|  |
| --- |
| СОДЕРЖАНИЕ  [Введение 6](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672992)  [1 Постановка задачи 7](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672993)  [2 Описание предметной области 9](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672994)  [3 Морфологическая модель 10](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672995)  [4 Функциональная модель 12](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672996)  [5 Диграмма состояний 14](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672997)  [6 Диаграмма деятельности 16](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672998)  [7 Диаграмма «сущность-связь» 18](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672999)  8 Диаграмм[а компонентов 20](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672999)  9 Диаграмм[а размещения 22](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672999)  [10 Диграмма классов 24](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672997)  [11 Диграмма взаимодействия 26](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672997)  12 Теория о BPWIN [29](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672999)  12 Теория о IBM Rational Rose [32](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311672999)  [Выводы 34](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311673013)  [Список использованной литературы 35](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311673014)  [Приложение А Техническое задание 36](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311673015)  [Приложение Б Листинг классов 38](file:///E:\учеба\2%20курс\алгоритмы%20и%20структуры%20данных\курсач\варианты\18_двоичные%20Б%20деревья\ПЗ_v2.docx#_Toc311673017) |

ВВЕДЕНИЕ

Визуальное моделирование - это способ представления идей и проблем реального мира с помощью моделей. Модель помогает понять проблему всем участникам, задействованным в реализации проекта на различных этапах: заказчику, эксперту, аналитику, проектировщику, автору документации, программисту и др. Моделирование обеспечивает более точную оценку необходимых ресурсов, четкую проработку планов и эффективное функционирование создаваемых систем.

Нотация - важная составляющая любой модели, своего рода связующее звено между процессами. Унифицированный язык моделирования (UML) предлагает достаточно полную нотацию, которая расширяется при переходе от анализа к проектированию.

Успешно разработанный проект удовлетворяет или превосходит ожидание заказчика, выполняется в срок с оптимальными затратами и может быть адаптирован к изменению условий. Жизненный цикл разработки должен способствовать творческим и новаторским идеям. Правильно управляемый итеративный и инкрементальный жизненный цикл обеспечивает необходимый контроль и поддерживает творческий процесс на нужном уровне. В итеративном и инкрементальном жизненном цикле разработка осуществляется с помощью серии версий, которые развиваются в направлении конечной системы. Каждая версия состоит из одного или более компонентов процесса: построение бизнес - модели, определение требований к системе, анализ, проектирование, реализация, тестирование и внедрение.  
В качестве средства управления итеративным и инкрементальным жизненным циклом разработки применяется методика Rational Unified Process, с помощью которой можно подробно описать технические и организационные аспекты разработки программного обеспечения на стадиях определения требований к системе, анализа и проектирования.

Семейство продуктов Rational Rose призвано обеспечить разработчика программ полным набором инструментов визуального моделирования для эффективного решения сложных бизнес-задач с использованием архитектуры клиент/сервер, распределенных сред и систем реального времени.

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Общие определения

Разрабатываемая программа предназначена для хранения и структуризации данных системы банкомата. Данные, хранимые в проектируемой системе:

– информация о ПИН-кодах;

– информация о транзакциях;

– информация о банковских счетах;

1.2 Цель и задачи разработки

Целью данной работы является проектирование и объектно-ориентированный анализ программного продукта «Банкомат».

Проектируемый продукт должен помочь пользователю в организации работы банкомата.

1.3 Постановка задачи разработки программного продукта

Необходимо спроектировать и проанализировать программный продукт для банкомата. Он должен обеспечивать такие функции:

– работа с базой информации о клиентах.

– создание транзакций;

– создание отчетов;

1.4 Задачи и обязанности группы разработчиков

1.4.1 Руководитель проекта

Принимает решения, осуществляет координацию действий. Утверждает техническое задание, сроки и бюджет проекта. Руководитель – главный человек в команде, ответственность за срыв сроков или провал проекта ложится на него.

