ВВЕДЕНИЕ

Рассматриваемая дипломная работа написана на базе Донецкой ОАО Донецкая мануфактура для магазина Cleonelly.

Одним из ведущих направлений деятельности ОАО Донецкая мануфактура выпускает швейные изделия в широком ассортименте, преимущественно купальные халаты, простыни и полотенца. Кроме того, предприятие производит крашеную хлопчатобумажную пряжу для ткацкого и трикотажного производства.

Развитие автоматизированных информационных технологий идет параллельно с появлением новых видов технических средств обработки и передачи информации, совершенствованием организационных форм использования ЭВМ, насыщением инфраструктуры новыми средствами коммуникаций. Развитие рыночных отношений привело к появлению новых видов предпринимательской деятельности и, прежде всего, к созданию фирм, занятых информационным бизнесом, разработкой информационных технологий, их совершенствованием, распространением компонентов автоматизированных информационных технологий, в частности программных продуктов, автоматизирующих информационные и вычислительные процессы. К их числу относят также вычислительную технику, средства коммуникаций, офисное оборудование и специфические виды услуг – информационное, техническое и консультационное обслуживание, обучение и т.п. Это способствовало быстрому распространению и эффективному использованию информационных технологий в управленческих и производственных процессах, практически к повсеместному их применению и большому многообразию.

Предприятия, занимающиеся проектированием и разработкой устройств различного назначения, в настоящее время широко используют различные средства как автоматизированного проектирования – САПР (CAD), так и мониторинга производственных процессов – АСУТП

(SCADA/DCS). Однако для устройств собственной разработки необходимо разрабатывать собственные средства контроля их работоспособности и анализа качества продукции.

Технологический процесс учета продукции на складе в магазине Cleanelly включает этап ведение учетности продаваемой продукции.

Целью настоящего дипломного проекта является реализация автоматизированного рабочего места (АРМ) позволяющего осуществить учет продукции на складе магазина.

Для достижения вышеуказанной цели в необходимо решить следующие задачи:

- провести анализ бизнес-процессов магазина;
- исследовать информационные потоки, возникающие на этапе сдачи разрабатываемого изделия;
- разработать концептуальную и логическую модели данных;
- разработать программное обеспечение для АРМ учета продукции
- провести оценку экономической эффективности информационной системы.

1 РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

1.1 Анализ существующих решений

В настоящее время существует широкий спектр компаний, совмещающих как непосредственную разработку изделий, так разработку систем управления этими изделиями. Подобные системы разрабатываются как такими широко известными компаниями как 1:С предприятие и “Звезда”. В таких системах осуществляется контроль и учет материалов, и обработка полученной информации.

"1С:Предприятие" представляет собой систему прикладных решений, построенных по единым принципам и на единой технологической

платформе. Руководитель может выбрать решение, которое соответствует актуальным потребностям предприятия и будет в дальнейшем развиваться по мере роста предприятия или расширения задач автоматизации.

Система программ "1С:Предприятие" предназначена для решения широкого спектра задач автоматизации учета и управления, стоящих перед динамично развивающимися современными предприятиями. Решение актуальных задач учета и управления Состав программ системы "1С:Предприятие" ориентирован на актуальные потребности предприятий. Фирма "1С" выпускает тиражные программные решения, предназначенные для автоматизации типовых задач учета и управления на предприятиях. Отличительной особенностью тиражных решений фирмы "1С" является тщательная проработка состава функциональности, включаемой в типовые решения. Фирма "1С" анализирует опыт пользователей, применяющих программы системы "1С:Предприятие" и отслеживает изменение их потребностей.

К основным преимуществам моей системы Оптовая База можно отнести относительную низкую стоимость внедрения данной системы ,а также еще ряд преимуществ:

- Надежность создаваемых приложений. Программный комплекс (ПК) должен быть устойчив не только к ошибкам пользователей, но и к сбоям в системе коммуникаций.
- Удобство пользования интерфейсом;
- Высокий уровень безопасности системы, что подразумевает не только контроль доступности тех или иных ресурсов системы и защищенность информации на всех этапах функционирования, но и отслеживание выполняемых действий с высокой степенью достоверности.

1.2 Анализ предметной области

Особенность анализа предметной области состоит в том, что он позволяет увидеть всю совокупность операций организации.

Для проведения анализа и реорганизации бизнес-процессов предназначено CASE – средство верхнего уровня All Fusion Process Modeler (BPwin), поддерживающие методологии IDEF0 (функциональная модель), DFD (Dataflow Diagram) и IDEF3 (Workflow Diagram). BPwin является мощным программным продуктом для создания моделей, позволяющих анализировать, документировать и планировать изменения сложных бизнес-процессов. BPwin предлагает средство, для сбора всей необходимой информации о работе предприятия и графического изображения этой информации в виде целостной и непротиворечивой модели. [24]

С точки зрения функциональности системы. В рамках методологии IDEF0 (Integration Definition for Function Modeling) бизнес-процесс представляется в виде набора элементов-работ, которые взаимодействуют между собой, а также показывается информационные, людские и производственные ресурсы, потребляемые каждой работой. Функциональная модель предназначена для описания существующих бизнес-процессов на предприятии (так называемая модель AS-IS) и идеального положения вещей – того, к чему нужно стремиться (модель TO-BE). Методология IDEF0 предписывает построение иерархической системы диаграмм, т.е. единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействие с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция – система разбивается на подсистемы и каждая система описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее для достижения нужной степени подробности. [26]

Если в процессе моделирования нужно осветить специфические стороны технологии предприятия, BPwin позволяет переключиться на любой ветви модели на нотацию DFD или IDEF3. Диаграммы DFD (Data Flow Diagramming) могут дополнить то, что уже отражено в модели IDEF3,

поскольку они описывают потоки данных, позволяя проследить, каким образом происходит обмен информацией между бизнес-функциями внутри системы. В тоже время диаграммы DFD оставляют без внимания взаимодействие между бизнес-функциями. [26]

С точки зрения последовательности выполняемых работ. И еще более точную картину можно получить, дополнив модель диаграммами IDEF3. Этот метод привлекает внимание к очередности выполнения событий. В IDEF3 включены элементы логики, что позволяет моделировать и анализировать альтернативные сценарии развития бизнес-процесса.

Для рассмотрения бизнес – процессов выполняющихся на складе магазина, необходимо использовать только две методологии IDEF0 и DFD. Процесс моделирования какой-либо системы в IDEF0 начинается с определения контекста, т.е. наиболее абстрактного уровня описания системы или бизнес-процессов в целом.

Модель IDEF0. Для изучения бизнес-процессов «Формирование заказа поставщика», «Получение товара», «Отпуск товара», рассмотрим диаграммы которые представлены в виде IDEF0 диаграмме. IDEF0 система представляется как совокупность взаимодействующих работ или функций.

В основе методологии IDEF0 лежат четыре основных понятия.

Первым из них является понятие **функционального блока (Activity Box)**. Функциональный блок графически изображается в виде прямоугольника и олицетворяет собой некоторую конкретную функцию в рамках рассматриваемой системы

Каждая из четырех сторон функционального блока имеет свое определенное значение (роль), при этом:

- верхняя сторона имеет значение «Управления» (Control);
- левая сторона имеет значение «Вход» (Input);
- правая сторона имеет значение «Выход» (Output);
- нижняя сторона имеет значение «Механизм» (Mechanism).

Вторым «китом» методологии IDEF0 является понятие интерфейсной

дуги (Arrow). Графическим отображением интерфейсной дуги является однонаправленная стрелка. Каждая интерфейсная дуга должна иметь свое наименование (Arrow Label). С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. При этом стрелки, в зависимости от того в какую грань прямоугольника работы они входят или из какой грани выходят, делятся на:

- стрелки входа (входят в левую грань функционального блока) – изображают изменяемые в ходе выполнения работы данные или объекты;
- стрелки управления (входят в верхнюю грань функционального блока) – изображают правила и ограничения, благодаря которым выполняется работа;
- стрелки выхода (выходят из правой грани функционального блока) – изображают данные или объекты, появляющиеся в результате выполнения работы;
- стрелки механизма (входят в нижнюю грань функционального блока) – изображают ресурсы (например, оборудование, людские ресурсы).

