирования. За счет его применения можно максимально сблизить представления различных специалистов, которые осуществляют моделирование бизнес процессов и работают с моделями. Помимо этого, есть и преимущества, обусловленные удобством работы с программным пакетом Rational Rose.

Преимуществами Rational Rose являются:

- *поддержка командной работы*. В этом CASE средстве реализована простая поддержка всех участников проекта. Пользователи могут работать со своими собственными уникальными моделями и в своем собственном окружении без смены рабочего места, при этом сохраняется взаимосвязь с общими моделями;
- *управление моделями*. Все создаваемые модели могут быть легко изменены. Изменения в одной модели автоматически отражаются во взаимосвязанных моделях. Для управления моделями применяется система контроля версий и управления конфигурацией. Это позволяет легко проводить изменения в любых моделях бизнес процессов;
- *контроль ошибок*. Rational Rose обеспечивает отслеживание ошибок, возникающих при моделировании. Это позволяет исправить ошибки с учетом их наследования и передачи на очередной уровень моделирования;
- *документирование моделей*. Пользователи могут создавать необходимые им отчеты и документы по моделям процессов. Документы формируются под потребности пользователя и могут настраиваться для

применения к разным моделям;

- *управление конфигурацией*. Пользователи могут настраивать конфигурацию интерфейса и части приложений под свои потребности. В Rational Rose применяется графический пользовательский интерфейс (GUI), за счет которого можно настроить необходимое окружение для комфортной работы.

Аналоги:

Microsoft Visio

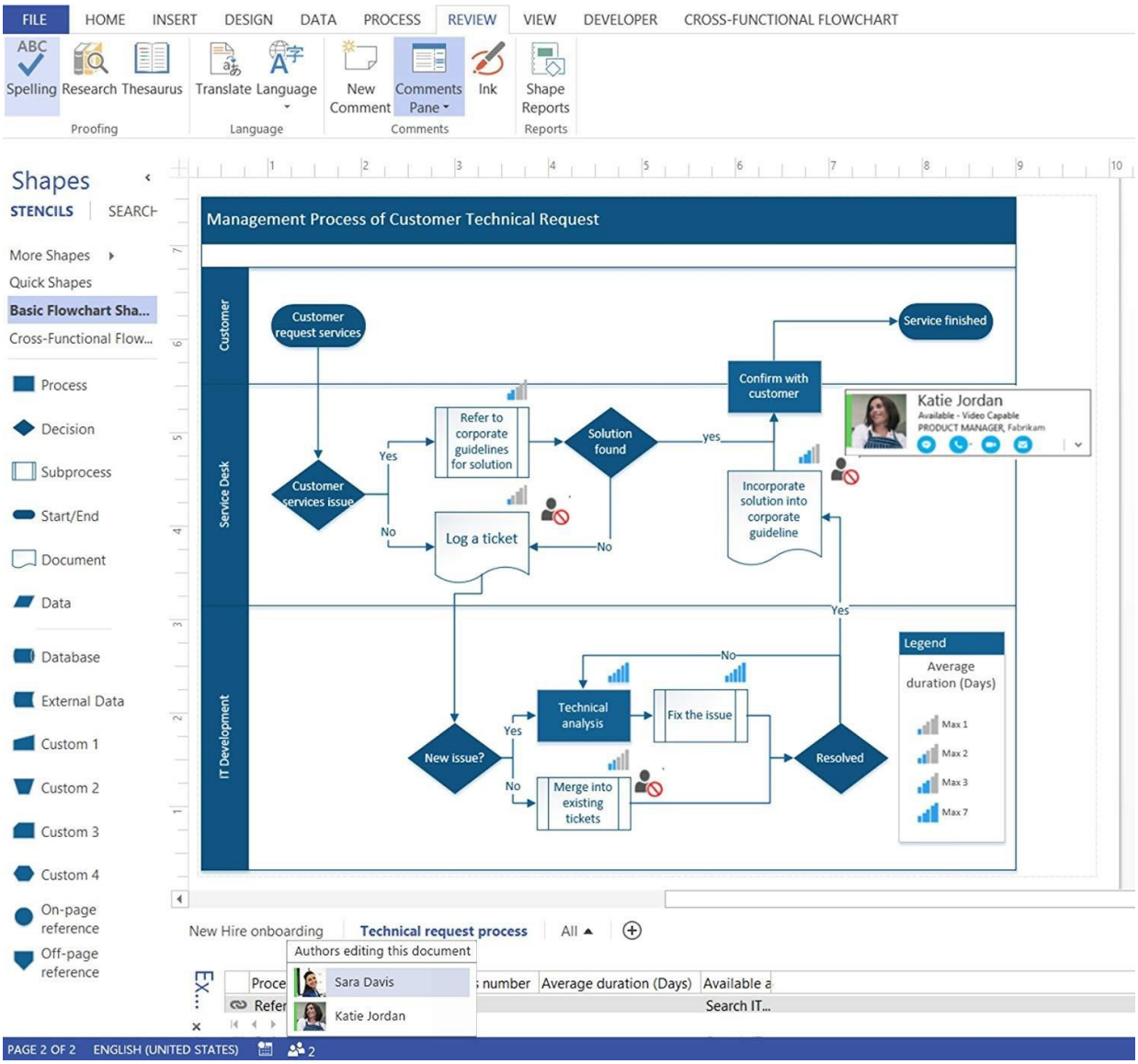

Рисунок 2. Пользовательский интерфейс MS Visio

Microsoft Visio - это программное обеспечение для рисования различных диаграмм: блок-схем, организационных диаграмм, планов зданий, поэтажных планов, диаграмм потоков данных, технологических схем, моделирования бизнес-процессов, swim lane блок-схем, 3D-карт. Visio создан в 1990-х Shapeware Corparation. Продукт быстро получил признание, и в1995, компания была переименована в Visio Corp.

Возможности Visio

Графическое оформление схем. С помощью средств Visio можно задать

различные эффекты для фигур на схемах процессов, выбрать темы оформления схем, изменять фигуры, сохраняя макеты схем и метаданные фигур.

Совместная работа над схемами. Используя web браузер можно организовать общий доступ к просмотру схем. При дополнительной установке SharePoint Server и Microsoft Lync 2013 у пользователей появляется возможность комментировать схемы, осуществлять совместную работу с ними и обмениваться сообщения.

Взаимосвязь схем с наборами данных. Каждую фигуру из схемы можно связать с набором данных из Excel, SharePoint, службы SharePoint Business Connectivity Services и SQL Server. Для наглядного представления данных можно использовать большое количество графиков и цветовых схем.

Создание схем с помощью стандартных нотаций. Для проверки корректности создаваемых схем в Visio встроены правила, позволяющие контролировать правильность применения элементов. Эти правила заданы для стандартных нотаций, таких как BPMN . При необходимости, такие правила можно задавать самостоятельно.

Преимущества Microsoft Visio 2016

Быстрое начало работы. При запуске предлагается набор встроенных схем и подсказки по контексту, что помогает пользователю быстро создать, отредактировать и закончить схему.

Продуктивная работа. Visio содержит множество фигур, которые соответствуют различным отраслевым стандартам, включая BPMN 2.0, UML

2.4 и IEEE. Данное преимущество поможет с соблюдением всех требований построить организационную диаграмму, задокументировать бизнес-процесс, нарисовать современный план этажа или быстро зарисовать блок-схему с доски.

Простая связь схем с данным. Можно связать схемы с такими источниками данных, как Excel, Active Directory, SharePoint и SQL Server. Схемы, связанные с данными, могут обновляться автоматически и отображать разные иконки, символы и цвета для отражения изменений лежащих в основе их данных.

UML Designer

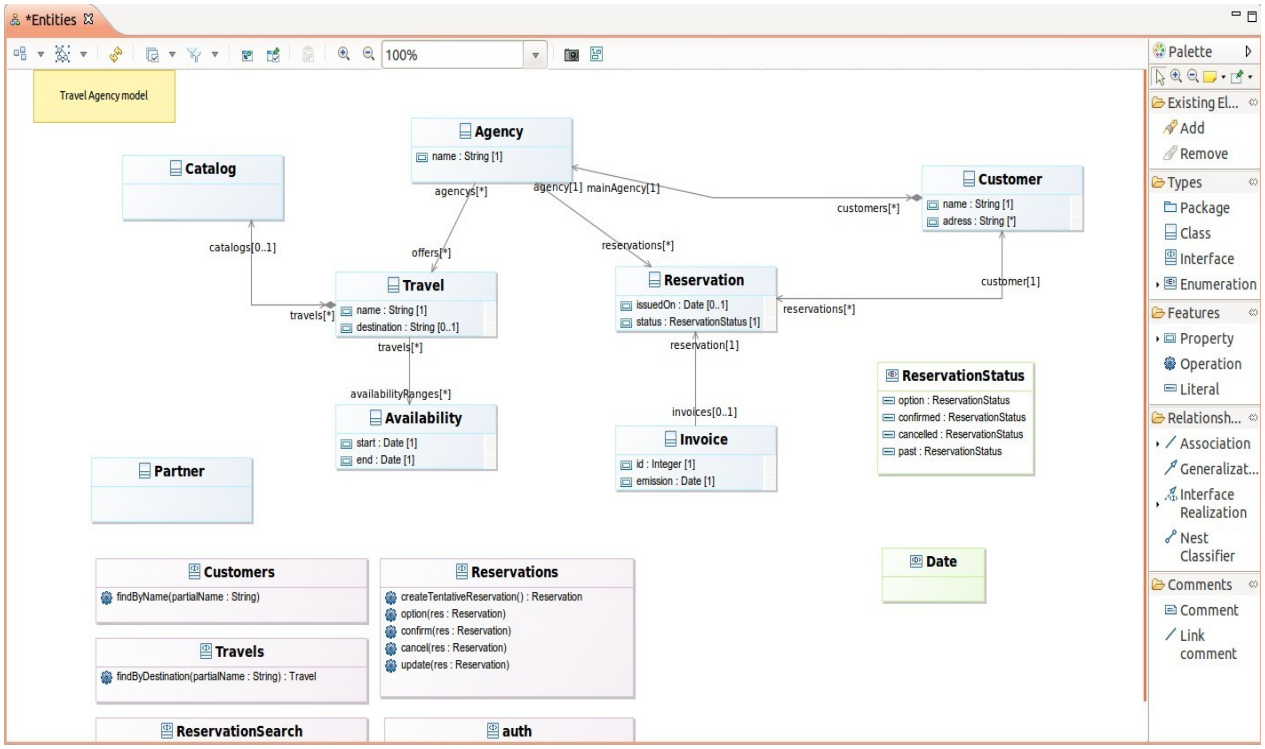

Рисунок 3. Пользовательский интерфейс UML Designer

UML Designer - это инструмент графического моделирования для UML2, основанный на подключаемом модуле Eclipse UML2 и определенный OMG . Он обеспечивает поддержку основных схем UML и профилей UML.

Поскольку он основан на Sirius, модели UML можно комбинировать с моделями для конкретных предметных областей. Каждое определение диаграммы может быть расширено и адаптировано к конкретным потребностям пользователя или объединено с предметными языками.

Draw.io

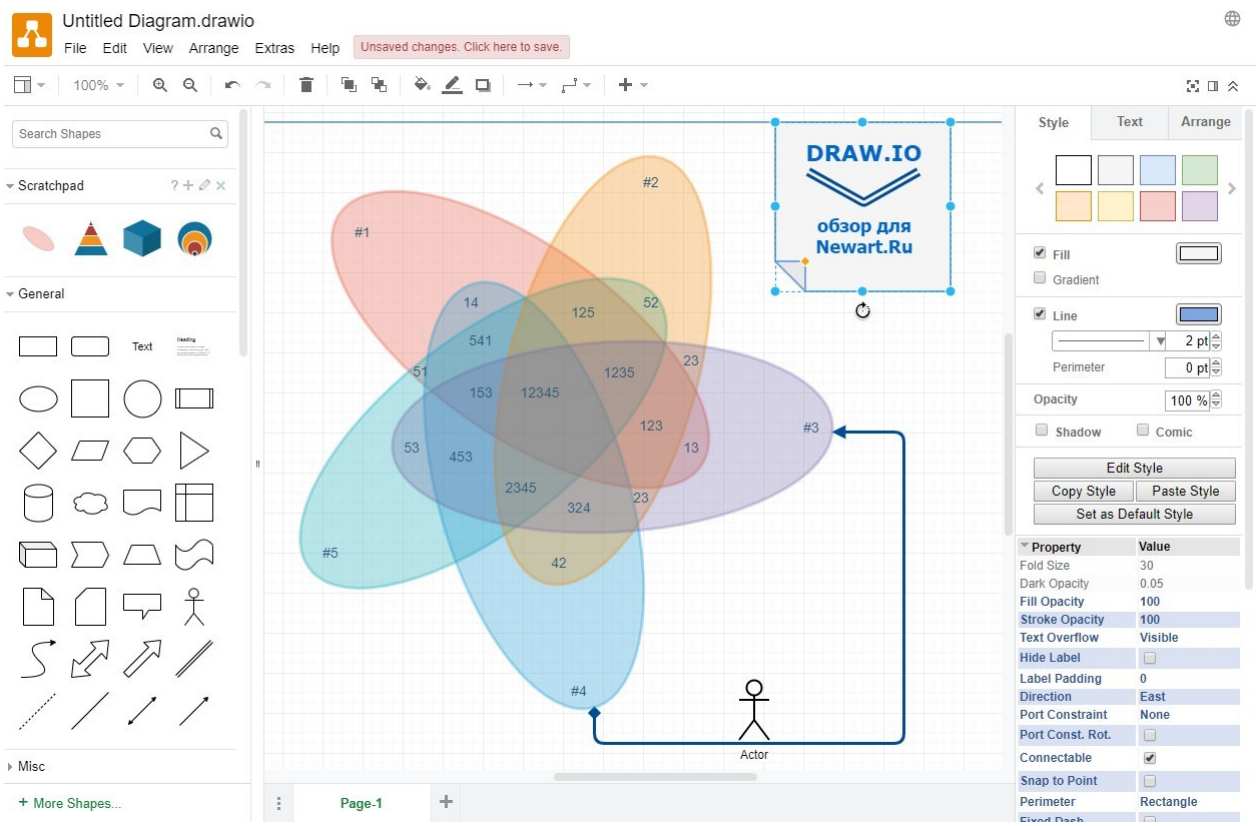

Рисунок 4. Пользовательский интерфейс Draw.io

Draw.io - это бесплатное онлайн-приложение для создания диаграмм для рабочих процессов, BPM, орг-диаграмм, UML, ER, сетевых диаграмм. Вход в систему или регистрация не требуются, а функции включают возможность локального сохранения (включая svg), набор трафаретов, импорт .vsdx, Lucidchart и Gliffy и совместное использование диаграмм в реальном времени.