1.4.2 Аналитик

Анализирует историю предметной области с последующим формированием критериев этой истории. Осуществляет контроль над данными критериями для выявления закономерностей, определение нормативов. Определяет причину отклонения показателей и индикаторов от норматива, внесение своих предложений по улучшению данных систем. Прогнозирует показатели проделанной работы.

1.4.3 Тестер

Изучает программный продукт и ищет ошибки, сбои работы продукта. Очень важно, чтобы это был независимый человек. Нежелательно возлагать эти функции на кого-либо из членов команды

1.4.4 Программист

На основании технического задания и проектной документации выполняет кодирование CGI-программ. Предоставляет непосредственно сам программный продукт данной системы в виде кода программы. Должен быть высококвалифицированным специалистом, допускающим менее ошибок, чтобы потом не приходилось переделывать систему. Иметь большой опыт в сфере проектирования крупный проектов.

2 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Работа с банкоматом начинается после того как клиент вставляет свою пластиковую карту в приёмник пластиковых карт банкомата. Клиенту предлагается ввести секретный ПИН-код, для идентификации информации о пользователе. После ввода ПИН-кода банкомат проверяет правильность введённого числа, сверяя его с кодом, хранящимся на карте. Если код указан неверно, пользователю предоставляется ещё две попытки для ввода правильного ПИН-кода. В случае повторных неудач, карта перемещается в хранилище пластиковых карт и сеанс обслуживания заканчивается. В случае правильного ввода пользователю предлагают выбрать язык на каком будет вестись обслуживание. После этого банкомат предлагает пользователю выбрать операцию. При выборе операции «Снять наличные», банкомат предлагает указать сумму наличных которые нужно выдать пользователю. Банкомат шифрует информацию которая находиться на магнитной полосе карты(ПИН-код, номер кредитной карты, паспортные данные клиента, номер моб. телефона клиента), запрашиваемую операцию «Снять наличные», формирует и посылает запрос центральному компьютеру банка. В банке проверяется состояние личного счета клиента, и если он не заблокирован и там есть требуемая сумма, банкомату дается команда на выдачу денег, после чего банкомат проверяет, имеется ли требуемая сумма в его хранилище. Если такой суммы нет, тогда выводиться сообщение «Не возможно произвести операцию, в банкомате недостаточно денег для выдачи». И после этого посылается запрос банку, о том что в этом банкомате закончились деньги. Если такая сумма есть в банкомате, тогда выводиться сообщение «Выньте карту» и после того как карта будет извлечена из приёмника, банкомат выдает наличные в лоток выдачи. Банкомат запрашивает, нужно ли печатать справку? Если пользователь выбирает «Печатать справку», то напечатать справку, если нет, тогда пропустить этот шаг и выйти в главное меню.

Если выбрана операция «Проверка наличных», то производится запрос в банк об информации личного счета пользователя, после чего выводиться на экран остаток на счету.

3 МОРФОЛОГИЧЕСКАЯ МОДЕЛЬ

Морфологическая модель описывает состав и структуру системы, ее взаимосвязи с окружающей средой.

Описание системы начинается с представления ее в виде черного ящика. Границы входа и выхода системы выделяются в первую очередь. Это позволяет определить границы системы. Под входами понимают то, что необходимо системе для получения выхода. Все, с чем связанна система попадает во внешнюю среду системы.

После определения внешней среды, входов и выходов системы определяется состав системы, т.е. основные подсистемы и элементы. Исходя из цели моделирования, исходная система декомпозируется на части. Каждая из этих частей может быть тоже декомпозирована. Элементом системы будем называть некоторую неделимую часть.

Существует несколько способов описания морфологической модели. Одним из методов является описание морфологической модели в матричном виде. В основе метода лежит гиперкомплексная матрица (под гиперкомплексностью понимается свойство матрицы, состоящее в том, что ее элементами являются объекты разной природы).