Третьим основным понятием стандарта IDEF0 является декомпозиция (Decomposition). Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции.

Декомпозиция позволяет постепенно и структурировано представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Последним из понятий IDEF0 является глоссарий (Glossary). Для каждого из элементов IDEF0: диаграмм, функциональных блоков, интерфейсных дуг существующий стандарт подразумевает создание и поддержание набора соответствующих определений, ключевых слов, повествовательных изложений и т.д., которые характеризуют объект,

отображенный данным элементом. Этот набор называется глоссарием и является описанием сущности данного элемента. [24]

Рассмотрим диаграммы бизнес-процессов протекающие на складе магазина ОАО ДММ, «Cleonelly»:

Для общей видимости системы необходимо построить контекст «Деятельность склада предприятия» (смотри рисунок 1.1).

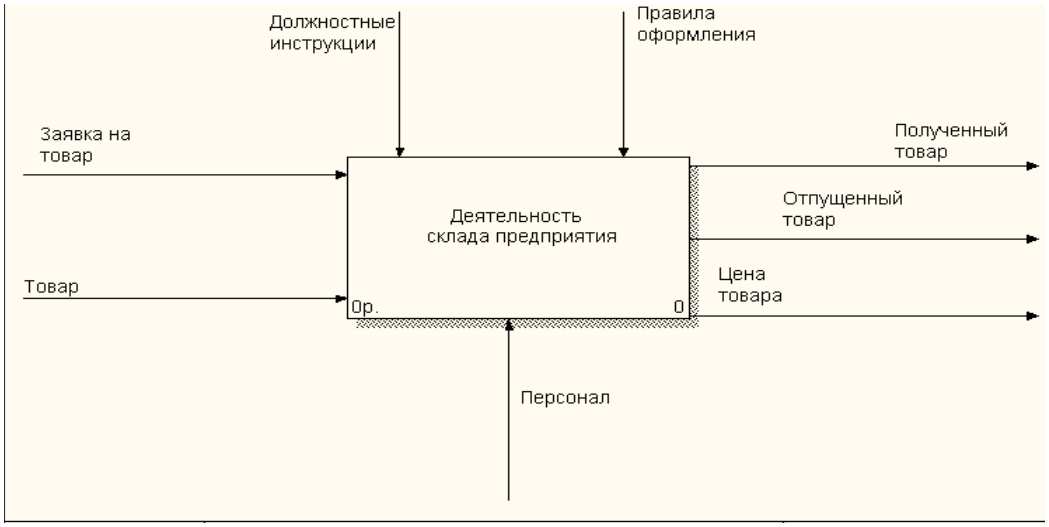


Рисунок 1.1 – Диаграмма «Деятельность склада предприятия»

После того как контекст установлен, проводится декомпозиция, т.е. построение следующих диаграмм в иерархии.

Каждая последующая диаграмма является более подробным описанием одной из работ на вышестоящей диаграмме. Пример декомпозиции контекстной работы показан на рисунке 1.2. Таким образом, вся система разбивается на подсистемы до нужного уровня детализации, данная система разбивается на три уровня.

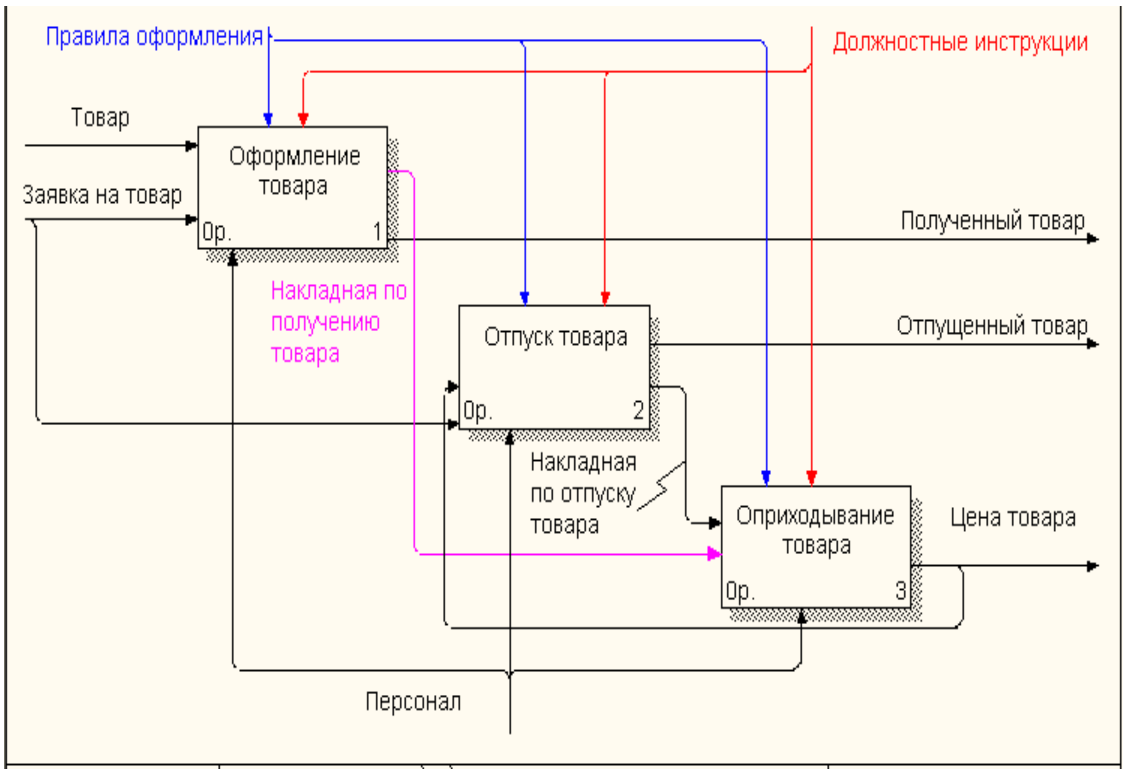


Рисунок 1.2 – Диаграммы декомпозиции первого уровня

Далее каждый из блоков декомпозиции системы будет еще разбиваться, декомпозиция «Оформление товара» можно увидеть на рисунке 1.3, «Отпуск товара» – рисунок 1.4, «Оприходывание товара» – рисунок 1.5.

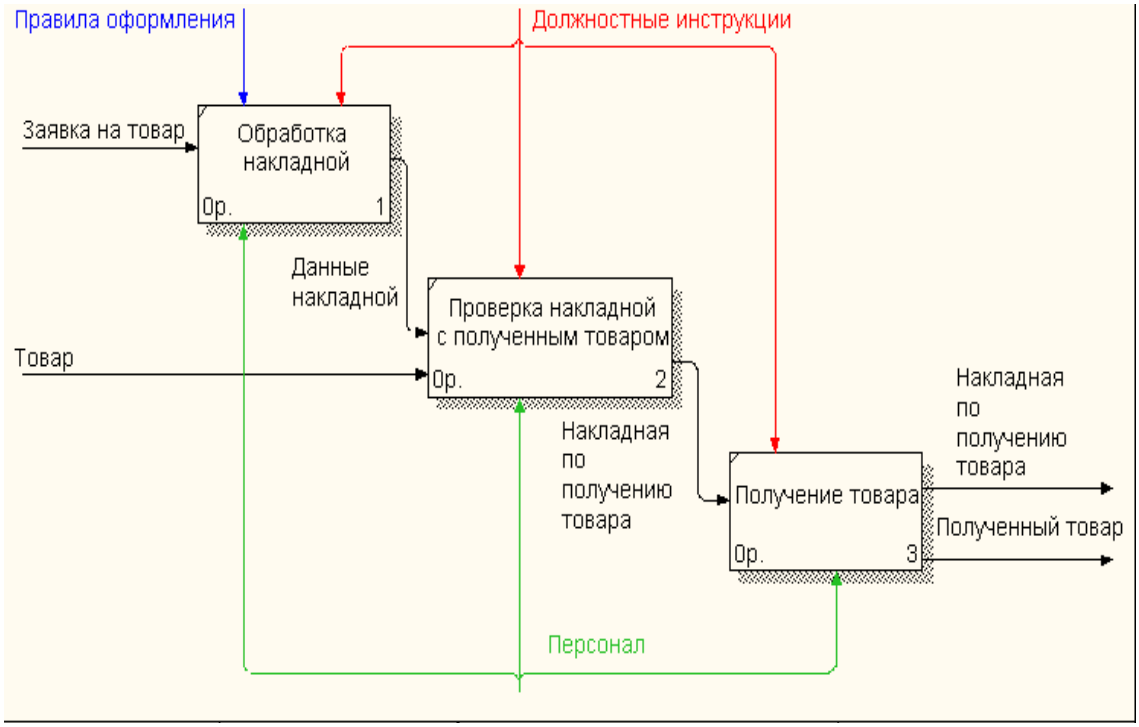


Рисунок 1.3 – Диаграмма «Оформление товара»