Возможности

- UML-диаграммы
- ER диаграммы
- Пользовательские библиотеки
- Электрические схемы
- Онлайн и оффлайн доступ
- Древовидные диаграммы
- Плагины для Atlassian Confluence и JIRA
- Мультиформатный импорт и экспорт
- Публикация диаграмм и обмен ими
- Обработка версий (поддержка ISO 9001)
- Диаграммы процессов
- Создание организационной схемы
- Диаграммы архитектуры AWS
- Библиотека форм
- Макеты
- Автоматическая раскладка
- Отслеживание изменений и восстановление

- Каркасные модели
- Импорт / экспорт файлов VSDX
- SWOT-диаграммы
- Многостраничные диаграммы
- Сетевые диаграммы

Его *главное достоинство* – за пользование ресурсом не взнимается плата, что делает его ещё более доступным. Кроме того, для полноценной работы не нужно проходить регистрацию и авторизацию на сайте.

При входе на главную страницу предстоит выбрать путь для сохранения проекта. Конечные результаты можно хранить на удаленном хранилище – «облаках» («Google Drive», «Dropbox», «OneDrive»), на ресурсе

«GitHub», на жёстком диске устройства «Device» или непосредственно в среду для управления разработкой веб-приложений и программ «Trello».

Разработка диаграммы вариантов использования и редактирование свойств ее элементов в среде Rational Rose

Диаграмма вариантов использования - диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Основное назначение диаграммы — описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

При моделировании системы с помощью диаграммы прецедентов системный аналитик стремится:

- чётко отделить систему от её окружения;
- определить действующих лиц (акторов), их взаимодействие с системой и ожидаемую функциональность системы;
- определить в глоссарии предметной области понятия, относящиеся к детальному описанию функциональности системы (то есть прецедентов).

Для отражения модели прецедентов на диаграмме используются:

- рамки системы — прямоугольник с названием в верхней части и эллипсами (прецедентами) внутри. Часто может быть опущен без потери полезной информации,
- актор — стилизованный человечек, обозначающий набор ролей пользователя (понимается в широком смысле: человек, внешняя сущность, класс, другая система), взаимодействующего с некоторой сущностью (системой, подсистемой, классом). Акторы не могут быть связаны друг с другом (за исключением отношений обобщения/наследования),
- прецедент — эллипс с надписью, обозначающий выполняемые системой действия, приводящие к наблюдаемым акторами результатам. Надпись может быть именем или описанием того, «что» делает система.

Имя прецедента связано с непрерывным сценарием — конкретной последовательностью действий, иллюстрирующей поведение. В ходе сценария

акторы обмениваются с системой сообщениями. Сценарий может быть приведён на диаграмме прецедентов в виде UML-комментария. С одним прецедентом может быть связано несколько различных сценариев.

Часть дублирующейся информации в модели прецедентов можно устранить указанием связей между прецедентами:

- обобщение прецедента — стрелка с не закрашенным треугольником (треугольник ставится у более общего прецедента),
- включение прецедента — пунктирная стрелка со стереотипом «include»,
- расширение прецедента — пунктирная стрелка со стереотипом «extend» (стрелка входит в расширяемый прецедент, в дополнительном разделе которого может быть указана точка расширения и, возможно в виде комментария, условие расширения).

Практическая часть

При выполнении данной практической работы может использоваться среда проектирования Rational Rose 7.0 или аналогичная.

Создать диаграмму вариантов использования:

Дважды щелкните на главной диаграмме вариантов использования (Main) в браузере, чтобы открыть ее.

С помощью кнопки Use Case панели инструментов поместите на диаграмму новый вариант использования.

Назовите этот вариант использования «Ввести новый заказ».

Повторите шаги 2 и 3, чтобы поместить на диаграмму остальные варианты использования: Изменить существующий заказ, Напечатать инвентарную опись, Обновить инвентарную опись, Оформить заказ, Отклонить заказ.

С помощью кнопки Actor панели инструментов поместите на диаграмму новое действующее лицо.

Назовите его «Продавец».

Повторите шаги 5 и 6, поместив на диаграмму остальных действующих лиц: Управляющий магазином, Клерк магазина, Бухгалтерская система.

Пометить абстрактные варианты использования:

Щелкните правой кнопкой мыши на варианте использования «Отклонить заказ» на диаграмме.

В открывшемся меню выберите пункт Open Specification

(Открыть спецификацию).

Пометьте контрольный переключатель Abstract (Абстрактный), чтобы сделать вариант использования абстрактным.

Добавить ассоциации:

С помощью кнопки Unidirectional Association (Однонаправленная ассоциация) панели инструментов нарисуйте ассоциацию между действующим лицом Продавец и вариантом использования «Ввести новый заказ».

Повторите этот шаг, чтобы поместить на диаграмму остальные ассоциации.

Добавить связь расширения:

С помощью кнопки Dependency панели инструментов нарисуйте связь между вариантом использования «Отклонить заказ» и вариантом использования «Оформить заказ». Стрелка должна протянуться от первого варианта использования ко второму. Связь расширения означает, что вариант использования

«Отклонить заказ» при необходимости дополняет функциональные возможности варианта использования «Оформить заказ».

Щелкните правой кнопкой мыши на новой связи между вариантами использования «Отклонить заказ» и «Оформить заказ».

В открывшемся меню выберите пункт Open Specification

(Открыть спецификацию).

В раскрывающемся списке стереотипов введите слово extend (расширение), затем нажмите ОК.

Слово «extend» появится на линии данной связи.

Добавить описания к вариантам использования:

Выделите в браузере вариант использования «Ввести новый заказ».

В окне документации введите следующее описание к этому варианту использования: «Этот вариант использования дает клиенту возможность ввести новый заказ в систему».

С помощью окна документации введите описания ко всем остальным вариантам использования.

Добавить описания к действующему лицу:

Выделите в браузере действующее лицо Продавец.

В окне документации введите для этого действующего лица следующее описание: «Продавец - это служащий, доставляющий и старающийся продать продукцию».

С помощью окна документации введите описания к оставшимся действующим лицам.

Прикрепление файла к варианту использования:

1. Для описания главного потока событий варианта использования «Ввести новый заказ» создайте файл OrderFlow.doc, содержащий следующий текст:

Продавец выбирает пункт «Создать новый заказ» из имеющегося меню.

Система выводит форму «Подробности заказа».

Продавец вводит номер заказа, заказчика и то, что заказано.

Продавец сохраняет заказ.

Система создает новый заказ и сохраняет его в базе данных.

Щелкните правой кнопкой мыши на варианте использования «Ввести новый заказ».

В открывшемся меню выберите пункт Open Specification (Открыть спецификацию)

Перейдите на вкладку файлов. Щелкните правой кнопкой мыши на белом поле и изоткрывшегося меню выберите пункт Insert File (Ввести файл).

Укажите файл OpenFlow.doc и нажмите на кнопку Open (Открыть), чтобы прикрепить файл к варианту использования.

В результате работы должны быть выполнены следующие задачи (как в учебном примере):

- Создана диаграмма вариантов использования
- Отмечены абстрактные варианты использования
- Добавлены ассоциации

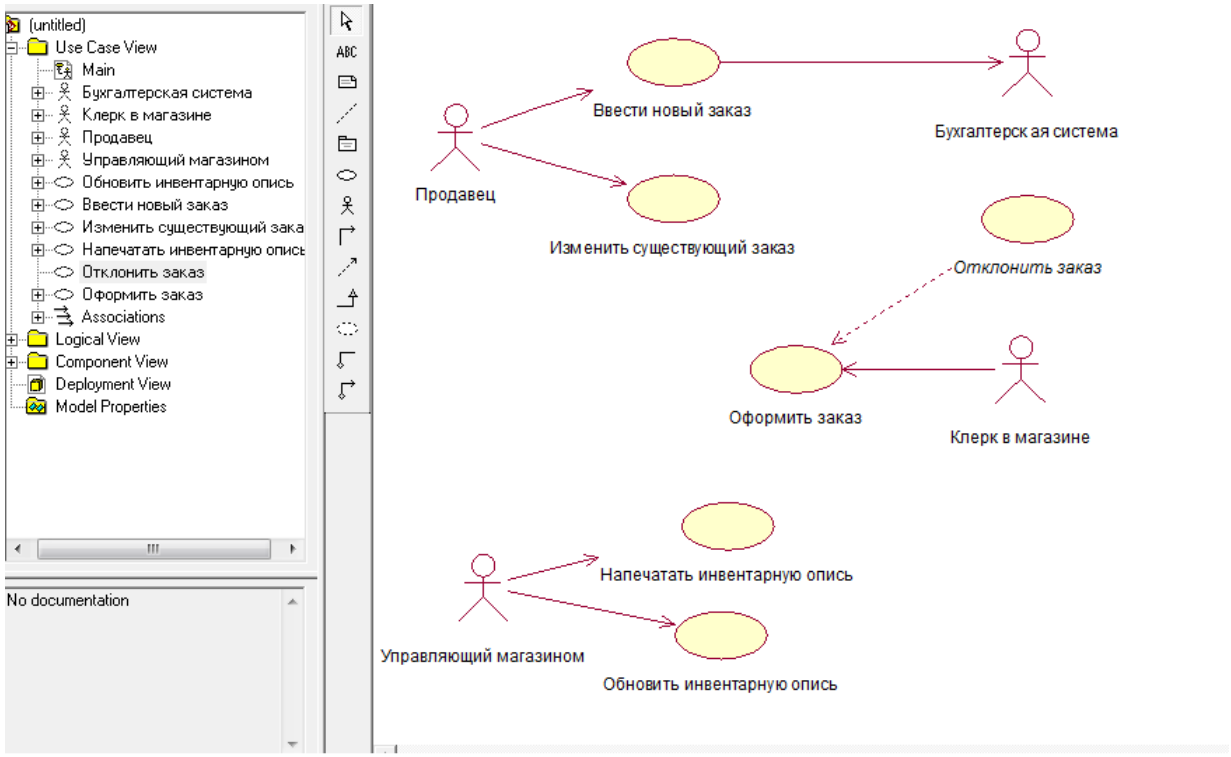

Рисунок 5 – создание диаграммы вариантов использования

Добавлена связь расширения:

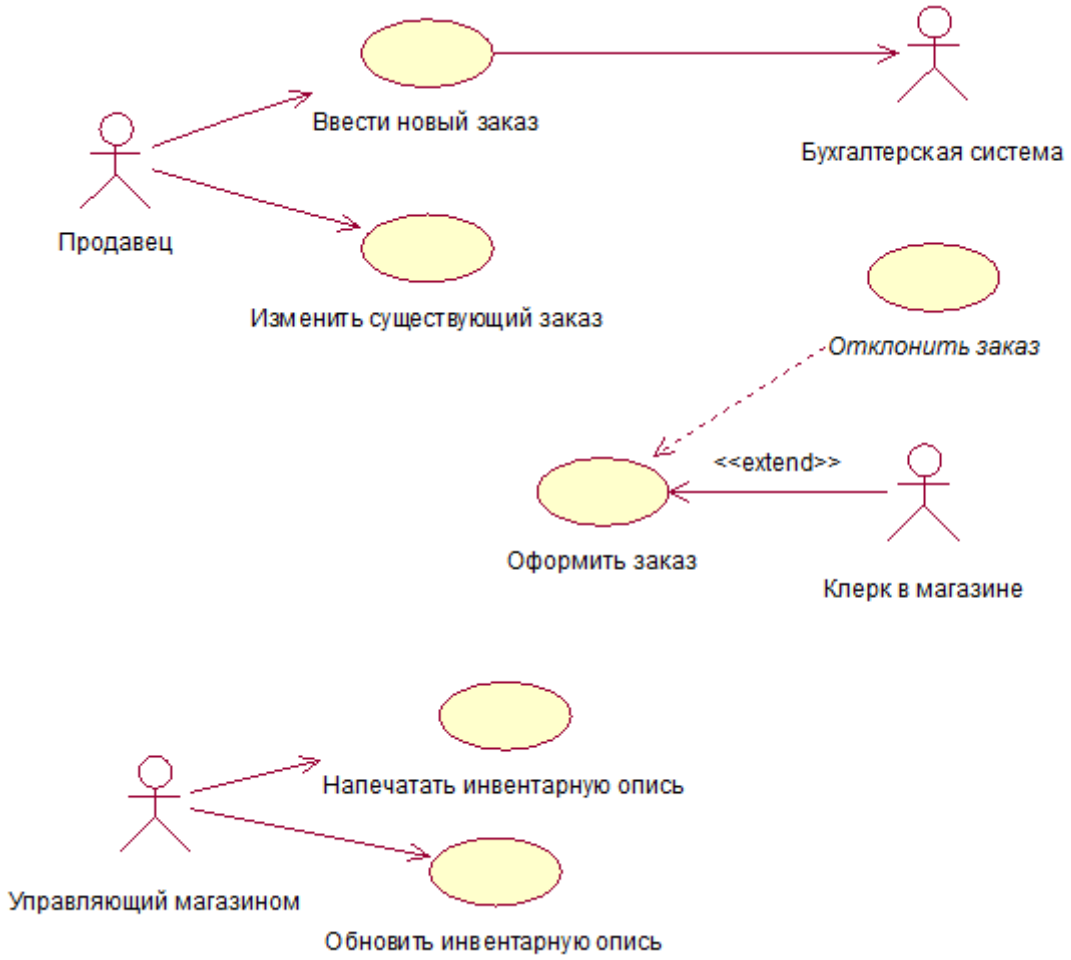

Рисунок 6 – добавлена связь расширения

Добавлены описания к вариантам использования:

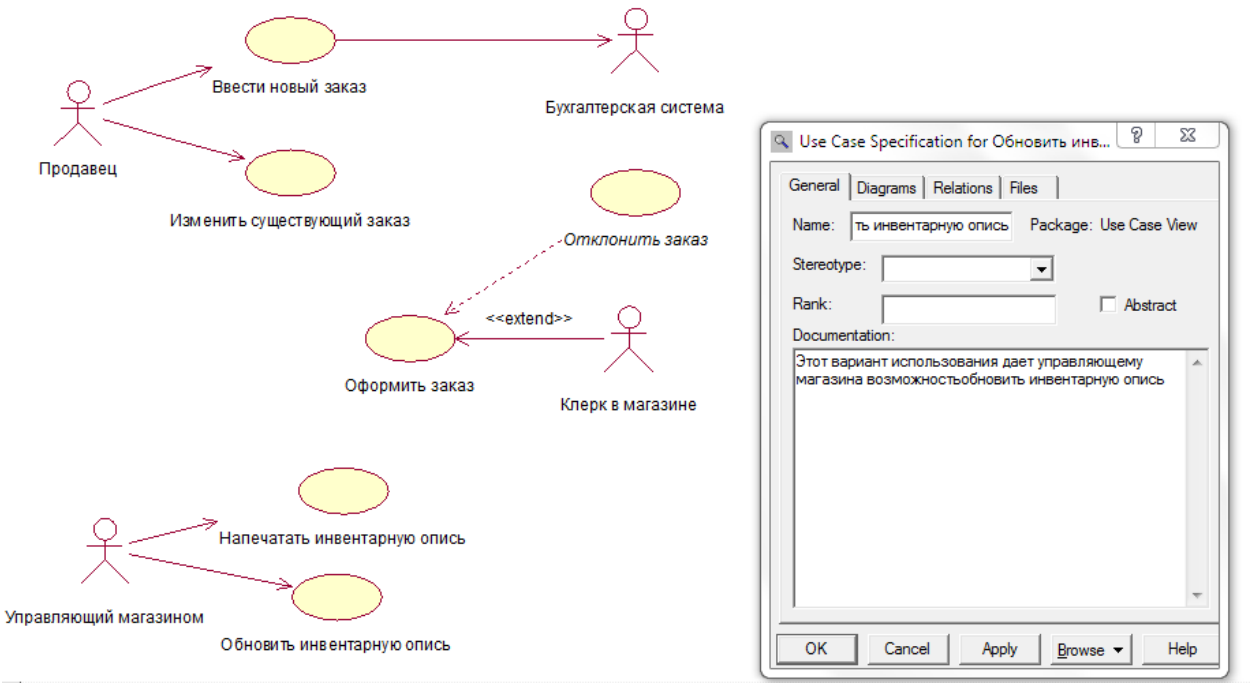

Рисунок 7 – описание к вариантам использования

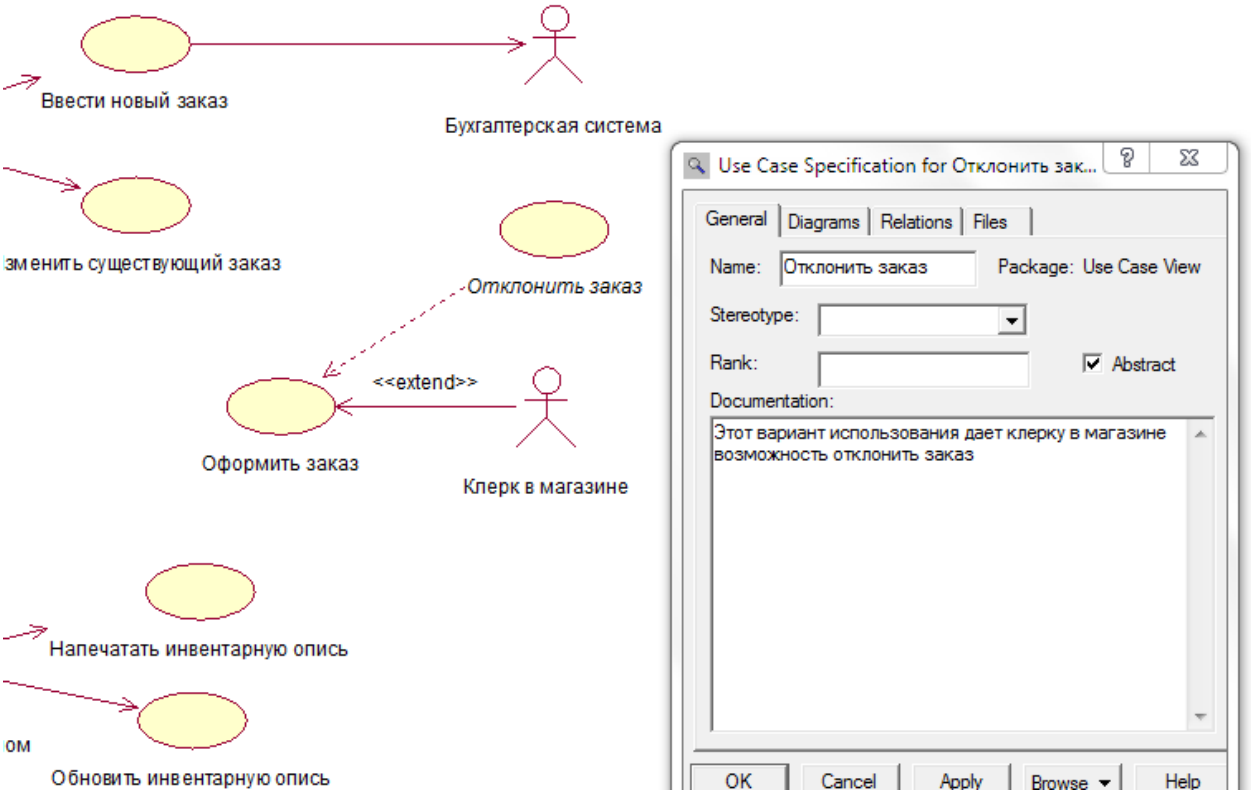

Рисунок 8 – описание к вариантам использования
Добавлены описания к действующему лицу:

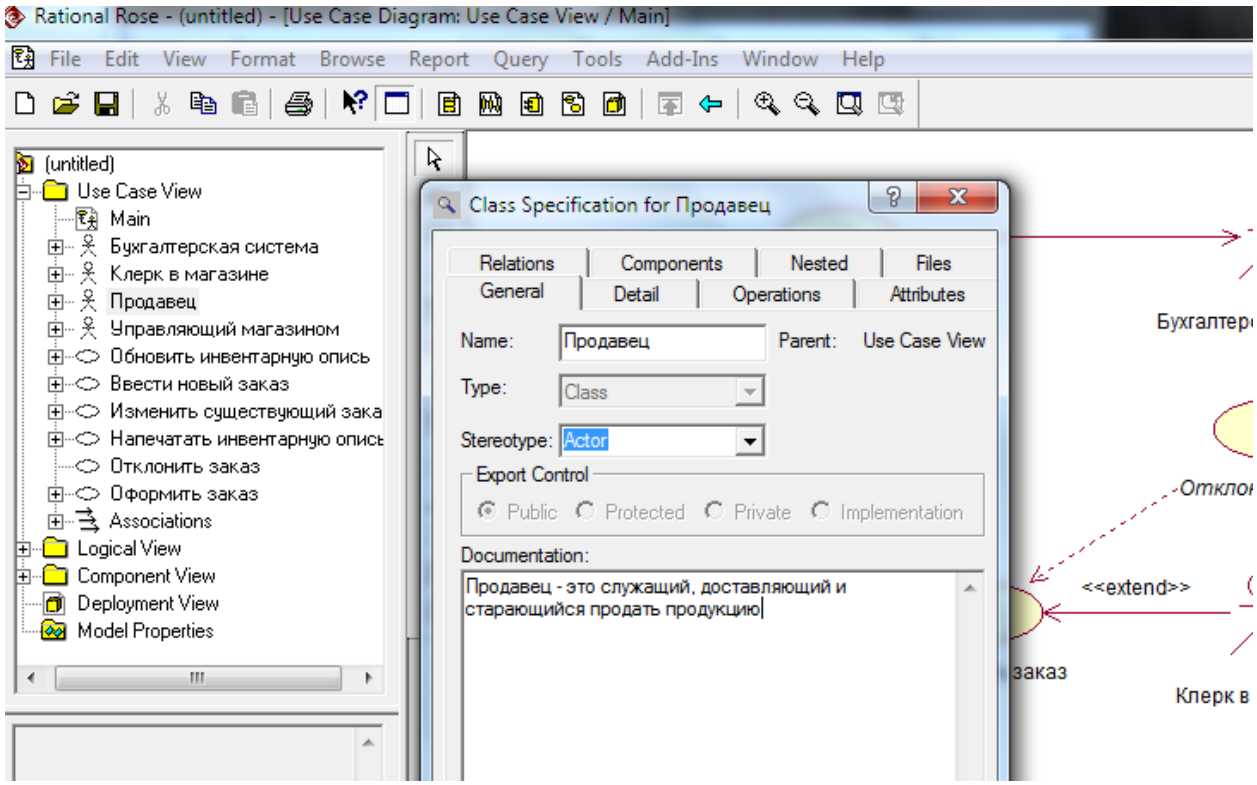

Рисунок 9 – Добавление описания

Прикрепление файла к варианту использования:

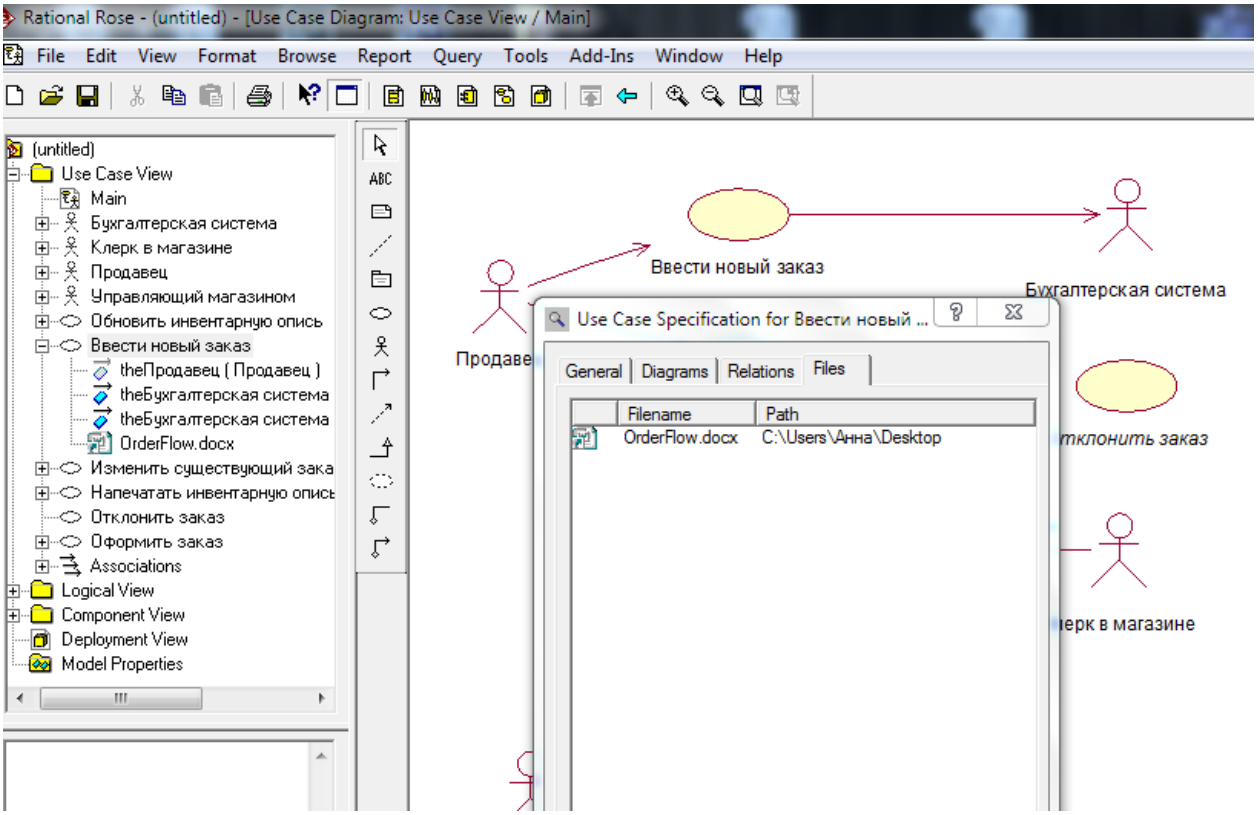

Рисунок 10 – прикрепление файла к варианту использования

В результате практической работы в среде моделирования Rational Rose 7.0 должна быть создана диаграмма вариантов использования для выбранной индивидуально информационной системы.

В результате работы были выполнены следующие задачи:

- Отмечены абстрактные варианты использования;
- Добавлены ассоциации;
- Добавлена связь расширения;
- Добавлены описания к вариантам использования;
- Добавлены описания к действующим лицам;
- Прикреплен файл к варианту использования.

Практическая работа №3. Методы математического моделирования в области проектирования и управления информационными системами (2 часа)

Исследуемые методы математического моделирования можно разделить на четыре класса:

- аналитические (априорные);
- имитационные (априорно-апостериорные) модели; эмпирико-статистические (апостериорные) модели;
- модели, в которых в той или иной форме представлены идеи искусственного интеллекта (самоорганизация, эволюция, нейросетевые конструкции и т.д.).

Аналитические модели (англ. analytical models) – один из классов математического моделирования, широко используемый в экологии. При построении таких моделей исследователь сознательно отказывается от детального описания экосистемы, оставляя лишь наиболее существенные, с его точки зрения, компоненты и связи между ними, и использует достаточно малое число правдоподобных гипотез о характере взаимодействия компонентов и структуры экосистемы. Аналитические модели служат, в основном, целям выявления, математического описания, анализа и объяснения свойств или наблюдаемых феноменов, присущих максимально широкому кругу экосистем. Так, например, широко известная модель конкуренции Лотки–Вольтерра позволяет указать условия взаимного сосуществования видов в рамках различных сообществ.

Имитационные модели (англ. simulation models) – один из основных классов математического моделирования. Целью построения имитаций является максимальное приближение модели к конкретному (чаще всего уникальному) экологическому объекту и достижение максимальной точности его описания. Имитационные модели претендуют на выполнение как объяснительных, так и прогнозных функций, хотя выполнение первых для больших и сложных имитаций проблематично (для удачных имитационных моделей можно говорить лишь о косвенном подтверждении непротиворечивости положенных в их основу гипотез).

В настоящее время можно отметить два направления развития имитационного моделирования, где предлагаются достаточно конструктивные методы компенсации априорной неопределенности, проистекающей от нестационарного и стохастического характера экологических систем. Первое направление оформилось в виде методики решения задач идентификации и верификации как последовательного процесса определения и уточнения численных значений коэффициентов модели. Второе направление связано со стратегией поиска скрытых закономерностей моделируемой системы и интеграции их в модель.