Построение гиперкомплексной матрицы происходит в несколько этапов.

1. Определение числа иерархических уровней и числа элементов на каждом уровне.

2. Установление взаимосвязи между элементами и подсистемами на каждом уровне.

3. Формирование матрицы в виде квадрата, условная длина стороны которого определяется общим числом элементов на самом нижнем уровне иерархии.

4. Сторона квадрата разбивается на части, число которых равно количеству элементов на самом высоком уровне иерархии.

5. Полученный квадрат на главной диагонали в свою очередь разбивается на части, соответствующие числу элементов на следующем уровне иерархии. Процесс иерархического разбиения продолжается до тех пор, пока все уровни будут учтены.

6. По главной диагонали выписываются элементы системы, представляя тем самым модель состава системы.

Рассмотрим построение морфологической модели банкомата.

Основными подсистемами являются банк, банкомат, клиент.

На рисунке 3.1 представлена гиперкомплексная матрица для банкомата.

Банк

Банкомат

Клиент

Рисунок 3.1 – Морфологическая модель Банкомата

4 ФУНКЦИОНАЛЬНАЯ МОДЕЛЬ

Функциональная модель описывает порядок работы системы. Существуют различные формальные методики описания функционирования системы. Одной из наиболее распространенных методик является IDEF технология. Особенностями IDEF являются:

- четкое определение границы системы;

- все функции рассматриваются исходя из цели создания системы, в рамках методологии получила название точки зрения;

- описывается функционирование верхнего уровня, а каждая функция

декомпозируется на несколько уровней.

Рассмотрим технологию построения модели в рамках IDEF.

Каждая IDEF диаграмма содержит блоки и дуги. Блоки изображают функцию системы, а дуги связывают блоки и описывают взаимосвязь между ними.

Функциональные блоки изображают прямоугольниками, внутри которых содержатся название. Название блока должно содержать глаголы (выполнить задания, оценить качество). На диаграмме должно быть не меньше трех и не больше пяти блоков.

Блоки размещаются на диаграмме по степени важности (доминирования). Доминирующим блоком может быть первый блок цепочки или блок, выход которого является управлением для других блоков диаграммы.

Наиболее доминирующий блок располагается в верхнем левом углу диаграммы, а наименее доминирующий блок в правом нижнем углу. Порядок доминирования обозначен цифрой, расположенный в правом нижнем углу блока.

Построение модели начинается с контекстной диаграммы. На ней система представляется виде одного блока. Дуги отображают связи системы с внешними системами. Название дуг блока являются достаточно общими, так как характеризуют всю систему целиком. Кроме того, на контекстной диаграмме указывается цель создания модели системы и точка зрения (с каких позиций рассматривается система.) Контекстная диаграмма обозначается как A0. Далее система декомпозируется на три – пять блоков и обозначается А1. Затем каждый из блоков в свою очередь декомпозируется на три – пять блоков на диаграмме следующего уровня. Диаграмму обозначаются буквой А с цепочкой номеров декомпозированных блоков.

На рисунке 4.1 представлена контекстная диаграмма для банкомата.

А 0

Код банка,

код карточки

Пароль,

вид проводки,

тип счета, суммаВыдача

денег,

квитанции,

сообщение

на экране

Клиент Карточка

Выполнить транзакцию

0р. 0

Рисунок 4.1 – Контекстная диаграмма банкомат

5 ДИАГРАММА СОСТОЯНИЙ

Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Понятие автомата в контексте UML обладает довольно специфической семантикой, основанной на теории автоматов. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние. Диаграммы состояний могут быть вложены друг в друга, образуая вложенные диаграммы более детального представления отдельных элементов модели. Для понимания семантики конкретной диаграммы состояний необходимо представлять не только особенности поведения моделируемой сущности, но и знать общие сведения по теории автоматов.