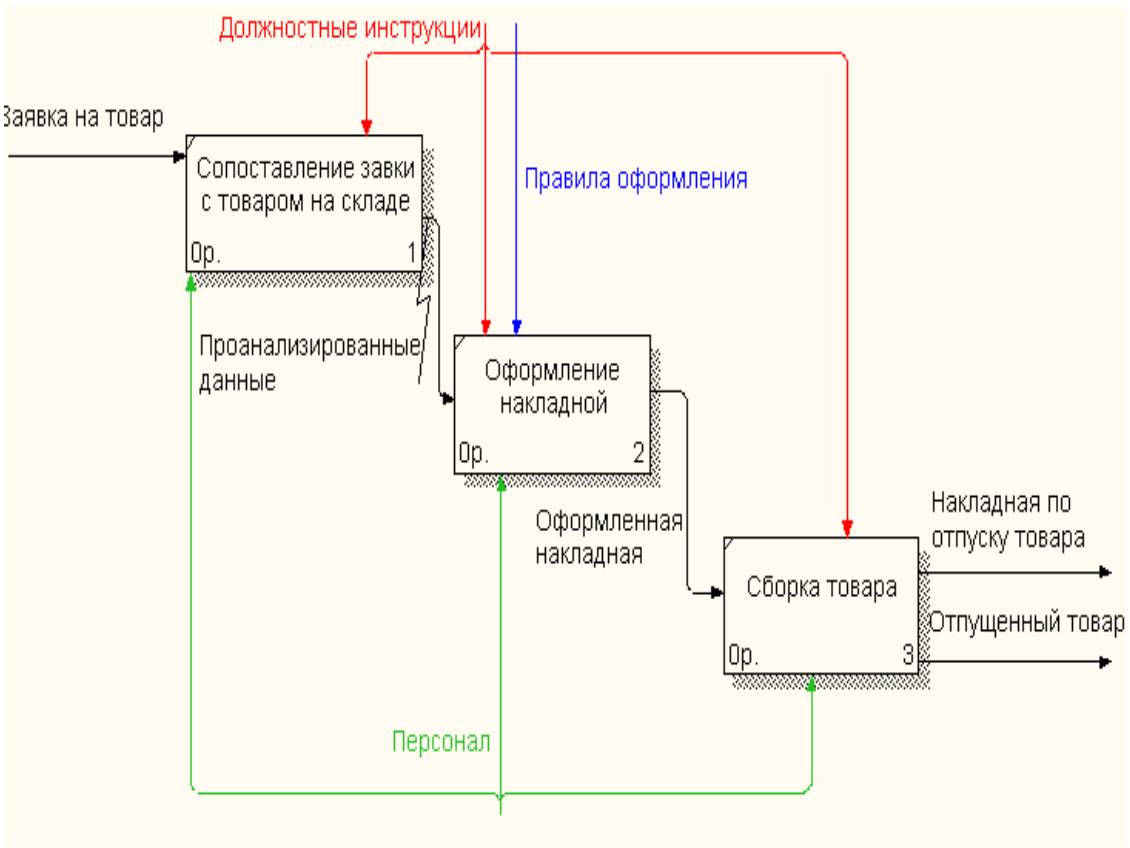


Рисунок 1.4 – Диаграмма «Отпуск товара»

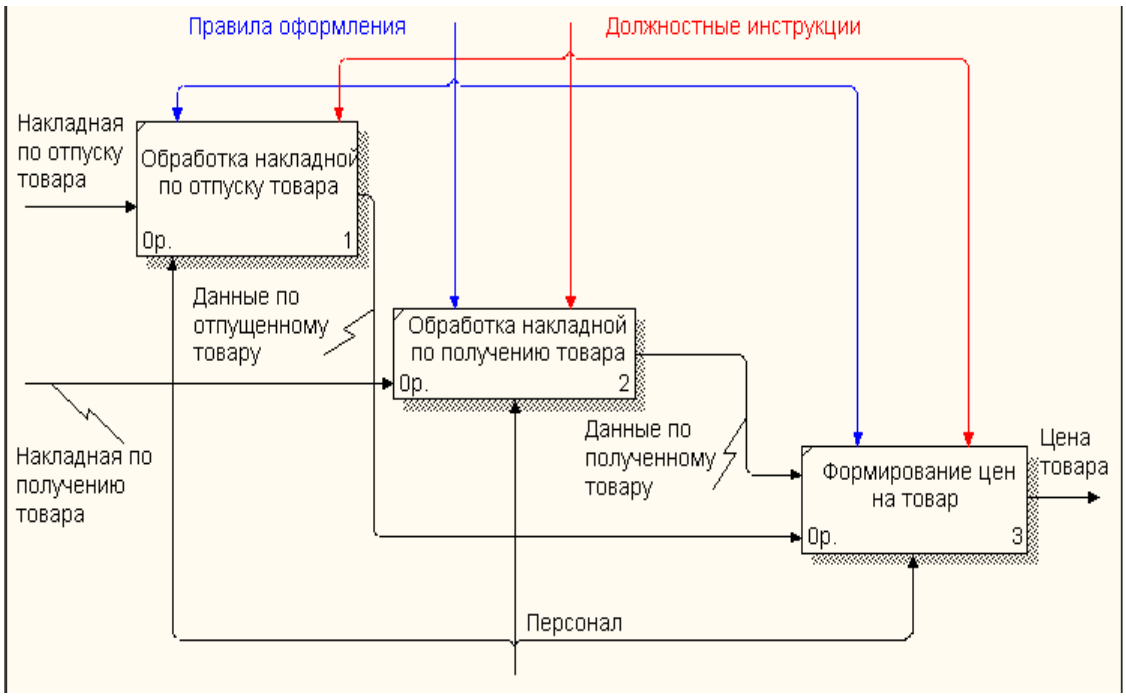


Рисунок 1.5 – Диаграмма «Оприходывание товара»

DFD. В основе данной методологии лежит построение модели анализируемой ИС – проектируемой или реально существующей. В соответствии с методологией модель системы определяется как иерархия диаграмм потоков данных (DFD), описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи пользователю. Диаграммы DFD обычно строятся для наглядного изображения текущей работы системы документооборота организации. Чаще всего диаграммы DFD используют в качестве дополнения модели бизнес-процессов, выполненной в IDEF0.

Основными компонентами диаграммы потоков данных являются:

- внешние сущности (графически изображены квадратом) – обозначают материальный предмет или физическое лицо, представляющее собой источник или приемник информации. Например: заказчики, персонал, поставщики, клиенты, склад;

- системы/подсистемы (графически выглядит как прямоугольник с округленными углами) – работы обозначающие функции или процессы, которые обрабатывают и изменяют информацию;

- накопители данных – представляют собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми. Накопитель данных в общем случае является прообразом будущей базы данных и описание хранящихся в нем данных должно быть увязано с информационной моделью;

- потоки данных – определяет информацию, передаваемую через некоторое соединение от источника к приемнику. Поток данных на диаграмме изображается линией, оканчивающейся стрелкой, которая показывает направление потока.