Эмпирико-статистические модели объединяют в себе практически все биометрические методы первичной обработки экспериментальной информации. Основная цель построения этих моделей состоит в следующем:

- упорядочение или агрегирование информации;
- поиск, количественная оценка и содержательная интерпретация причинно-следственных отношений между переменными системы;
- оценка достоверности и продуктивности различных гипотез о взаимном влиянии наблюдаемых явлений и воздействующих факторов;
- идентификация параметров расчетных уравнений различного назначения.

Часто эмпирико-статистические модели являются "сырьем" и обоснованием подходов к построению моделей других типов (в первую очередь, имитационных).

Важным методологическим вопросом является определение характера зависимости между факторами и результативными показателями: функциональная она или стохастическая, прямая или обратная, прямолинейная или криволинейная и т.д. Здесь используются теоретико- статистические критерии, практический опыт, а также способы сравнения параллельных и динамичных рядов, аналитических группировок исходной информации, графические методы и др.

Детерминированный анализ представляет собой методику исследования влияния факторов, связь которых с результативным показателем носит явно выраженный функциональный характер, т.е. когда результативный показатель представляется в виде произведения, частного или алгебраической суммы исходных факторов.

Стохастический анализ представляет собой обширный класс методов, опирающихся на теоретико-вероятностные представления, теоремы,критерии и методы параметрической и непараметрической статистики.

Искусственный интеллект ИИ (artificial intelligence) обычно трактуется как свойство автоматических систем брать на себя отдельные функции мыслительной способности человека, например, выбирать и принимать оптимальные решения на основе ранее полученного опыта и рационального анализа внешних воздействий. Речь идет, в первую очередь, о системах, в основу которых положены принципы обучения, самоорганизации и эволюции при минимальном участии человека, но привлечении его в качестве учителя и партнёра, гармоничного элемента человеко-машиннойсистемы.

Практическое задание – реализовать для выбранной индивидуальной темы перечисленные ниже этапы построения математической модели

Сущность построения математической модели состоит в том, что реальная система упрощается, схематизируется и описывается с помощью того или иного математического аппарата. Можно выделить следующие основные этапы построения моделей.

1. **Содержательное описание моделируемого объекта.** Объекты моделирования описываются с позиций системного подхода. Исходя из цели исследования устанавливаются совокупность элементов, взаимосвязи между элементами, возможные состояния каждого элемента, существенные характеристики состояний и отношения между ними. Например, фиксируется, что если значение одного параметра возрастает, то значение другого — убывает

и т.п. Вопросы, связанные с полнотой и единственностью выбора характеристик, не рассматриваются. Естественно, в таком словесном описании возможны логические противоречия, неопределенности. Это исходная естественно-научная концепция исследуемого объекта. Такое предварительное, приближенное представление системы называют концептуальной моделью. Для того чтобы содержательное описание служило хорошей основой для последующей формализации, требуется обстоятельно изучить моделируемый объект. Нередко естественное стремление ускорить разработку модели уводит исследователя от данного этапа непосредственно к решению формальных вопросов. В результате построенная без достаточного содержательного базиса модель оказывается непригодной к использованию. На этом этапе моделирования широко применяются качественные методы описания систем, знаковые и языковые модели.

2. **Формализация операций.** Формализация сводится в общих чертах к следующему. На основе содержательного описания определяется исходное множество характеристик системы. Для выделения существенных характеристик необходим хотя бы приближенный анализ каждой из них. При проведении анализа опираются на постановку задачи и понимание природы исследуемой системы. После исключения несущественных характеристик выделяют управляемые и неуправляемые параметры и производят символизацию. Затем определяется система ограничений на значения управляемых параметров. Если ограничения не носят принципиальный характер, то ими пренебрегают.

Дальнейшие действия связаны с формированием целевой функции модели. В соответствии с известными положениями выбираются показатели исхода операции и определяется примерный вид функции полезности на исходах. Если функция полезности близка к пороговой (или монотонной), то оценка эффективности решений возможна непосредственно по показателям исхода операции. В этом случае необходимо выбрать способ свертки показателей (способ перехода от множества показателей к одному обобщенному показателю) и произвести саму свертку. По свертке показателей формируются критерий эффективности и целевая функция.

Если при качественном анализе вида функции полезности окажется, что ее нельзя считать пороговой (монотонной), прямая оценка эффективности решений через показатели исхода операции неправомочна. Необходимо определять функцию полезности и уже на ее основе вести формирование критерия эффективности и целевой функции.

В целом замена содержательного описания формальным — это итеративный процесс.

3. **Проверка адекватности модели.** Требование адекватности находится в противоречии с требованием простоты, и это нужно учитывать при проверке модели на адекватность. Исходный вариант модели предварительно проверяется по следующим основным аспектам:

- Все ли существенные параметры включены в модель?
- Нет ли в модели несущественных параметров?

– Правильно ли отражены функциональные связи между параметрами?
– Правильно ли определены ограничения на значения параметров?

Для проверки рекомендуется привлекать специалистов, которые не принимали участия в разработке модели. Они могут более объективно рассмотреть модель и заметить ее слабые стороны, чем ее разработчики. Такая предварительная проверка модели позволяет выявить грубые ошибки. После этого приступают к реализации модели и проведению исследований. Полученные результаты моделирования подвергаются анализу на соответствие известным свойствам исследуемого объекта. Для установления соответствия создаваемой модели оригиналу используются следующие пути:

– сравнение результатов моделирования с отдельными экспериментальными результатами, полученными при одинаковых условиях;
– использование других близких моделей;
– сопоставление структуры и функционирования модели с прототипом.

Главным путем проверки адекватности модели исследуемому объекту выступает практика. Однако она требует накопления статистики, котораядалеко не всегда бывает достаточной для получения надежных данных. Для многих моделей первые два приемлемы в меньшей степени. В этом случае остается один путь: заключение о подобии модели и прототипа делать на основе сопоставления их структур и реализуемых функций. Такие заключения не носят формального характера, поскольку основываются на опыте и интуиции исследователя.

По результатам проверки модели на адекватность принимается решение о возможности ее практического использования или о проведении корректировки.

4. **Корректировка модели.** При корректировке модели могут уточняться существенные параметры, ограничения на значения управляемых параметров, показатели исхода операции, связи показателей исхода операции с существенными параметрами, критерий эффективности. После внесения изменений в модель вновь выполняется оценка адекватности.

5. **Оптимизация модели.** Сущность оптимизации моделей состоит в их упрощении при заданном уровне адекватности. Основными показателями, по которым возможна оптимизация модели, выступают время и затраты средств для проведения исследований на ней. В основе оптимизации лежит возможность преобразования моделей из одной формы в другую. Преобразование может выполняться либо с использованием математических методов, либо эвристическим путем.

Практическая работа №4. Проектирование структур баз данных и баз знаний (2 часа)

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их кооперaций, атрибутов, методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Целью создания диаграммы классов является графическое представление статической структуры декларативных элементов системы.

Для удобства восприятия диаграмму классов можно также дополнить представлением пакетов, включая вложенные.

При представлении сущностей реального мира разработчику требуется отразить их текущее состояние, их поведение и их взаимные отношения. На каждом этапе осуществляется абстрагирование от маловажных деталей и концепций, которые не относятся к реальности. Классы можно рассматривать с позиции различных уровней. Как правило, их выделяют три основных: аналитический уровень, уровень проектирования и уровень реализации.

Выполнение практического задания в среде Rational Rose

Настройка:

В меню модели выберите пункт *Tools* > *Options* (Инструменты > Параметры).

Перейдите на вкладку диаграмм.

Убедитесь, что помечен контрольный переключатель *Show Stereotypes* (Показать стереотипы).

Убедитесь, что помечены контрольные переключатели *Show All Attributes* (Показать все атрибуты) и *Show All Operations* (Показать все операции).

Убедитесь, что не помечены переключатели *Suppress Attributes* (Подавить вывод атрибутов) и *Suppress Operations* (Подавить вывод операций).

Создание пакетов:

Щелкните правой кнопкой мыши на логическом представлении браузера.

В открывшемся меню выберите пункт *New* > *Package* (Создать > пакет).

Назовите новый пакет *Entities* (Сущности).

Повторите этапы с первого по третий, создав пакеты *Boundaries* (границы) и *Control* (управление).

Создание главной диаграммы классов:

Дважды щелкните на главной диаграмме классов прямо под логическим представлением браузера, чтобы открыть ее.

Перетащите пакет *Entities* из браузера на диаграмму.

Перетащите пакеты *Boundaries* и *Control* из браузера на диаграмму.

Создание диаграммы классов для сценария «Ввести новый заказ» со всеми

классами:

Щелкните правой кнопкой мыши на логическом представлении браузера.

В открывшемся меню выберите пункт *New* > *Class Diagram* (Создать > Диаграмму классов).

Назовите новую диаграмму Классов *Add New Order* (Введение нового заказа).

Щелкните в браузере на этой диаграмме дважды, чтобы открыть ее.

Перетащите из браузера все классы *(OrderOptions, OrderDetail, Order, OrderMgr* и *TransactionMgr).*

Добавление стереотипов к классам:

Щелкните правой кнопкой мыши на классе *OrderOptions* диаграммы.

В открывшемся меню выберите пункт *Open Specification* (Открыть спецификацию)

В поле стереотипа введите слово *Boundary.*

Нажмите на кнопку ОК.

Щелкните правой кнопкой мыши на классе *OrderDetail* диаграммы.

В открывшемся меню выберите пункт *Open Specification* (Открыть спецификацию).

В раскрывающемся списке в поле стереотипов теперь будет стереотип *Boundary.* Укажите его. Нажмите на кнопку ОК.

Повторите шаги 1-4, связав классы *OrderMgr* и *TransactionMgr* со стереотипом *Control,* а класс *Order* - со стереотипом *Entity.*

Объединение классов в пакеты:

Перетащите в браузере класс *OrderOptions* на пакет *Boundaries.*

Перетащите класс *OrderDetail* на пакет *Boundaries.*

Перетащите классы *OrderMgr* и *TransactionMgr* на пакет *Control.*

Перетащите класс *Order* на пакет *Entities*

Добавление диаграмм классов к каждому пакету:

Щелкните правой кнопкой на пакете *Boundaries* браузера.

В открывшемся меню выберите пункт *New* > *Class Diagram* (Создать > Диаграмму классов).

Введите имя новой диаграммы -*Main* (Главная).

Дважды щелкните мышью на этой диаграмме, чтобы открыть ее.

Перетащите на нее из браузера классы *OrderOptions* и *OrderDetail.*

Закройте диаграмму.

Щелкните правой кнопкой на пакете *Entities* браузера.

В открывшемся меню выберите пункт *New* > *Class Diagram* (Создать > Диаграмму классов).

Введите имя новой диаграммы - *Main* (Главная).

Дважды щелкните мышью на этой диаграмме, чтобы открыть ее.

Перетащите на нее из браузера класс *Order.*

Закройте диаграмму.

Щелкните правой кнопкой на пакете *Control* браузера.

В открывшемся меню выберите пункт *New* > *ClassDiagram* (Создать > Диаграмму классов).

Введите имя новой диаграммы -*Main* (Главная).

Дважды щелкните мышью на этой диаграмме, чтобыоткрыть ее.

Перетащите на нее из браузера классы *OrderMgr* и *TransactionMgr*.

Закройте диаграмму.

В результате работы должна быть создана диаграмма классов для сценария «Ввести новый заказ».

Были выполнены следующие действия:

1. Создание пакетов
2. Создание главной диаграммы классов
3. Создание диаграммы классов для сценария «Ввести новый заказ» со всеми классами
4. Добавление стереотипов к классам
5. Объединение классов в пакеты
6. Добавление диаграмм классов к каждому пакету
7. Добавление атрибутов
8. Добавление связей
9. Добавление ассоциаций.