Автомат в языке UML представляет собой некоторый формализм для моделирования поведения элементов модели и системы в целом. В метамодели UML автомат является пакетом, в котором определено множество понятий, необходимых для представления поведения моделируемой сущности в виде дискретного пространства с конечным числом состояний и переходов. С другой стороны, автомат описывает поведение отдельного объекта в форме последовательности состояний, которые охватывают все этапы его жизненного цикла, начиная от создания объекта и заканчивая его уничтожением. Каждая диаграмма состояний представляет некоторый автомат.

В языке UML под состоянием понимается абстрактный метакласс, используемый для моделирования отдельной ситуации, в течение которой имеет место выполнение некоторого условия. Состояние может быть задано в виде набора конкретных значений атрибутов класса или объекта, при этом изменение их отдельных значений будет отражать изменение состояния моделируемого класса или объекта.

Диаграмма состояний для проектируемой предметной области представлена на рисунке 5.1.



Рисунок 5.1 – Диаграмма состояний.

6 ДИАГРАММА ДЕЯТЕЛЬНОСТИ

Диаграмма деятельности — диаграмма, на которой показано разложение некоторой деятельности на её составные части. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Традиционно для этой цели использовались блок-схемы или структурные схемы алгоритмов. Каждая такая схема акцентирует внимание на последовательности выполнения определенных действий или элементарных операций, которые в совокупности приводят к получению желаемого результата.

Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Применяемая в них графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления не деятельностей, а действий, и в отсутствии на переходах сигнатуры событий. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой, операции в предыдущем состоянии.

Таким образом, диаграммы деятельности можно считать частным случаем диаграмм состояний. Именно они позволяют реализовать в языке UML особенности процедурного и синхронного управления, обусловленного завершением внутренних деятельностей и действий.

В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий.

Диаграмма деятельности для проектируемой предметной области представлена на рисунке 6.1.



Рисунок 6.1 – Диаграмма деятельности.

7 ДИАГРАММА «СУЩНОСТЬ-СВЯЗЬ»

Модель сущность-связь (ER-модель) — [модель данных](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), позволяющая описывать [концептуальные схемы](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0) [предметной области](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C).

ER-модель используется при высокоуровневом (концептуальном) [проектировании баз данных](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B0%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Во время [проектирования баз данных](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B0%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) происходит преобразование ER-модели в конкретную [схему базы данных](http://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) на основе выбранной модели данных ([реляционной](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), [объектной](http://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85&action=edit&redlink=1), [сетевой](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) или др.).

ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма сущность-связь (ER-диаграмма).

Понятия ER-модель и ER-диаграмма часто ошибочно не различают, хотя для визуализации ER-моделей предложены и другие графические нотации.

Согласно нотации Питера Чена множества сущностей изображаются в виде прямоугольников, множества отношений изображаются в виде ромбов. Если сущность участвует в отношении, они связаны линией. Если отношение не является обязательным, то линия пунктирная. Атрибуты изображаются в виде овалов и связываются линией с одним отношением или с одной сущностью.

Диаграмма сущность-связь для разрабатываемой предметной области представлена на рисунке 7.1.



Рисунок 7.1 – Диаграмма «сущность-связь»

8 ДИАГРАММА КОМПОНЕНТОВ

Диаграмма компонентов — статическая структурная [диаграмма](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), показывает разбиение программной системы на структурные [компоненты](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать [файлы](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB), библиотеки, модули, исполняемые файлы, пакеты и т. п.

Компоненты связываются через зависимости, когда соединяется требуемый [интерфейс](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) 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.

В первую очередь, необходимо решить, из каких физических частей (файлов) будет состоять программная система. На этом этапе следует обратить внимание на такую реализацию системы, которая обеспечивала бы не только возможность повторного использования кода за счет рациональной декомпозиции компонентов, но и создание объектов только при их необходимости.

После общей структуризации физического представления системы необходимо дополнить модель интерфейсами и схемами базы данных.