Рассмотрим диаграмму потоков данных (DFD) «Отпуск товара» рисунок 1.6. На этой диаграмме показано движение документов при поступлении в организацию «заявки на товар».

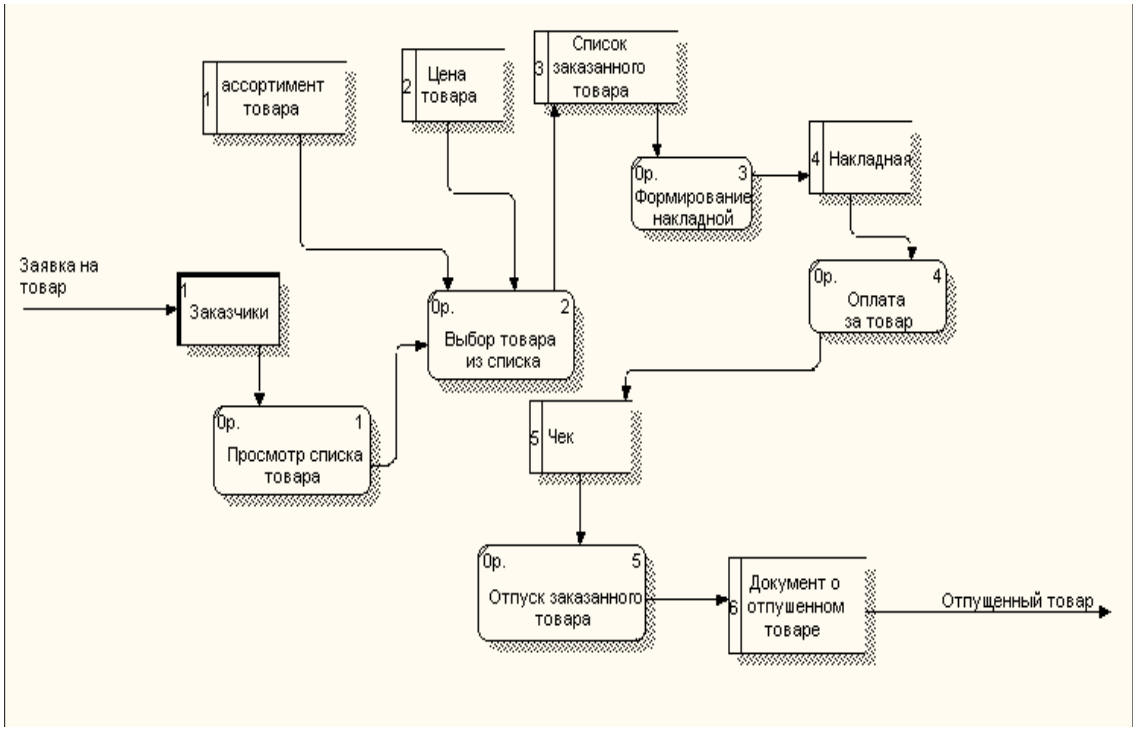


Рисунок 1.6 – Диаграмма DFD «Отпуск товара»

Рассмотрим следующую диаграмму потоков данных «Оформление товара» (смотри рисунок 1.7). Здесь показано процесс выполнения работ и

движение документов при «отпуске товара».

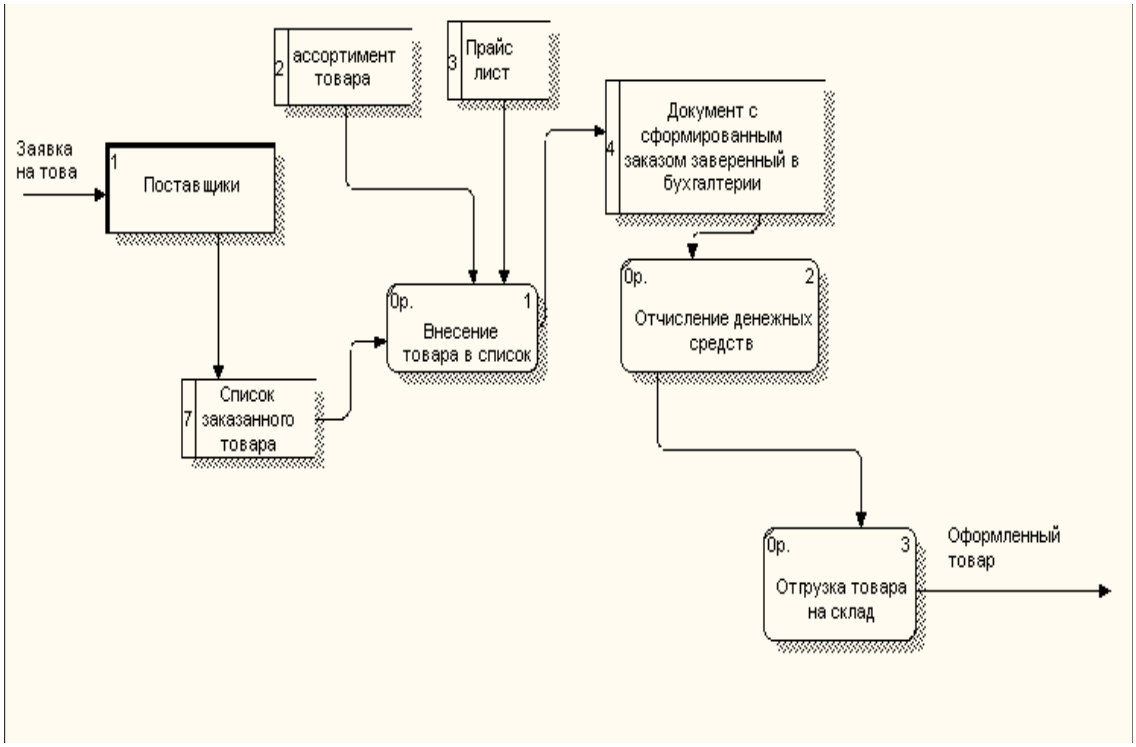


Рисунок 1.7 – Диаграмма DFD «Оформление товара»

В диаграммах потоков данных все используемые символы складываются в общую картину, которая дает четкое представление о том, какие данные используются, и какие функции выполняются системой документооборота. При этом часто выясняется, что существующие потоки информации, важные для деятельности компании, реализованы ненадежно и нуждаются в реорганизации.*******

Организационная структура предприятия, занимающегося продажей махровых изделий, рассмотрена на примере компании ОАО “Донецкая Мануфактура М” магазина Cleonelly:

В направлении разработки систем контроля и учета материалов могут успешно решать проблемы:

1. Это контроль за поставляемыми и хранящимися на складе товарами.

2. Информацию о поставщиках и потребителях

3. Также содержится информация информация и операции по товару

4. Содержится журнал отчета отпущенного товара

5. Содержится справочник товаров

6. Автоматизация складских функций (приход, расход, списание, резервирование товара)

7. Регистрация и хранение счетов на приобретаемый и продаваемый товар и за услуги, а также выписка счетов по предоплате, с отсрочкой платежа и с доставкой товара

8. Создание накладных и учет выданного товара

9. Проведение инвентаризации складов с созданием сличительной ведомости, акта недостачи и излишек

10. Создание комплектов товаров

Как указано основной сферой деятельности этого предприятия является продажа хлопчатобумажных изделий. Процесс проектирования включает множество этапов тщательно отрабатываемых управленческими структурами проектных предприятий в течение всего времени жизни данного предприятия. Данный процесс не может быть единовременно изменен, так как в нем задействовано множество подразделений самого предприятия, внешних субподрядчиков и клиентов проектного предприятия. Поэтому предприятия с осторожностью относятся к внедрению информационных систем, связанных с процессами управления проектирования и разработки. Как правило, российские предприятия используют собственные наработки в этой области.