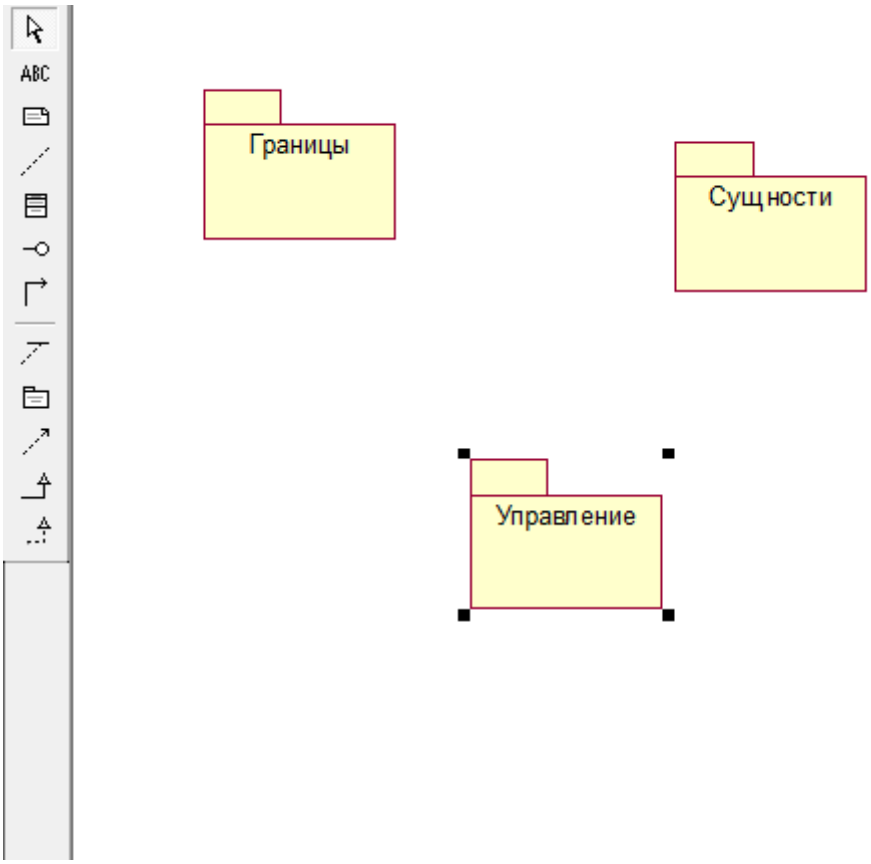

Рисунок 11 – Главная диаграмма классов системы обработки заказов

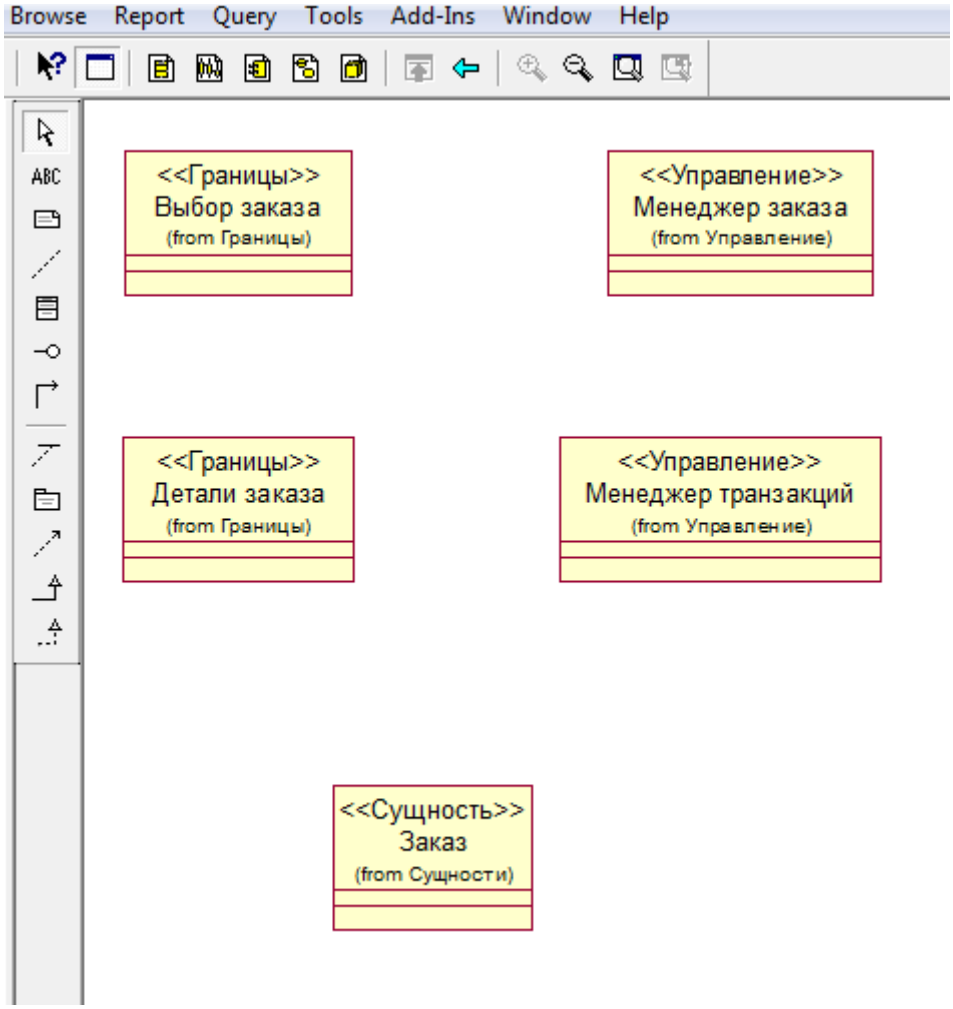

Рисунок 12 – Стереотипы классов для варианта использования «Ввести новый заказ»

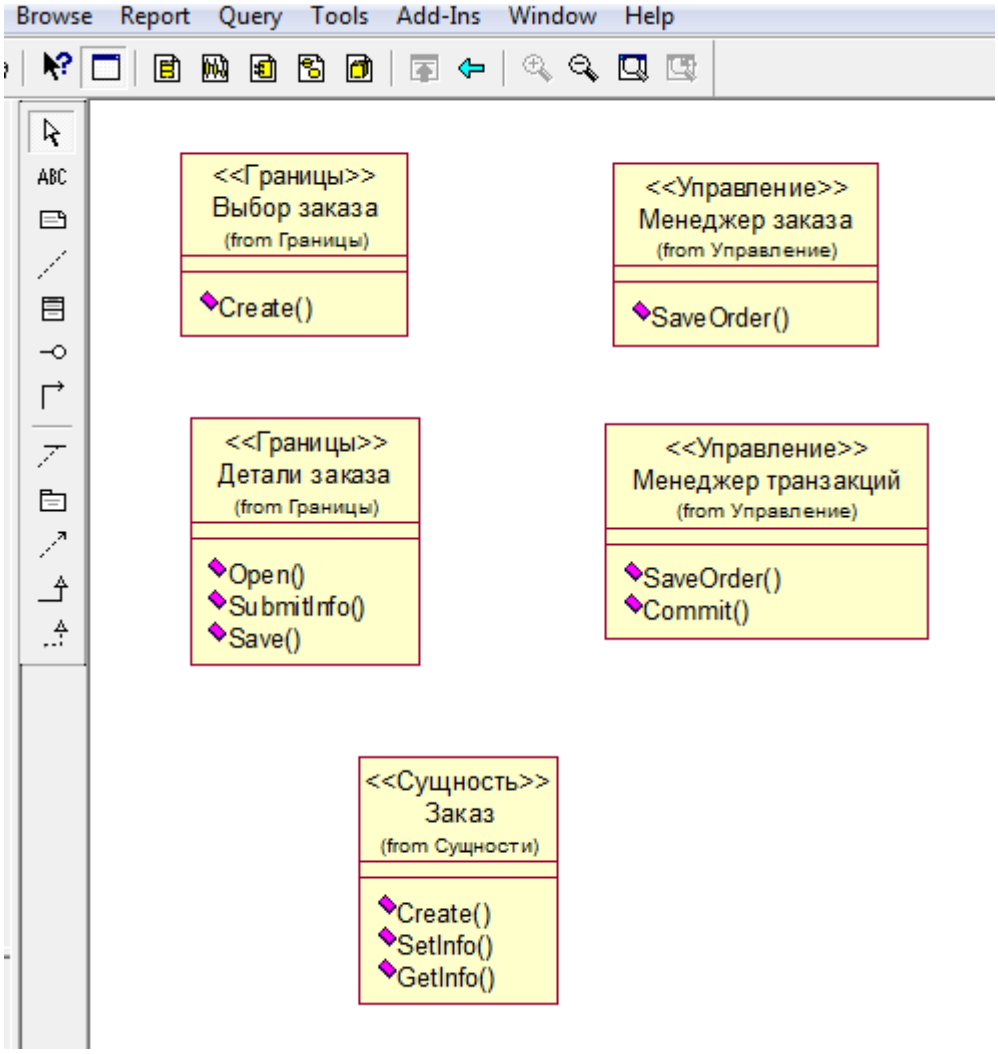

Рисунок 13 – Добавление операций

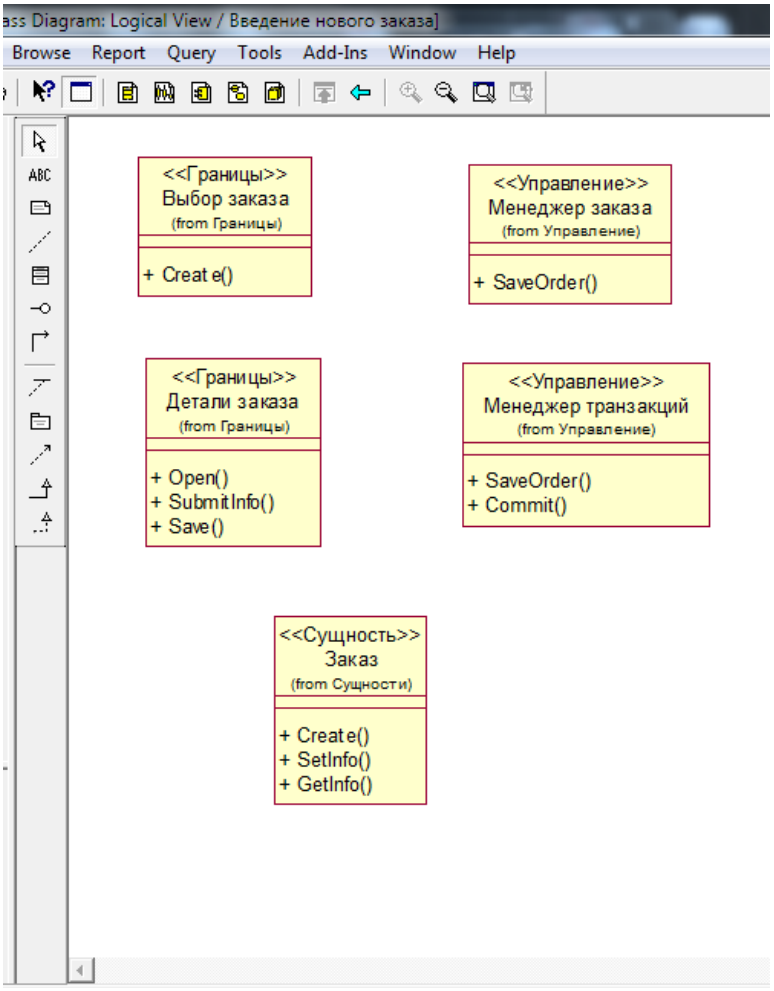

Рисунок 14 – Добавление операций

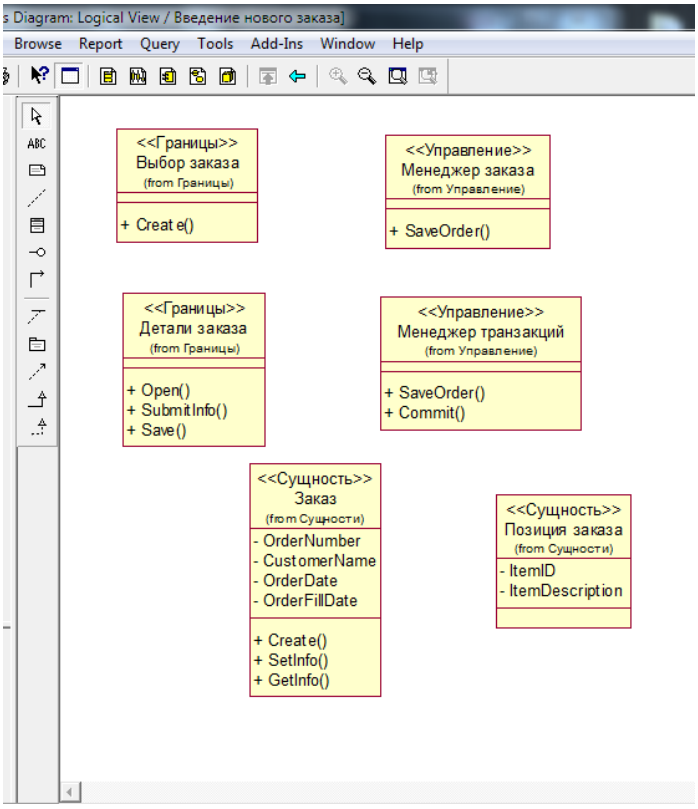

Рисунок 15 – Добавление атрибутов

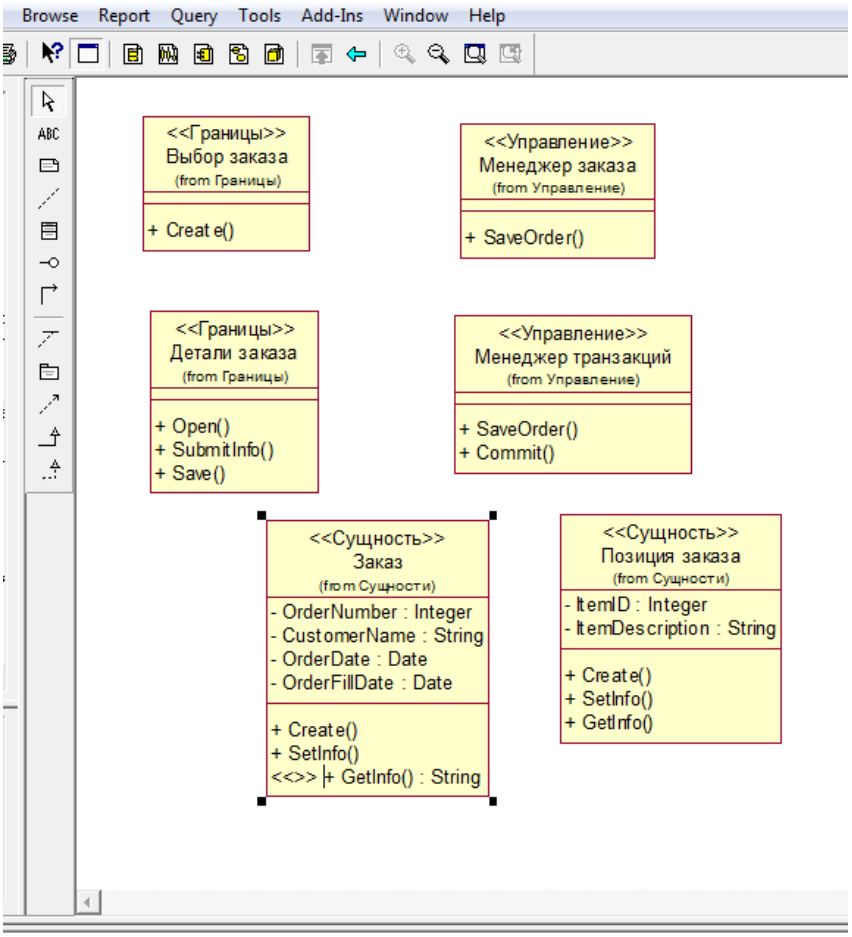

Рисунок 16 – Добавление атрибутов

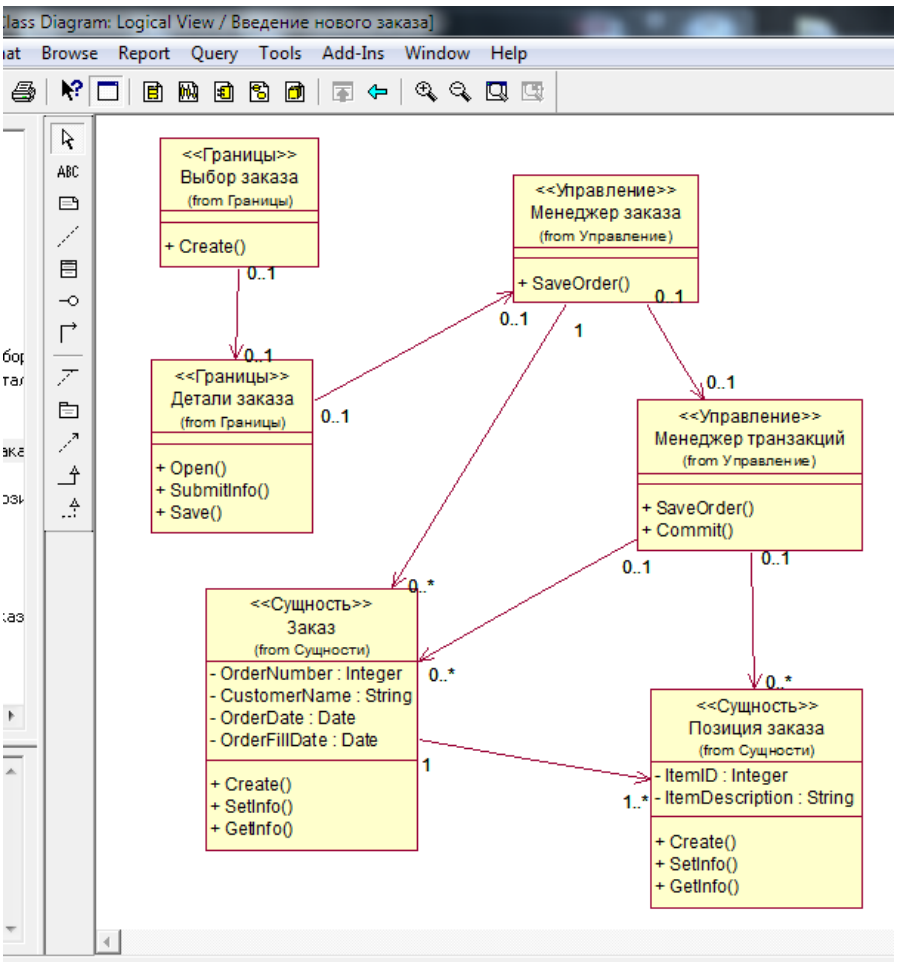

Рисунок 17 – Ассоциации варианта использования «Ввести новый заказ»