Наконец, завершающий этап построения диаграммы компонентов связан с установлением и нанесением на диаграмму взаимосвязей между компонентами, а также отношений реализации. Эти отношения должны иллюстрировать все важнейшие аспекты физической реализации системы, начиная с особенностей компиляции исходных текстов программ и заканчивая исполнением отдельных частей программы на этапе ее выполнения. Для этой цели можно использовать различные виды графического изображения компонентов.

Диаграмма компонентов для проектируемой предметной области представлена на рисунке 8.1.



Рисунок 8.1 – Диаграмма компонентов.

9 ДИАГРАММА РАЗМЕЩЕНИЯ

Диагра́мма развёртывания (размещения), в [UML](http://ru.wikipedia.org/wiki/UML) моделирует физическое развертывание [артефактов](http://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%82%D0%B5%D1%84%D0%B0%D0%BA%D1%82) на [узлах](http://ru.wikipedia.org/w/index.php?title=Node_(UML)&action=edit&redlink=1). Например, чтобы описать веб-сайт диаграмма развертывания должна показывать, какие аппаратные компоненты («узлы») существуют (например, веб-сервер, сервер базы данных, сервер приложения), какие программные компоненты («артефакты») работают на каждом узле (например, веб-приложение, база данных), и как различные части этого комплекса соединяются друг с другом.

Узлы представляются как прямоугольные параллелепипеды с артефактами, расположенными в них, изображенными в виде прямоугольников. Узлы могут иметь под-узлы, 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:

* Узел устройства
* Узел среды выполнения

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 предоставляет собой сервис, выполняющий другие исполняемые программные элементы.

Диаграмма развёртывания для проектируемой предметной области представлена на рисунке 9.1.



Рисунок 9.1 – Диаграмма развёртывания.

10 ДИАГРАММА КЛАССОВ

В [UML](http://ru.wikipedia.org/wiki/UML) диаграмма классов является типом диаграммы статической структуры. Она описывает структуру системы, показывая её [классы](http://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), их [атрибуты](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0) и [операторы](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_(%D1%8F%D0%B7%D1%8B%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), а также [взаимосвязи](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) этих классов.

Взаимосвязь — это особый тип логических отношений между сущностями, показанных на [диаграммах](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) классов и [объектов](http://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BE%D0%B2&action=edit&redlink=1). В UML’е представлены следующие виды отношений:

[Ассоциация](http://ru.wikipedia.org/w/index.php?title=%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)&action=edit&redlink=1) показывает, что объекты одной сущности (класса) связаны с объектами другой сущности.

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

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

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

[Агрегация](http://ru.wikipedia.org/wiki/%D0%90%D0%B3%D1%80%D0%B5%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) — это разновидность ассоциации при отношении между целым и его частями. Как тип ассоциации агрегация может быть именованной. Одно отношение агрегации не может включать более двух классов (контейнер и содержимое).

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

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

Диаграмма классов для проектируемой предметной области представлена на рисунке 10.1.



Рисунок 10.1 – Диаграмма классов.

11 ДИАГРАММА ВЗАИМОДЕЙСТВИЯ

В языке UML взаимодействие элементов рассматривается в информационном аспекте их коммуникации, т. е. взаимодействующие объекты обмениваются между собой некоторой информацией. При этом информация принимает форму законченных сообщений. Другими словами, хотя сообщение и имеет информационное содержание, оно приобретает дополнительное свойство оказывать направленное влияние на своего получателя. Это полностью согласуется с принципами ООАП, когда любые виды информационного взаимодействия между элементами системы должны быть сведены к отправке и приему сообщений между ними.

Для моделирования взаимодействия объектов в языке UML используются соответствующие диаграммы взаимодействия. Говоря об этих диаграммах, имеют в виду два аспекта взаимодействия. Взаимодействия объектов можно рассматривать во времени, и тогда для представления временных особенностей передачи и приема сообщений между объектами используется диаграмма последовательности.