1.3 Сбор требований

При проектировании информационной системы (ИС) «АРМ Оптового Магазина», было необходимо собрать требования, которые помогли бы создать интерфейс таким образом, что конечному пользователю (работнику

магазина) было удобно работать с разработанной ИС.

Разработка требований - это процесс, включающий мероприятия, необходимые для создания и утверждения документа, содержащего спецификацию системных требований.

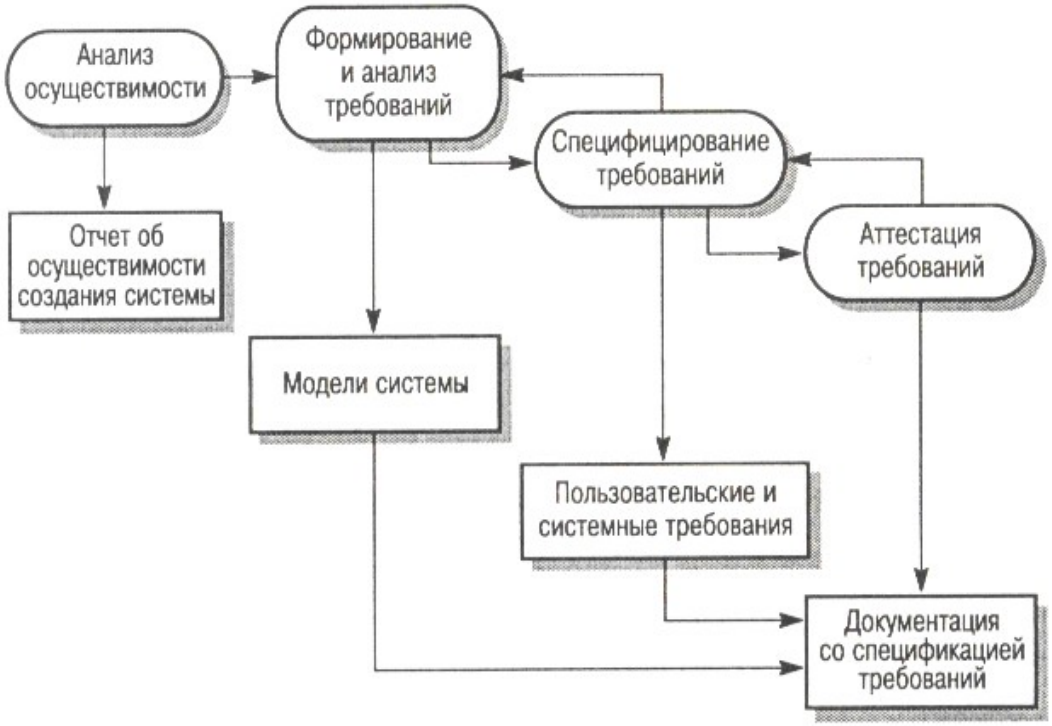
Для реализации процесса автоматизация учета и контроля материалов необходимо, чтобы информационная система могла выполнять следующие функциональные требования:

- документирование результатов.
- сохранить данные в базе;
- рассчитать количество материала на складе;
- Информационная система должна быть реализована как программа на базе интегрированной среды Visual Fox Pro.

Работа программы осуществляется в операционной системе Windows 2000/NT/XP.

Различают четыре основных этапа процесса разработки требований (рисунок 1.8):

- анализ технической осуществимости создания системы;
- формирование и анализ требований;
- специфицирование требований и создание соответствующей документации;
- аттестация требований.



Сбор требований является важным этапом проектирования ПО, поскольку именно здесь все требования заказчика должны быть правильно, и корректно сформулированы.

1.4 Спецификация требований

Определение корректных требований — это, наверное, самый ответственный этап программного проекта. Очень важно, чтобы формат проекта соответствовал требованиям к ПО, собранным командой разработчиков, в противном случае эти требования не смогут быть поддержаны и представлены в программном продукте. Спецификация требований к ПО - Software Requirements Specification(SRS) имеет ключевое значение для всего жизненного цикла разработки программного продукта. Это не только производный документ, в котором определены спецификации программного проекта, но и основной документ, применяемый с целью проведения аттестационных и приемочных испытаний. Аттестация — это оценка качества работы менеджеров проекта. Она определяет степень

соответствия программного продукта установленным требованиям. Спецификация SRS выступает в роли некоего механизма фиксирования системных требований, которые используются в качестве критериев при аттестации .

На основании SRS достигается согласие между заказчиками и производителями программного продукта. В спецификации SRS полностью описаны функции, которые должен выполнять разрабатываемый программный продукт. Это позволяет потенциальным пользователям определить степень соответствия продукта их потребностям, а также пути модификации продукта для того, чтобы он был максимально полезен в решении их задач.

Снижаются временные затраты на разработку. В подготовке спецификации SRS задействованы различные группы в организации заказчика. Они тщательно исследуют все требования еще до того, как начнется непосредственная разработка проекта. Это снижает вероятность последующей повторной разработки проекта, кодирования и тестирования.

При тщательном изучении требований, представленных в спецификации SRS, можно обнаружить недосмотры, недоразумения и противоречия еще на ранних этапах цикла разработки, когда проблемы устранять гораздо легче, чем на более поздних этапах.

Спецификация SRS становится основой для оценки стоимости и составления графика работ. Описание продукта — это реальный базис для оценки стоимости проекта. В среде, где существует понятие формального предложения, SRS используют для утверждения оценки предложения или цены.

С помощью правильно составленных спецификаций SRS на уровне организации могут разрабатывать намного более продуктивные планы аттестаций и проверок. Являясь частью договора на разработку, SRS обеспечивает точку отсчета для оценки соответствия техническим условиям.

Благодаря спецификации SRS облегчается передача программного

продукта новым пользователям, а также его установка на других компьютерах. Таким образом, заказчикам становится проще переносить программный продукт в другие подразделения организации, а разработчикам — передавать другим заказчикам.

Спецификация SRS служит основой для модернизации. В этом документе рассматривается сам продукт, а не процесс разработки проекта, поэтому на ее основании можно производить расширение завершенного продукта.

После того как процесс определения и спецификации требований завершён, необходимо осуществить аттестацию требований.

Спецификация требований к программному проекту должна быть представлена в приложении А.

1.5 Аттестация требований

Аттестация должна продемонстрировать, что требования действительно определяют ту систему, которую хочет иметь заказчик. Проверка требований важна, так как ошибка в спецификации требований могут привести к переделке системы и большим затратам, если будут обнаружены во время процесса разработки системы или после введения её в эксплуатацию.

Во время процесса аттестации требований должны быть выполнены различные типы проверок документации требований:

1. Проверка правильности требований.
2. Проверка на непротиворечивость.
3. Проверка на полноту.
4. Проверка на выполнимость.

Существует ряд методов аттестации требований, которые можно использовать совместно или каждый в отдельности:

1. Обзор требований.

2. Прототипирование.

3. Генерация тестовых сценариев.