Практическая работа №5. Методологии проектирования и управления информационными системами в парадигмах программирования (2 часа)

Одна из ключевых проблем современного программирования – повторное использование модулей и компонентов (КПИ). Ими могли быть программы, подпрограммы, алгоритмы, спецификации и т. п., пригодные для использования при разработке новых более сложных ПС.

Модуль – это логически законченная часть программы, выполняющая определенную функцию, обладающая свойствами завершенности, повторного использования и др.

Модуль возник как обобщение понятия стандартных подпрограмм и процедур, которые описывались в ЯП. В их заголовке задавались внешние входные данные, а в теле модуля – операторы вычислений и вызовов подпрограмм по их имени и списку фактических параметров. Последовательность и число формальных параметров соответствовало фактическим параметрам. Вызов заготовок на одном ЯП не является проблемным, так как типы данных параметров совпадают с типами данных ЯП.

Межмодульный интерфейс – это интерфейсный модуль-посредник между двумя взаимодействующими программными объектами, выполняя функции передачи и приема данных между ними. Впервые разработан язык определения интерфейсов (ЯОИ) для описания операторов вызова модулей, параметров и их типов, а также операций проверки правильности обмена данными.

Диаграмма компонентов системы — элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

С помощью диаграммы компонентов представляются классы вместе с их интерфейсными оболочками, портами и внутренними структурами.

Компоненты связываются через зависимости, когда соединяется требуемый интерфейс одного компонента с имеющимся интерфейсом другого компонента. Таким образом, иллюстрируются отношения клиент-источник между двумя компонентами.

Зависимость показывает, что один компонент предоставляет сервис, необходимый другому компоненту. Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу.

Когда диаграмма компонентов используется, чтобы показать внутреннюю структуру компонентов, предоставляемый и требуемый интерфейсы составного компонента могут делегироваться в соответствующие интерфейсы внутренних компонентов.

Практическое задание:

Создание диаграмм компонентов системы

На данный момент уже определены все классы, требуемые для варианта использования «Ввести новый заказ». По мере реализации других вариантов использования на диаграмму следует добавлять новые компоненты.

Завершив анализ и проектирование системы, выберем в качестве языка программирования C++ и для каждого класса создадим соответствующие этому языку компоненты.

Создание пакетов компонентов:

Щелкните правой кнопкой мыши на представлении компонентов в браузере.

В открывшемся меню выберите пункт *New > Package*

(Создать > пакет).

Назовите этот пакет *Entities* (Сущности).

Повторите этапы с первого по третий, создав пакеты *Boundaries* (Границы) и *Control* (Управление).

Добавление пакетов на главную диаграмму компонентов:

Откройте главную диаграмму компонентов, дважды щелкнув на ней.

Перетащите пакеты *Entities, Boundary* и *Control* из браузера на главную диаграмму.

Рисование зависимостей между пакетами:

На панели инструментов нажмите кнопку *Dependency* (Зависимость).

Щелкните мышью на упаковке *Boundaries* Главной диаграммы компонентов.

Проведите линию зависимости до упаковки *Control.*

Повторите шаги 1-3, проведя еще линию зависимости от пакета *Control* до пакета *Entities.*

Добавление компонентов к пакетам и рисование зависимостей:

Дважды щелкните мышью на пакете *Entities* главной диаграммы компонентов, открыв главную диаграмму компонентов этого пакета.

На панели инструментов нажмите кнопку *Package Specification* (Спецификация пакета).

Поместите спецификацию пакета на диаграмму.

Введите имя спецификации пакета *Orderltem.*

Повторите шаги 2-4, добавив спецификацию пакета *Order.*

На панели инструментов нажмите кнопку *Package Body* (Тело пакета).

Поместите его на диаграмму.

Введите имя тела пакета *Orderltem.*

Повторите шаги 6-8, добавив тело пакета *Order.*

На панели инструментов нажмите кнопку *Dependency* (Зависимость).

Щелкните мышью на теле пакета *Orderltem.*

Проведите линию зависимости от него к спецификациипакета *Orderltem.*

Повторите этапы 10-12, добавив линию зависимости между телом пакета *Order* и спецификацией пакета *Order.*

Повторите этапы 10-12, добавив линию зависимости от спецификации пакета *Order* к спецификации пакета *Orderltem.*

С помощью описанного метода создайте следующие компоненты и зависимости:

Для пакета *Boundaries:*

Спецификацию пакета *OrderOptions.*

Тело пакета *OrderOptions.*

Спецификацию пакета *OrderDetail.*

Тело пакета *OrderDetail.*

Зависимости в пакете *Boundaries:*

От тела пакета *OrderOptions* до спецификации пакета *OrderOptions.*

От тела пакета *OrderDetail* до спецификации пакета *OrderDetail.*

От спецификации пакета *OrderOptions* до спецификации пакета *OrderDetail.*

Для пакета *Control:*

Спецификацию пакета *OrderMgr.*

Тело пакета *OrderMgr.*

Спецификацию пакета *TransactionMgr.*

Тело пакета *TransactionMgr.*

Зависимости в пакете *Control:*

От тела пакета *OrderMgr* до спецификации пакета *OrderMgr.*

#От тела пакета *TransactionMgr* до спецификации пакета *TransactionMgr.*

#От спецификации пакета *OrderMgr* до спецификации пакета *TransactionMgr.*

Создание диаграммы компонентов системы:

Щелкните правой кнопкой мыши на представлении компонентов в браузере.

В открывшемся меню выберите пункт *New > Component Diagram.*

Назовите новую диаграмму *System.*

Дважды щелкните на этой диаграмме.

Размещение компонентов на диаграмме компонентов системы:

Если это еще не было сделано, разверните в браузере пакет компонентов *Entities,* чтобы открыть его.

Щелкните мышью на спецификации пакета *Order* в пакете компонентов *Entities.*

Перетащите эту спецификацию на диаграмму.

Повторите этапы 2 и 3, поместив на диаграмму спецификацию пакета *Orderltem.*

С помощью этого метода поместите на диаграмму следующие компоненты:

Из пакета компонентов *Boundaries:*

#Спецификацию пакета *OrderOptions.*

#Спецификацию пакета *OrderDetail.*

Из пакета компонентов *Control:*

#Спецификацию пакета *OrderMgr.*

#Спецификацию пакета *TransactionMgr.*

На панели инструментов нажмите кнопку *Task Specification* (Спецификация задачи).

Поместите спецификацию задачи на диаграмму и назовите ее *OrderClientExe.*

Повторите этапы 6 и 7 для спецификации задачи *OrderServerExe.*

Добавление оставшихся зависимостей на диаграмму компонентов системы:

Уже существующие зависимости будут автоматически показаны на диаграмме компонентов системы после добавления

туда соответствующих компонентов. Теперь надо добавить остальные зависимости.

На панели инструментов нажмите кнопку *Dependency* (Зависимость).

Щелкните на спецификации пакета *OrderDetail.*

Проведите линию зависимости к спецификации пакета *OrderMgr.*

Повторите этапы 1-3, создав следующие зависимости:

От спецификации пакета *OrderMgr* к спецификации пакета *Order.*

#От спецификации пакета *TransactionMgr* к спецификации пакета *Orderltem.*

#От спецификации пакета *TransactionMgr* к спецификации пакета *Order.*

#От спецификации задачи *OrderClientExe* к спецификации пакета *OrderOptions.*

#От спецификации задачи *OrderServerExe* к спецификации пакета *OrderMgr.*

Соотнесение классов с компонентами:

В логическом представлении браузера найдите класс *Order* пакета *Entities.*

Перетащите этот класс на спецификацию пакета компонента *Order* в представлении компонентов браузера. В результате класс *Order* будет соотнесен со спецификацией пакета компонента *Order.*

Перетащите класс *Order* на тело пакета компонента *Order* в представлении компонентов браузера. В результате класс *Order* будет соотнесен с телом пакета компонента *Order.*

Повторите этапы 1-3, соотнеся с классами следующие компоненты:

Класс *Orderltem* со спецификацией пакета *Orderltem.*

#Класс *Orderltem* с телом пакета *Orderltem.*

#Класс *OrderOptions* со спецификацией пакета *OrderOptions.*

#Класс *OrderOptions* с телом пакета *OrderOptions.*

#Класс *OrderDetail* со спецификацией пакета *OrderDetail.*

#Класс *OrderDetail* с телом пакета *OrderDetail.*
#Класс *OrderMgr* со спецификацией пакета *OrderMgr.*
#Класс *OrderMgr* с телом пакета *OrderMgr.*
#Класс *TransactionMgr* со спецификацией пакета *TransactionMgr.*
#Класс *TransactionMgr* с телом пакета *TransactionMgr*

При выполнении практической работы может быть использована среда проектирования Rational Rose 7.0.

В результате работы должна была создана диаграмма компонентов системы и выполнены следующие шаги:

1. Создание пакетов компонентов
2. Добавление пакетов на главную диаграмму компонентов
3. Рисование зависимостей между пакетами
4. Добавление компонентов к пакетам и рисование зависимостей
5. Создание диаграммы компонентов системы
6. Размещение компонентов на диаграмме компонентов системы
7. Добавление оставшихся зависимостей на диаграмму компонентов системы
8. Соотнесение классов с компонентами.

Результаты выполнения задания по учебному примеру даны на рисунках ниже.