На диаграмме последовательности изображаются исключительно те объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами. Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени.

Линия жизни объекта изображается пунктирной вертикальной линией, ассоциированной с единственным объектом на диаграмме последовательности. Линия жизни служит для обозначения периода времени, в течение которого объект существует в системе и, следовательно, может потенциально участвовать во всех ее взаимодействиях. Если объект существует в системе постоянно, то и его линия жизни должна продолжаться по всей плоскости диаграммы последовательности от самой верхней ее части до самой нижней.

В процессе функционирования объектно-ориентированных систем одни объекты могут находиться в активном состоянии, непосредственно выполняя определенные действия или в состоянии пассивного ожидания сообщений от других объектов. Чтобы явно выделить подобную активность объектов, в языке UML применяется специальное понятие, получившее название фокуса управления.

Диаграмма последовательности для проектируемой предметной области представлена на рисунке 11.1.



Рисунок 11.1 Диаграмма последовательности

Рисунок 11.1 Диаграмма последовательности (продолжение)



12 ТЕОРИЯ О BPWIN

BPwin - мощный инструмент моделирования, который используется для анализа, документирования и реорганизации сложных бизнес-процессов. Модель, созданная средствами BPwin, позволяет четко документировать различные аспекты деятельности - действия, которые необходимо предпринять, способы их осуществления, требующиеся для этого ресурсы и др. Таким образом, формируется целостная картина деятельности предприятия - от моделей организации работы в маленьких отделах до сложных иерархических структур. При разработке или закупке программного обеспечения модели бизнес-процессов служат прекрасным средством документирования потребностей, помогая обеспечить высокую эффективность инвестиций в сферу IT. В руках же системных аналитиков и разработчиков BPwin - еще и мощное средство моделирования процессов при создании корпоративных информационных систем (КИС).

Модели BPwin дают основу для осмысления бизнес-процессов и оценки влияния тех или иных событий, а также описывают взаимодействие процессов и потоков информации в организации. Неэффективная, высокозатратная или избыточная деятельность может быть легко выявлена и, следовательно, усовершенствована, изменена или устранена в соответствии с общими целями организации.

Внешние обстоятельства зачастую вынуждают вносить изменения в деятельность организации. Последствия этих изменений должны быть тщательно изучены и осмыслены перед тем, как система будет переделана с их учетом. BPwin может помочь пользователю на протяжении всего цикла, предоставив возможность оптимизировать бизнес-процесс, которого коснутся эти изменения.

С помощью BPwin пользователь может сделать свою работу более продуктивной. Действия и другие объекты создаются буквально несколькими щелчками мыши, а затем легко отбуксированы в нужное место. Интерфейс BPwin, выполненный в стиле "проводника" облегчает навигацию и редактирование сложных процессов с иерархической структурой. Развитые возможности изменения масштаба представления позволяют быстро найти и сосредоточиться на необходимой для работы части модели процесса.

BPwin позволяет:

* Обеспечить эффективность операций, рассматривая текущие бизнес-операции через мощные инструменты моделирования;
* Совершенствовать бизнес-процессы, формулируя и определяя альтернативные реакции на воздействия рынка;
* Быстро исключать непродуктивные операции, легко и интуитивно сопоставляя операционные изменения. Неэффективные, неэкономичные или избыточные операции могут быть легко выявлены и, следовательно, улучшены, изменены или вовсе исключены - в соответствии с целями компании.

BPwin автоматизирует решение многих вспомогательных задач, которые обычно связаны с построением модели процесса, и обеспечивает логическую строгость, необходимую для достижения корректных и согласованных результатов. BPwin отслеживает связи в диаграммах, сохраняя их целостность при внесении изменений в модель. Динамическая "подсветка" объектов служит подсказкой при построении модели и предостерегает от повторения распространенных ошибок в моделировании. Кроме этого, BPwin поддерживает заданные пользователем свойства, что позволяет вносить соответствующую вашим потребностям информацию.