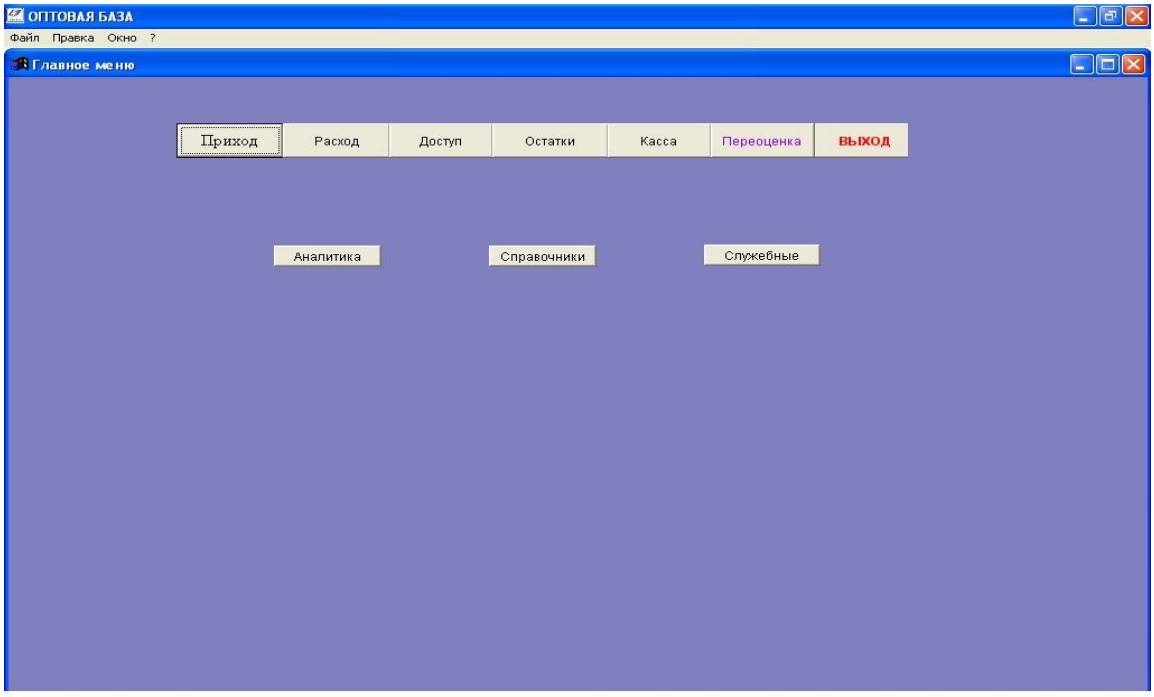
4. Автоматизированный анализ непротиворечивости.

Наиболее наглядным для заказчика системы является прототипирование.

Перед началом создания прототипов можно создать диаграмму потоков пользовательского интерфейса. Такая диаграмма используется для изучения взаимосвязей между основными элементами пользовательского интерфейса.

Следующим шагом аттестации требований является непосредственное создание прототипов.

Прототип ПО – это частичная или возможная реализация предлагаемого нового продукта. Прототипы позволяют решать три основные задачи: прояснение и завершение процесса формулировки требований, исследование альтернативных решений и создание конечного продукта.



Прототип основного меню данного модуля представлен на рисунке 1.9.

1.6 Выбор методологии проектирования информационной системы

Сущность структурного подхода к разработке ИС заключается в ее декомпозиции (разбиении) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. Пи этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны.

Все наиболее распространенные методологии структурного подхода базируются на ряде общих принципов. В качестве двух базовых принципов используются следующие принципы:

- принцип "разделяй и властвуй" – принцип решения сложных проблем путем их разбиения на множество меньших независимых задач, легких для понимания и решения;
- принцип иерархического упорядочивания - принцип организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.

В структурном анализе используются в основном две группы средств, иллюстрирующих функции, выполняемые системой и отношения между данными. Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными, среди которых являются следующие:

- SADT (Structured Analysis and Design Technique) модели и соответствующие функциональные диаграммы;
- DFD (Data Flow Diagrams) диаграммы потоков данных;
- ERD (Entity-Relationship Diagrams) диаграммы "сущность-связь".

На стадии проектирования ИС модели расширяются, уточняются и дополняются диаграммами, отражающими структуру программного

обеспечения: архитектуру ПО, структурные схемы программ и диаграммы экранных форм.

Перечисленные модели в совокупности дают полное описание ИС независимо от того, является ли она существующей или вновь разрабатываемой. Состав диаграмм в каждом конкретном случае зависит от необходимой полноты описания системы. [35]

2 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Архитектурное проектирование

При создании любой сложной информационной системы критическим аспектом является ее архитектура, где она представляет собой концептуальное видение структуры будущих функциональных процессов и технологий на системном уровне и во взаимосвязи. Обычно сложные информационные системы организаций проектируются как композиция компонентов, взаимодействующих на высоком уровне, которые сами могут быть системами. Архитектура информационной системы организации делает понимание системы легче, определяя ее функциональность и структуру способом, который раскрывает проектировочные решения и позволяет обозревателю задавать вопросы об удовлетворении проектных требований, распределении функциональности и реализации компонентов.

Архитектура информационной системы организации представляет собой модель того, как информационная технология будет поддерживать основные цели и стратегию развития автоматизируемого объекта. Она позволяет критически мыслить и ясно выразить представление того, как интегрированные наборы информационных систем должны быть структурированы для реализации этих целей. Архитектура информационной системы описывает, как информационные системы, приложения и люди работают в пределах всей организации в единообразной объединенной манере.

Таким образом, архитектура информационной системы включает в себя общепринятый набор компонентов, которые обеспечивают «строительные блоки» информационной системы. Эти «строительные блоки» и их характеристики определены на соответствующем уровне деталировки для соответствия потребностям, производимым планировочным решениям.

При проектировании современных информационных систем организаций их архитектура должна разрабатываться с учетом многих заинтересованных сторон, она должна быть понятной пользователям, дать возможность разработчикам сделать план и графики системы, позволять определять ключевые интерфейсы, функции и технологии, а также позволять оценить график и бюджет исполнения проекта. При этом от архитекторов современных информационных систем требуется ответственность за создание удовлетворительной и осуществимой концепции системы на самом раннем этапе ее разработки, поддержки цельности этой концепции на протяжении разработки и определения пригодности результирующей системы для использования клиентом. С другой стороны, разработка архитектуры информационной системы – это процесс описания архитектур информационных систем в достаточной деталировке, чтобы сделать их более полезными для разработки информационных систем.

Изучение зарубежного опыта показывает, что в развитых странах при разработке архитектуры информационной системы требуется соблюдение следующих условий:

- направленность на миссии организации;
- направленность на требованиях;
- направленность на разработке;
- возможность к адаптации;
- необходимость гибкости.

Соблюдение всех этих условий позволяет разработать архитектуру информационной системы организации более совершенной и эффективной.

Основными программными архитектурами, реализуемыми в настоящее

время являются:

- файл-серверная;
- клиент-серверная;
- многоуровневая.

Файл-сервер. Эта архитектура централизованных баз данных с сетевым доступом предполагает назначение одного из компьютеров сети в качестве выделенного сервера, на котором будут храниться файлы централизованной базы данных. В соответствии с запросами пользователей файлы с файл-сервера передаются на рабочие станции пользователей, где и осуществляется основная часть обработки данных. Центральный сервер выполняет в основном только роль хранилища файлов, не участвуя в обработке самих данных. После завершения работы пользователи копируют файлы с обработанными данными обратно на сервер, откуда их могут взять и обработать другие пользователи. Такая организация ведения данных обладает рядом недостатков, например, при одновременном обращении множества пользователей к одним и тем же данным производительность работы резко падает, так как необходимо дождаться, пока пользователь, работающий с данными, завершит работу. В противном случае возможно затирание исправлений, сделанных одними пользователями, изменениями, внесенными другими пользователями.

Клиент-сервер. В основе этой концепции лежит идея о том, что помимо хранения файлов базы данных, центральный сервер должен выполнять основную часть обработки данных. Пользователи обращаются к центральному серверу с помощью специального языка структурированных запросов (SQL, Structured Query Language), на котором описывается список задач, выполняемых сервером. Запросы пользователей принимаются сервером и порождают в нем процессы обработки данных. В ответ пользователь получает уже обработанный набор данных. Между клиентом и сервером передается не весь набор данных, как это происходит в технологии файл-сервер, а только данные, которые необходимы клиенту. Запрос

пользователя длиной всего в несколько строк способен породить процесс обработки данных, затрагивающий множество таблиц и миллионы строк. В ответ клиент может получить лишь несколько чисел. Технология клиент-сервер позволяет избежать передачи по сети огромных объемов информации, переложив всю обработку данных на центральный сервер. Кроме того, рассматриваемый подход позволяет избежать конфликтов изменений одних и тех же данных множеством пользователей, которые характерны для технологии файл-сервер. Технология клиент-сервер реализует согласованное изменение данных множеством клиентов, обеспечивая автоматическое соблюдение целостности данных. Эти и некоторые другие преимущества сделали технологию клиент-сервер очень популярной. К недостаткам этой технологии можно отнести высокие требования к производительности центрального сервера. Чем больше клиентов обращается к серверу, и чем больше объем обрабатываемых данных, тем более мощным должен быть центральный сервер.