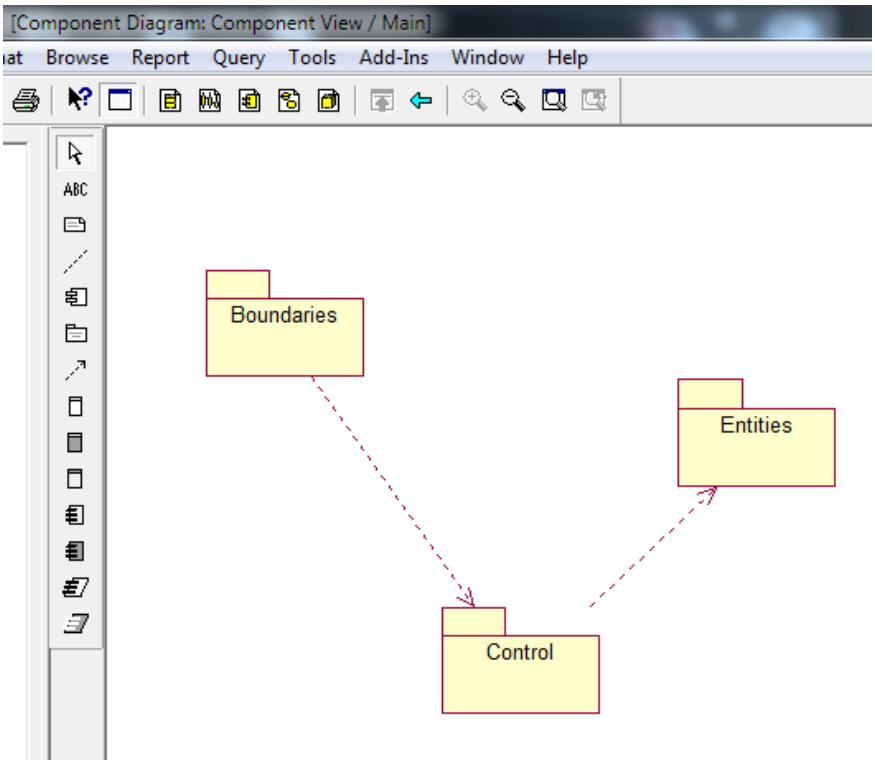

Рисунок 18 – Главная диаграмма компонентов системы

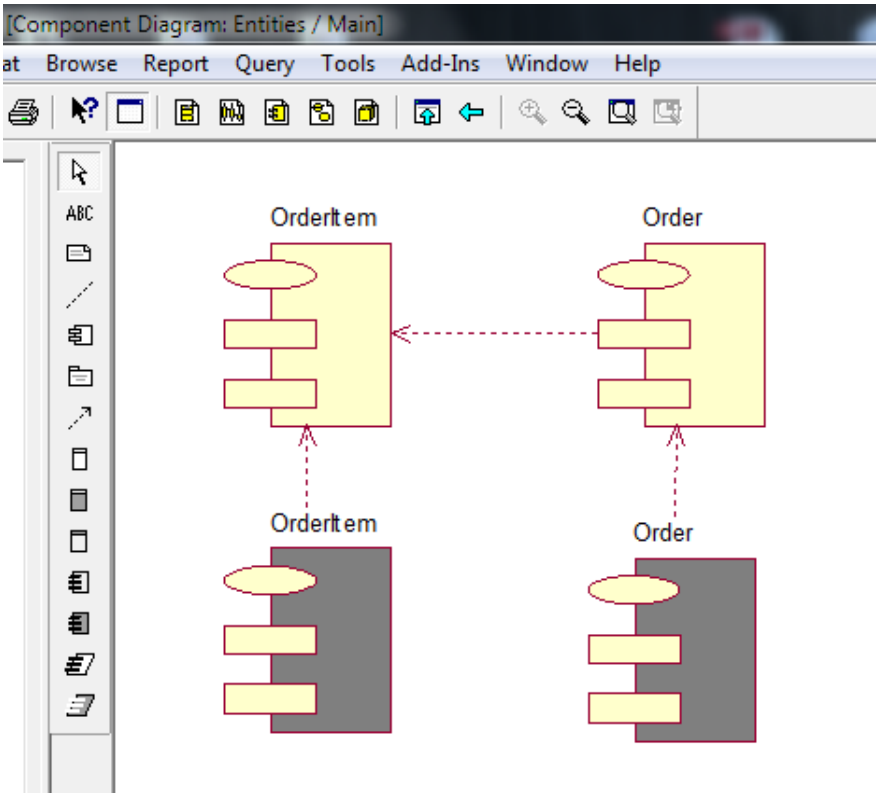

Рисунок 19 – Диаграмма компонентов пакета *Entities*

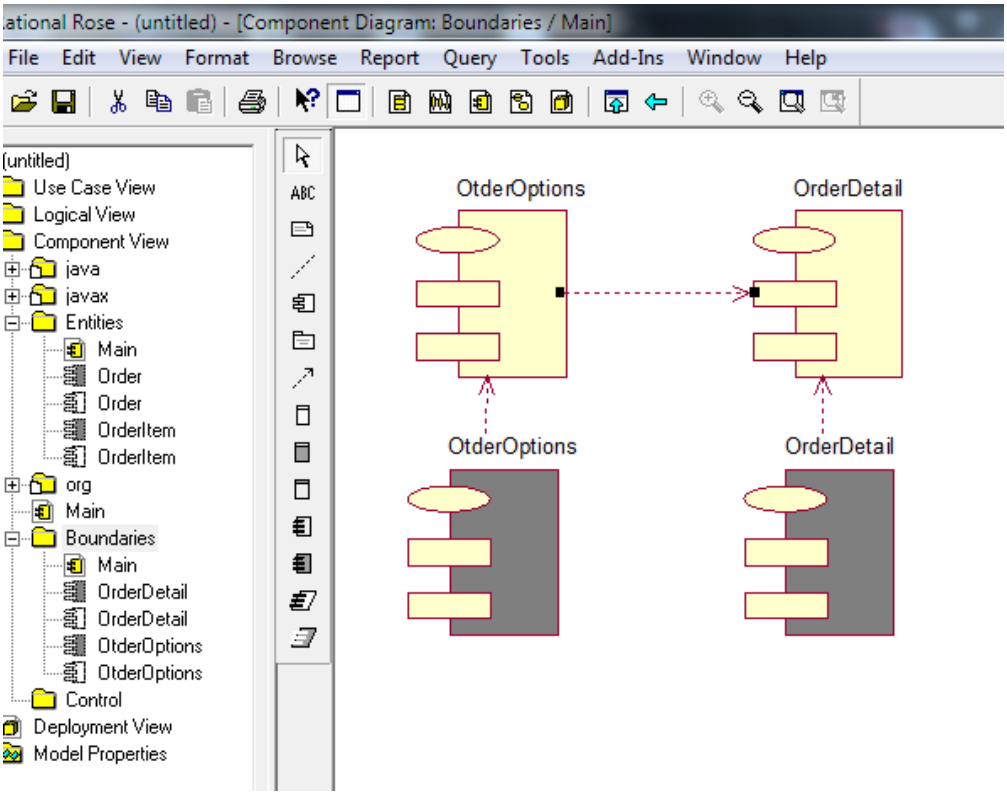

Рисунок 20 – Диаграмма компонентов пакета *Boundaries*

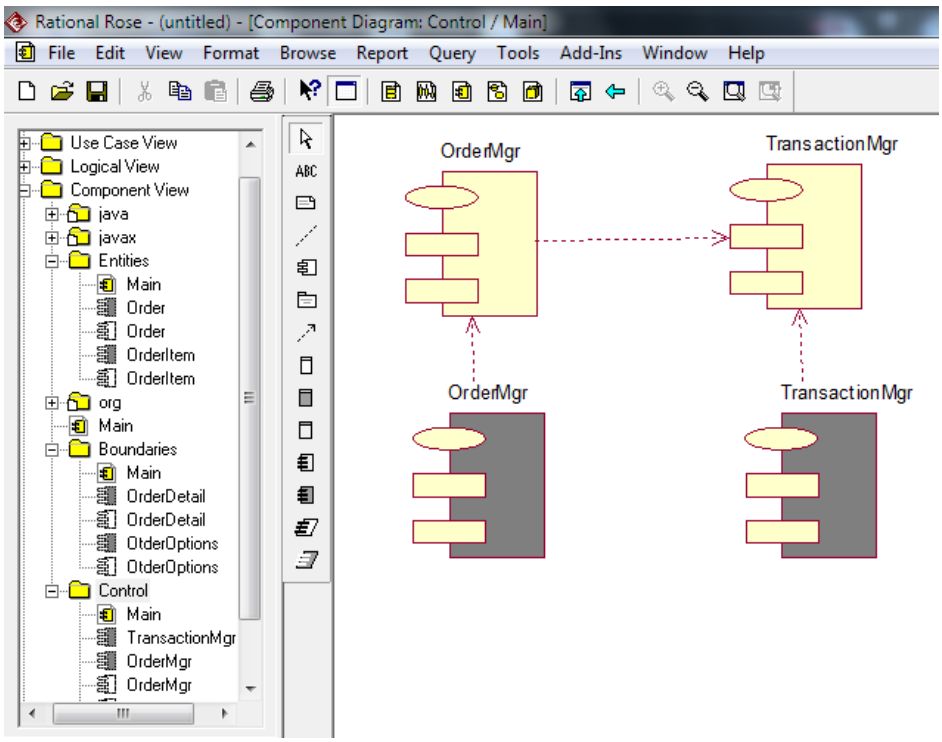

Рисунок 21 – Диаграмма компонентов пакета *Control*

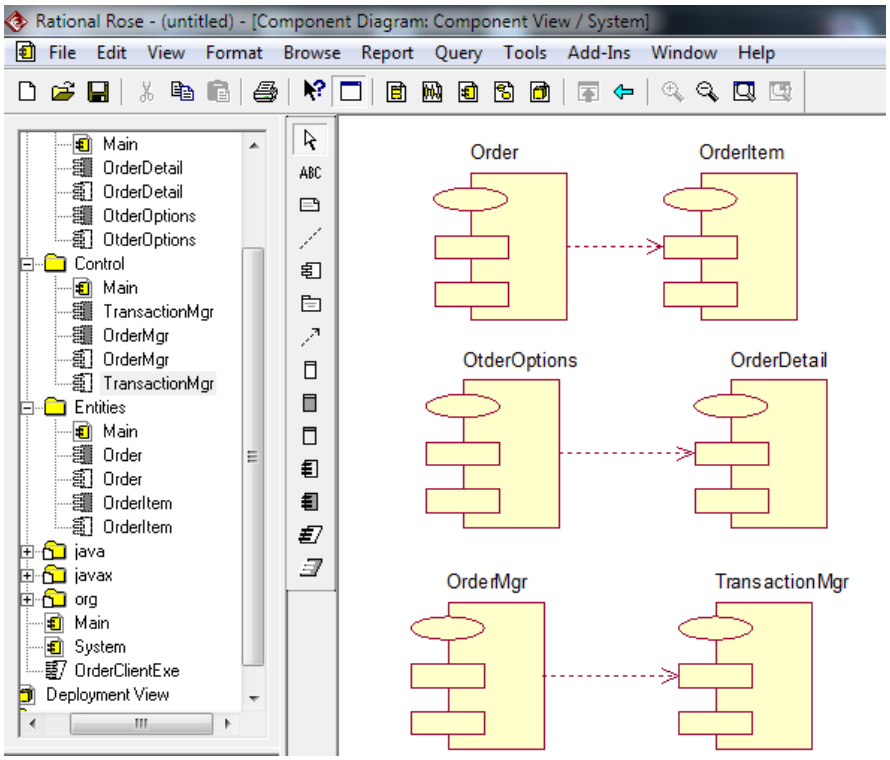

Рисунок 22 – Создание диаграммы компонентов системы

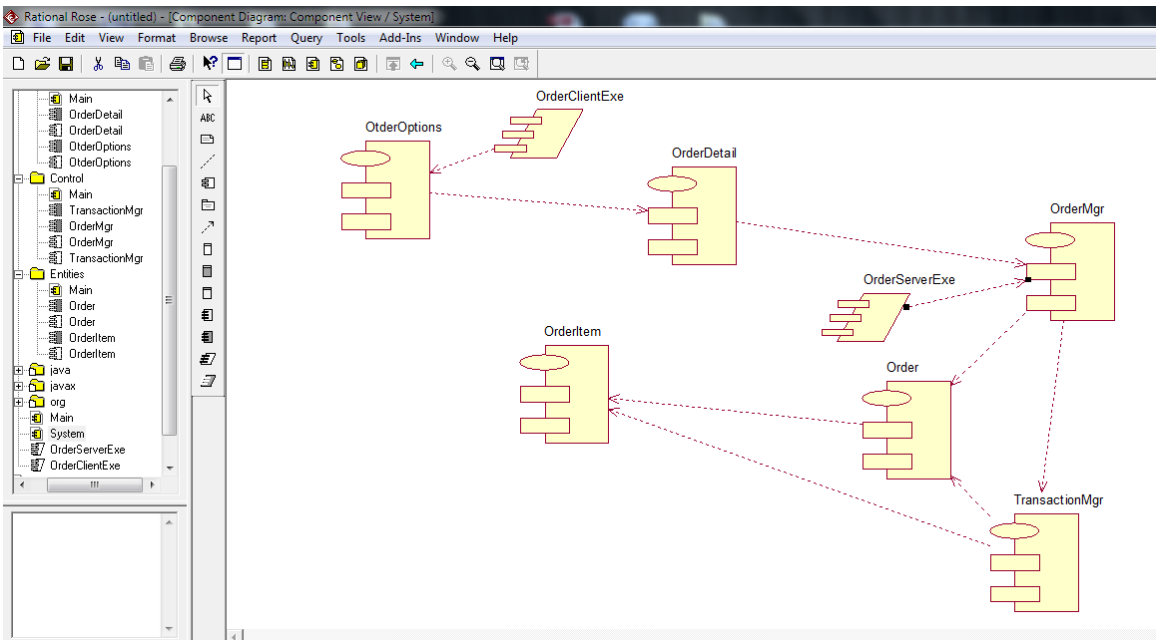

Рисунок 23 – Диаграмма компонентов системы

Практическая работа №6. Развитие методологий проектирования и управления информационными системами в фокусе различных научных подходов и технологических решений (2 часа)

ИТ-менеджмент охватывает управление всеми компьютерными и коммуникационными ресурсами предприятия.

Его основная задача состоит в создании и поддержании в работоспособном состоянии приложений и инфраструктуры, на которой они исполняются.

ИТ-проекты представляют собой проекты внедрения **новых информационных систем**, а также модернизацию существующих. При этом модернизация (изменения, дополнения) рассматривается как результат действий, выполненных по запросу и относящихся к функциональным или нефункциональным требованиям, которые не были специфицированы изначально, при разработке и внедрении системы.

Диаграмма размещения – в UML моделирует физическое развертывание артефактов на узлах. Узлы представляются как прямоугольные параллелепипеды с артефактами, расположенными в них, изображёнными в виде прямоугольников. Узлы могут иметь подузлы, которые представляются как вложенные прямоугольные параллелепипеды. Один узел диаграммы развертывания может концептуально представлять множество физических узлов, таких как кластер серверов баз данных.

Существует два типа узлов:

Узлы устройств — это физические вычислительные ресурсы со своей памятью и сервисами для выполнения программного обеспечения, такие как обычные ПК, мобильные телефоны.

Узел среды выполнения — это программный вычислительный ресурс, который работает внутри внешнего узла и который представляет собой сервис, выполняющий другие исполняемые программные элементы.

Практическая часть

Добавление узлов к диаграмме размещения:

Дважды щелкните мышью на представлении размещения в браузере, чтобы открыть диаграмму размещения.

На панели инструментов нажмите кнопку *Processor* (Процессор).

Щелкните на диаграмме, поместив туда процессор.

Введите имя процессора «Сервер базы данных».

Повторите шаги 2-4, добавив следующие процессоры:

Сервер приложения.

Клиентская рабочая станция № 1.

#Клиентская рабочая станция № 2.

На панели инструментов нажмите кнопку *Device* (Устройство).

Щелкните на диаграмме, поместив на нее устройство.

Назовите его «Принтер».

Добавление связей:

На панели инструментов нажмите кнопку *Connection* (Связь).

Щелкните на процессоре «Сервер базы данных».

Проведите линию связи к процессору «Сервер приложения».

Повторите этапы 1-3, добавив следующие связи:

От процессора «Сервер приложения» к процессору «Клиентская рабочая станция № **1».**

#От процессора «Сервер приложения» к процессору «Клиентская рабочая станция № 2».

#От процессора «Сервер приложения» к устройству «Принтер».

Добавление процессов:

Щелкните правой кнопкой мыши на процессоре «Сервер приложения» в браузере.

В открывшемся меню выберите пункт *New > Process* (Создать > Процесс).

Введите имя процесса *OrderServerExe.*

Повторите этапы 1-3, добавив еще процессы:

#На процессоре «Клиентская рабочая станция № 1» - процесс *OrderClientExe.*

#На процессоре «Клиентская рабочая станция № 2» - процесс *ATMClientEXE.*

Показ процессов на диаграмме:

Щелкните правой кнопкой мыши на процессоре «Сервер приложения».

В открывшемся меню выберите пункт *Show Processes* (Показать процессы).

Повторите этапы **1** и 2, показав процессы на следующих процессорах:

#Клиентская рабочая станция № **1.**

#Клиентская рабочая станция № 2.

Генерация кода C++

Ввод тел пакетов на диаграмму компонентов системы:

Откройте диаграмму компонентов системы.

Выберите в браузере *Entities:* тело пакета *Order.*

«Перетащите» тело пакета *Order* на диаграмму компонентов системы.