Различные варианты оформления с гибким использованием шрифтов, цвета и других средств форматирования придают документам большую наглядность. Пользователь может просматривать и распечатывать общее представление своей модели в виде древовидных диаграмм. С помощью средства создания FEO диаграмм (For Exposition Only) вариации модели или проблемной области можно проанализировать, не внося изменений в основную модель. Возможности настройки пользовательских палитр цветов позволяют легко адаптировать вид документов в соответствии с особенностями принтера или демонстрационного проектора без внесения изменений в саму модель.

BPwin позволяет адаптироваться к постоянно меняющимся реалиям современного рынка.

Конкуренция предполагает мгновенную реакцию на новые возможности, угрозы и потребности покупателей. Сегодня постоянные изменения стали нормой. Поскольку бизнес-процессы становятся все более сложными, требуются решения, представляющие интегрированный взгляд на функционирование компании.

Модели бизнес-процессов в масштабах всего предприятия могут оказаться очень сложными. BPwin предоставляет возможности, призванные облегчить инкрементальную разработку моделей и разграничение процессов. Средства объединения дают возможность нескольким проектным группам проводить анализ различных фрагментов деятельности, а затем создать глобальное представление. Иногда бывает необходимо более детально изучить определенную часть общей модели. BPwin позволяет разбить модель на фрагменты, поработать с ними, а затем вновь объединить их в одно целое.

BPwin совмещает в одном инструменте средства моделирования функций (IDEF0), потоков данных (DFD) и потоков работ (IDEF3), координируя эти три основных аспекта бизнеса для соответствия потребностям бизнес-аналитиков и системных аналитиков. BPwin позволяет повторно использовать ключевую информацию моделирования с точки зрения базовых аспектов, чтобы определить точки конфликтов и, в конечном счете, достичь их согласования.

С помощью функционального моделирования (нотация IDEF0), можно провести систематический анализ бизнеса, сосредоточившись на регулярно решаемых задачах (функциях), свидетельствующих об их правильном выполнении показателях, необходимых для этого ресурсах, результатах и исходных материалах (сырье).

Моделирование потоков данных (DFD), часто используемое при разработке программного обеспечения, сосредоточено вокруг потоков данных, передающихся между различными операциями, включая их хранение, для достижения максимальной доступности и минимального времени ответа. Такое моделирование позволяет рассмотреть конкретный процесс, проанализировать операции, из которых он состоит, а также точки принятия решений, влияющих на его ход.

Моделирование потоков работ (нотация IDEF3) позволяет рассмотреть конкретный процесс, проанализировать операции, из которых он состоит, а также точки принятия решений, влияющих на его ход.

13 ТЕОРИЯ О IMB RATIONAL ROSE

IBM Rational Rose - популярное средство визуального моделирования, которое считается стандартом де-факто среди средств визуального проектирования приложений. Этот продукт входит в состав пакета IBM Rational Suite и предназначен для моделирования программных систем с использованием широкого круга инструментальных средств и платформ. Инструментальное средство  IBM Rational Rose расширяет возможности моделирования программных систем, выходящих за рамки платформы J2EE и инструментальных средств моделирования в составе IBM Rational Professional Bundle.

Являясь простым и мощным решением для визуальной разработки  информационных систем любого класса, Rational Rose позволяет создавать, изменять и проверять корректность модели. Rational Rose объединяет команду разработчиков на базе универсального языка моделирования UML, который определяет стандартную графическую символику для описания архитектуры ПО. Любые участники проекта - аналитики, специалисты по моделированию, разработчики и другие - могут использовать модели, построенные в Rational Rose, для большей эффективности создания конечного продукта.