Исходя из этих рассуждений при проектировании архитектуры АРМ за основу была принята технология клиент-сервер. Диаграммы размещения отражают физические взаимосвязи между программными и аппаратными компонентами системы).

2.2 Проектирование интерфейса информационной системы

Под пользовательским интерфейсом часто понимают только внешний вид программы. Однако на деле пользователь воспринимает через него всю систему в целом, а значит, такое его понимание является слишком узким. В действительности пользовательский интерфейс включает в себя все аспекты дизайна, которые оказывают влияние на взаимодействие пользователя и системы. Это не только экран, который видит пользователь. Пользовательский интерфейс состоит из множества составляющих, таких как:

- набор задач пользователя, которые он решает при помощи системы;
- элементы управления системой;
- навигация между блоками системы;
- визуальный дизайн экранов программы.

Выделим несколько наиболее существенных преимуществ хорошего пользовательского интерфейса с точки зрения бизнеса:

- снижение количества ошибок пользователя;
- снижение стоимости поддержки системы;
- уменьшение потерь продуктивности работников при внедрении системы и более быстрое восстановление утраченной продуктивности;
- улучшение морального состояния персонала;
- уменьшение расходов на изменение пользовательского интерфейса по требованию пользователей;
- доступность функциональности системы для максимального количества пользователей.

АРМ оптовая база разрабатывается как приложение использующее технологию клиент-сервер.

2.2.1 Пользовательский интерфейс управляющей программы

Основным модулем «АРМ Оптовая База» является модуль Luck.exe, обеспечивающий реализацию основной функциональности диаграммы вариантов использования, представленной на рисунке 1.9 раздела 1.4.

При разработке информационной системой одной из главных задач, является создание наиболее простого и не загруженного интерфейса. Именно интерфейс программного продукта, помогает пользователям «общаться» с информационной системой, выступая как диалог общения пользователя и системой.

Интерфейс программы, администраторская часть:

1. стартовая форма программы. Данная форма запускается при запуске

программного продукта образуя, таким образом, начало диалога пользователя с системой (рисунок 2.3);

2. форма администратора. В этой форме осуществляется полное управление информационной системой, т.е. добавление, удаление, изменение данных в базе данных, а также при необходимости просмотр и печать отчетов (рисунок 2.4);

3. форма «Заказчики», благодаря этой форме можно видеть полную информацию о заказчиках предприятия (рисунок 2.7);

4. форма «Поставщики», благодаря этой форме можно видеть полную информацию о заказчиках предприятия (рисунок 2.8).

Интерфейс программы пользовательская часть:

В окне приход товара идет оформление товара. При выборе данной вкладке формы, пользователь сначала должен

В меню расход там тут происходят операции проводимые сотрудника склада по отпуску и продаже товара.

В меню остатки происходит подсчет товара, наименования хранящегося на складе.

В меню касса тут хранятся информация по приходным ордерам и расходным кассовым ордерам.(скриншоты)

2.2.2 Пользовательские интерфейсы компонентов управления

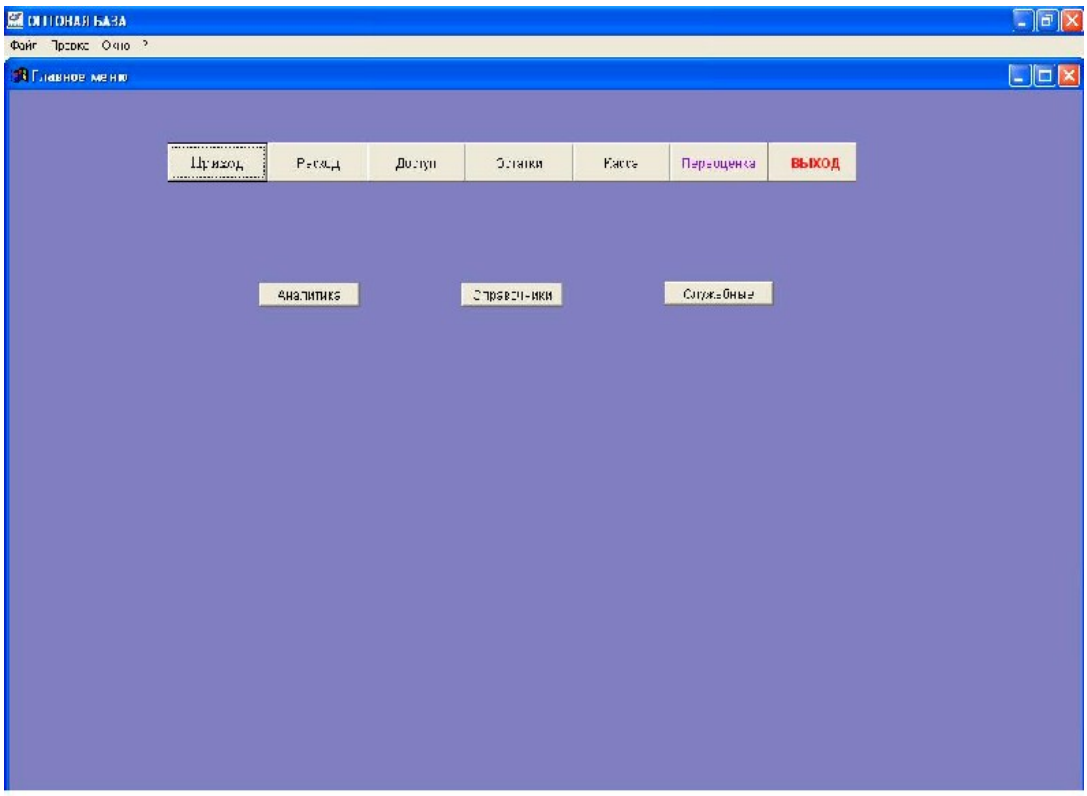


Рис 2.0 Главное меню программы

Главное окно программы показано на рис. 1.9. Как видно из рисунка, кроме главного меню, уже описанного выше, оно также будет содержать панель управления (кнопки «Приход», «Расход», «Доступ», «Остатки», «Касса»,«Переоценка»,«Аналитика»,«Справочники»,«Служебные» и «Выход из программы»).