Повторите шаги 2 и 3 для следующих компонентов:

Entities: тело пакета *Orderltem. Boundaries:* тело пакета *OrderOptions. Boundaries:* тело пакета *OrderDetail. Control:* тело пакета *TransactionMgr. Control:* тело пакета *OrderMgr.*

Установка языка C++:

Откройте спецификацию компонента *Order* (спецификацию пакета) в пакете компонентов *Entities.*

Выберите в качестве языка C++.

Повторите шаги 1 и 2 для следующих компонентов:

Entities: тело пакета *Order.*

Entities: спецификация пакета *Orderltem. Entities:* тело пакета *Orderltem.*

Boundaries: спецификация пакета *OrderOptions. Boundaries:* тело пакета *OrderOptions.*

Boundaries: спецификация пакета *OrderDetail. Boundaries:* тело пакета *OrderDetail.*

Control: спецификация пакета *TransactionMgr. Control:* тело пакета *TransactionMgr.*

Control: спецификация пакета *OrderMgr. Control:* тело пакета *OrderMgr.*

Спецификация задачи *OrderClientExe.*

Спецификация задачи *OrderServerExe.*

Генерация кода C++:

Откройте диаграмму компонентов системы.

Выберите все объекты на диаграмме компонентов системы.

Выберите *Tools* > C++ > *Code Generation* в меню.

Выберите каталог для генерации кода.

Щелкните по кнопке OK и выполните генерацию в окне генерации кода.

Просмотрите результаты генерации (меню *Tools* >C++ > *Browse Header* и *Tools* > C++ > *Browse Body).*

Результаты выполнения задания по учебному примеру даны на рисунках ниже.

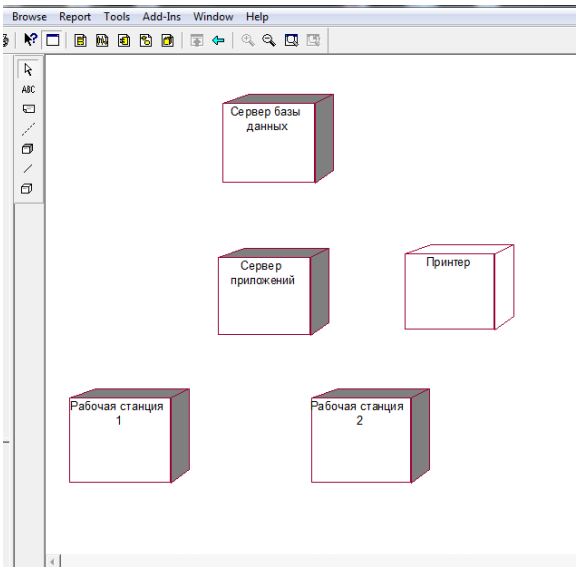

Рисунок 24 – Добавление узлов к диаграмме размещения

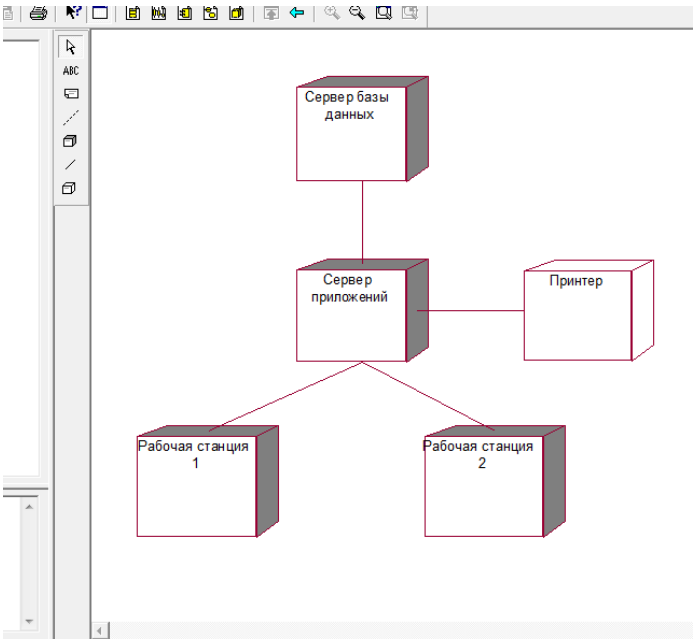

Рисунок 25 – Добавление связей

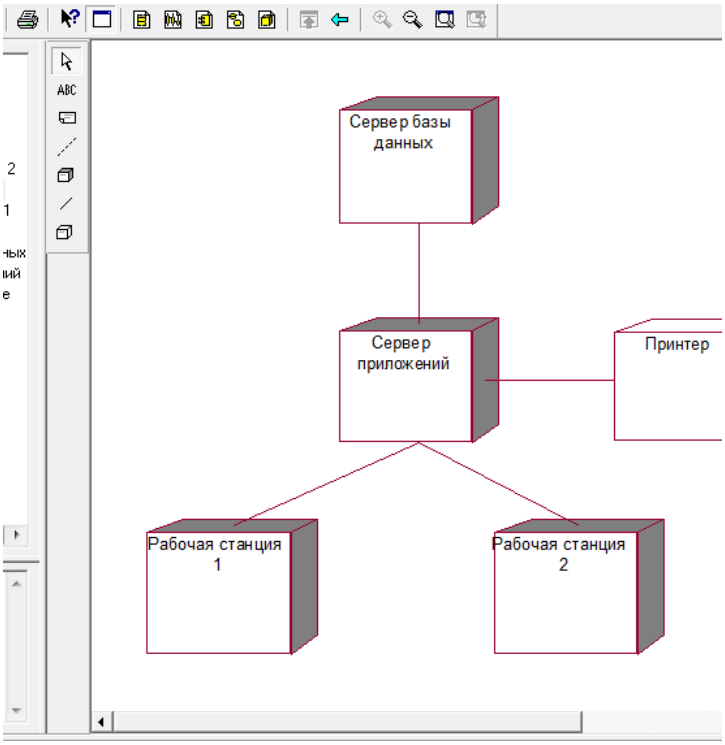

Рисунок 26 – Добавление процессов

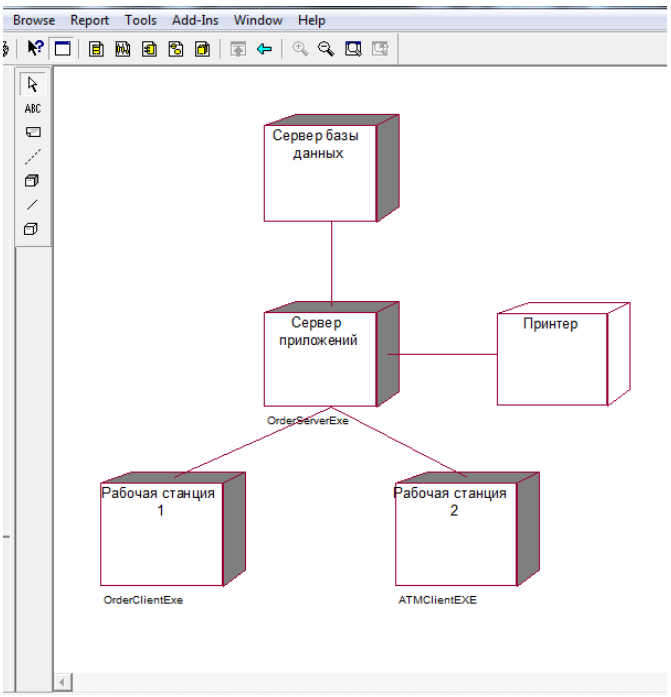

Рисунок 27 – Показ процессов на диаграмме

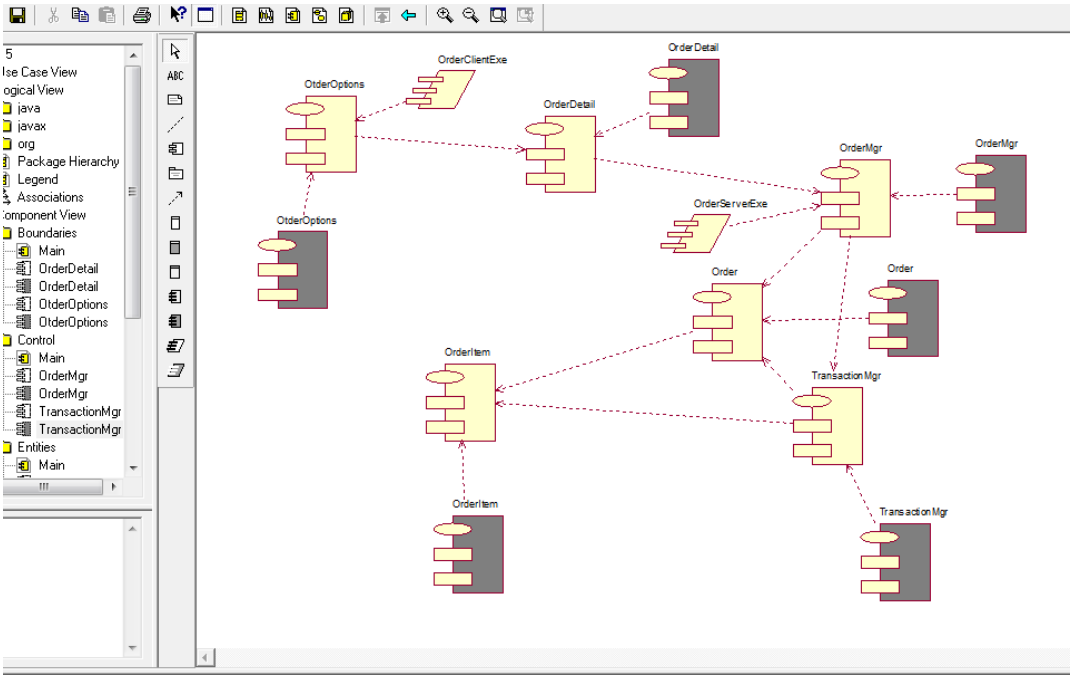

Рисунок 28 – Диаграмма компонентов системы *Order Entry*

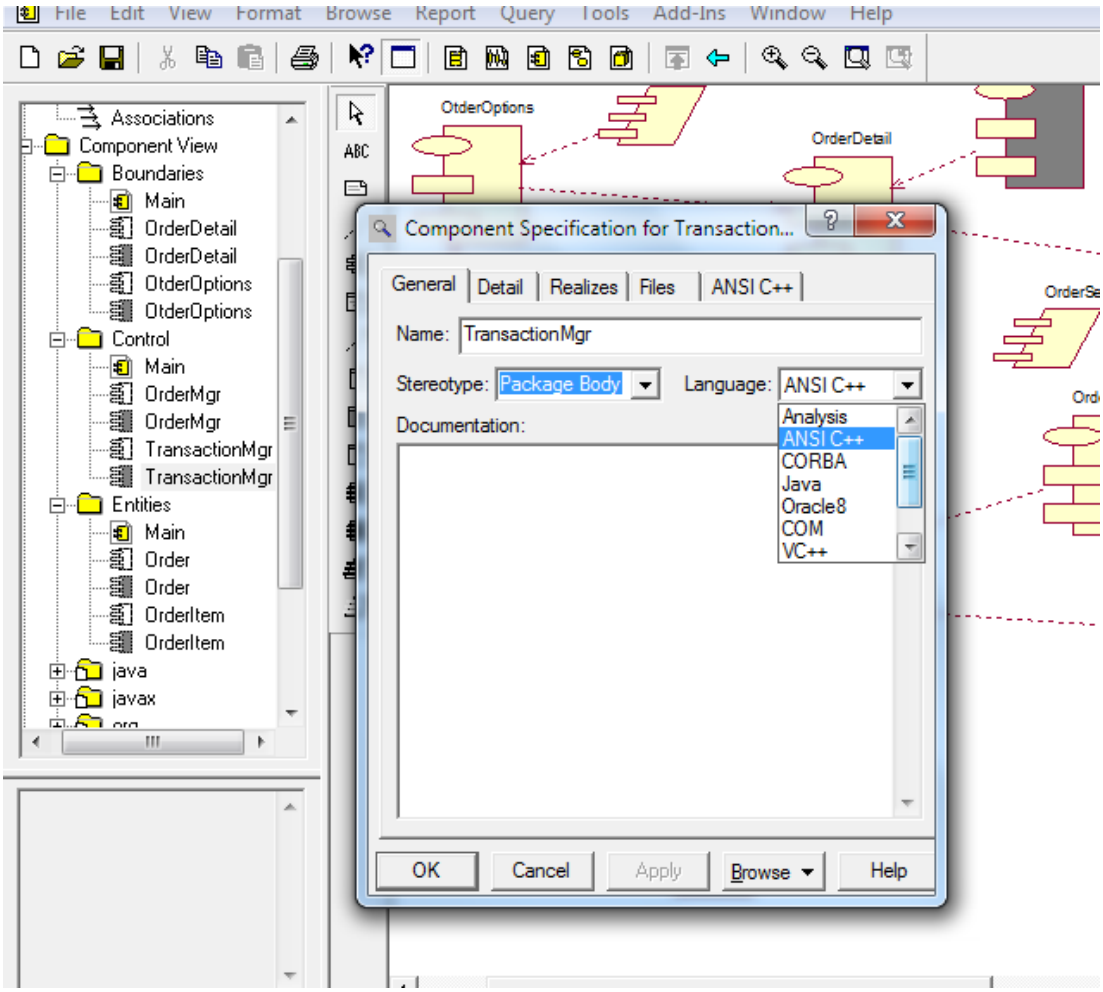

Рисунок 29 – Установка языка C+ +

```
#ifndef ORDER_H_HEADER_INCLUDED_9F3C506F
#define ORDER_H_HEADER_INCLUDED_9F3C506F

//##ModelId=6056014D0365
class Order
{
  public:
    //##ModelId=605601D4005D
    Create : Boolean();

    //##ModelId=605601D903CB
    Boolean SetInfo(Integer OrderNum, String Customer, Date OrderDate, Date FillDate, Integer ID = 0);

    //##ModelId=605601E500B8
    String GetInfo : String();

  private:
    //##ModelId=60BCFFCC0237
    Integer OrderNumber;
    //##ModelId=60BCFFE402AB
    String CustomerName;
    //##ModelId=60BCFFF3001A
    Date OrderDate;
    //##ModelId=60BCFFFB03DA
    Date OrderFillDate;
};


#endif /* ORDER_H_HEADER_INCLUDED_9F3C506F */
```

Рисунок 30 – Листинг кода