Для архитекторов - использование Rational Rose обеспечивает архитекторов средством визуального моделирования на основе UML, позволяя им представлять всю архитектуру приложения с разных точек зрения. Итерационная разработка обеспечивает соответствие моделей программному коду ПО.

Для специалистов по БД и аналитиков данных - Rational Rose является единым инструментом, языком и нотацией для всей команды. Rational Rose Data Modeler обеспечивает поддержку БД, включая объектно-ориентированное отображение (mapping), генерацию схем и итерационную разработку.

Для разработчиков на Visual Studio и WinDNA - Rational Rose плотно интегрируется с MS Visual Studio и обеспечивает поддержку семантики и схемы WinDNA, визуализацию и итерационную разработку кода COM/ATL, MTS и ADO, настройку и открытую разработку шаблонов для генерации многоуровневых приложений WinDNA.

Для интернет-разработчиков и XML-разработчиков - Rational Rose является единственным решения, которое обеспечивает понятную визуализацию интернет-архитектуры, включая Web Application Extension для UML, обратное проектирование семантики из JSP- и ASP-файлов, автоматизацию Web Application Extension для UML, визуализацию самых сложных по структуре интернет-сайтов и улучшенную поддержку XML.

Для Java- и EJB-разработчиков - использование Rational Rose обеспечивает качественную поддержку всех аспектов разработки: разработку архитектур "тонкого" клиента, полную поддержку моделирования Enterprise Java Beans, полную интеграцию с такими распространенными Java IDE, как JBuilder, Visual Age, Forte и Visual Cafe, а также совместим со всеми поддерживаемыми версиями J2SE и J2EE.

Только хорошо спроектированные приложения могут отвечать требованиям заказчика и быстро адаптироваться к изменившимся условиям ведения бизнеса. Для решений на WinDNA, Enterprise Java, Web и XML или для встроенных приложений Rational Rose ускоряет разработку, используя проверенные архитектурные модели для каждого из решений.

Rational Rose также является платформой для автоматизации использования лучших архитектурных принципов вместе со специальными технологиями.

Rational Rose позволяет визуализировать, понять и уточнить требования и архитектуру перед началом создания кода. Обладая способностью представлять пользовательский интерфейс отдельно от бизнес-логики и данных, Rational Rose позволяет держать контроль над разработкой ПО. Это дает возможность исключить затраты сил на ненужные переделки. Использование одного инструмента на протяжении всего жизненного цикла разработки помогает строить "правильную" систему для Вашего заказчика. Архитектурная модель может быть ассоциирована как с моделями бизнес-процессов, так и с системными требованиями. Rational Rose обеспечивает обратное проектирование на основе анализа существующих библиотек и компонентов, и создания моделей и их тестирования. Это снижает риски, увеличивает эффективность и легкость разработки, предсказуемость, позволяет сэкономить средства и время.

Rational Rose является единственным решением, с помощью которого можно изменять код, модель, код и модель одновременно, проводить в любое время синхронизацию между ними.

В связи с особенностями работы разработчиков и консультантов менеджеры по проектам озабочены вопросами сохранения интеллектуального капитала, накопленного в ходе выполнения проектов. Без систематизации хранения информация процедуры и процессы, основанные на полученном опыте, могут быть утеряны с уходом ключевых участников из проекта. С увеличением сложности проекта увеличиваются сроки проекта и риски потери информации. Использование визуального моделирования сохраняет знания о структуре проекта в информационной системе компании.

Все участники проекта могут непосредственно видеть элементы проекта, UML-модель, а не получать информацию из кода, старых записей или в устном виде. Модели содержат важные для успешного проекта артефакты, описания, проектные требования, примеры использования, классы и интерфейсы.

# АННЫХ ИСТОЧНИКОВ

1. Г. Буч Д. Рамбо, А. Джекобсон, Программирование: Язык UML. Руководство пользователя: Питер, 2005. -205 стр.
2. М. Фаулер, К. Скотт, Программирование: UML: Основы