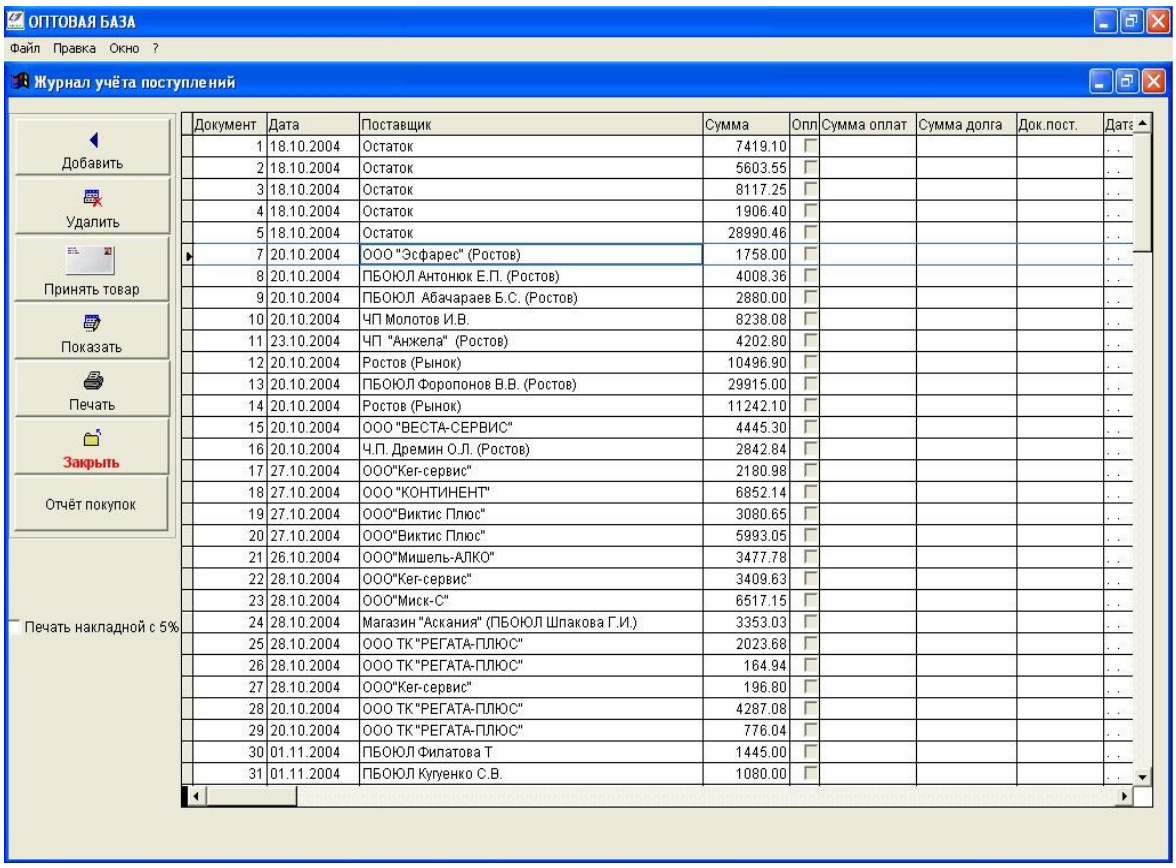


Рисунок 2.1 Окно меню прихода или поступления на склад .

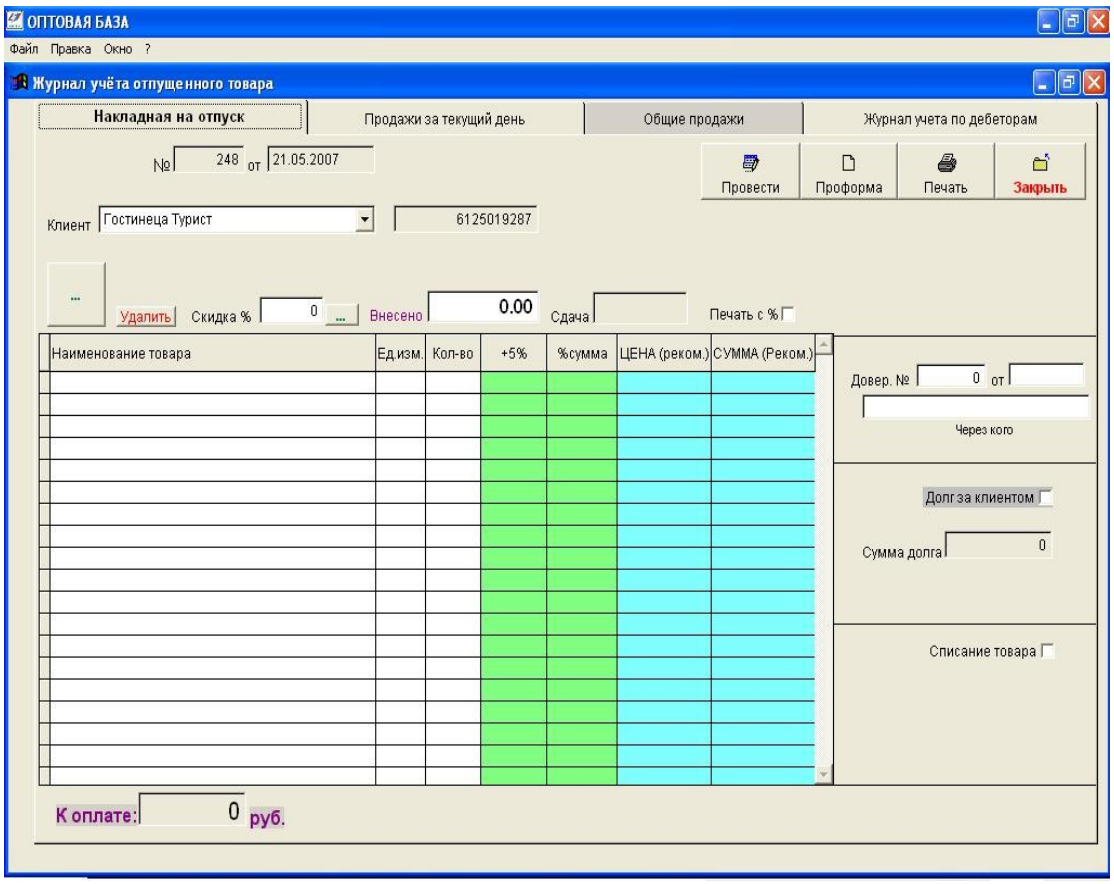


Рисунок 2.2 Окно меню расхода

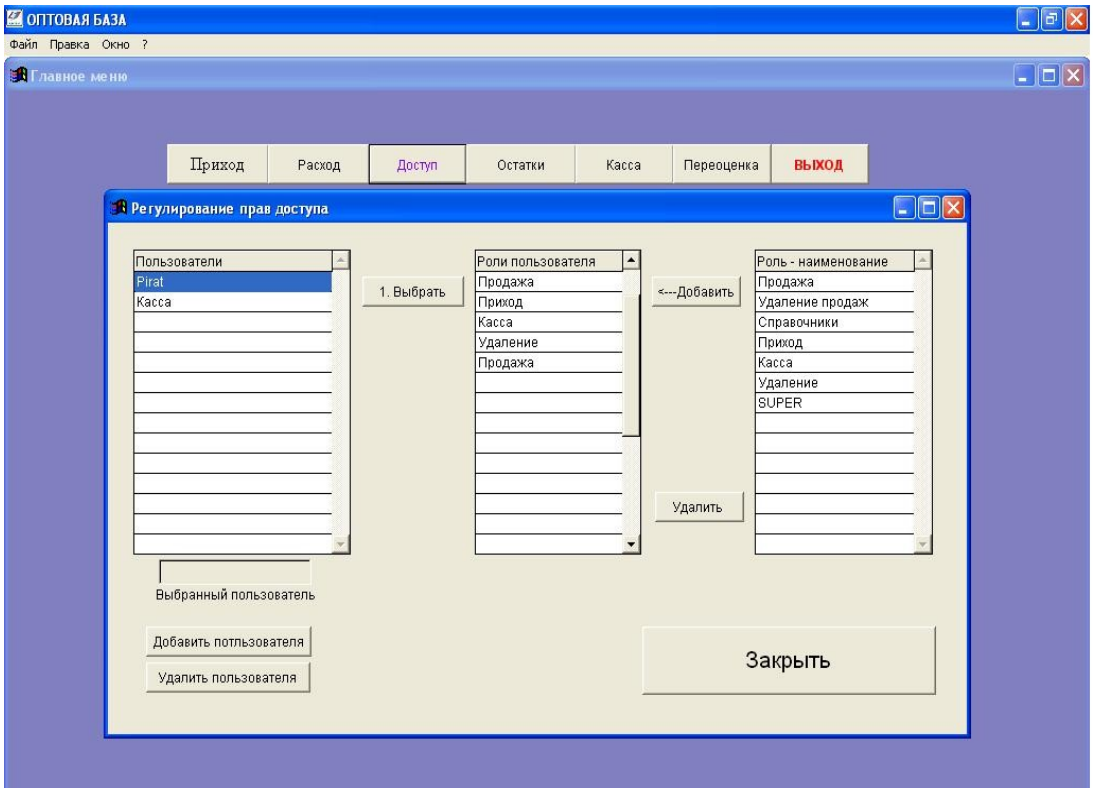


Рисунок 2.2 Окно меню регулирующее права доступа к программе.

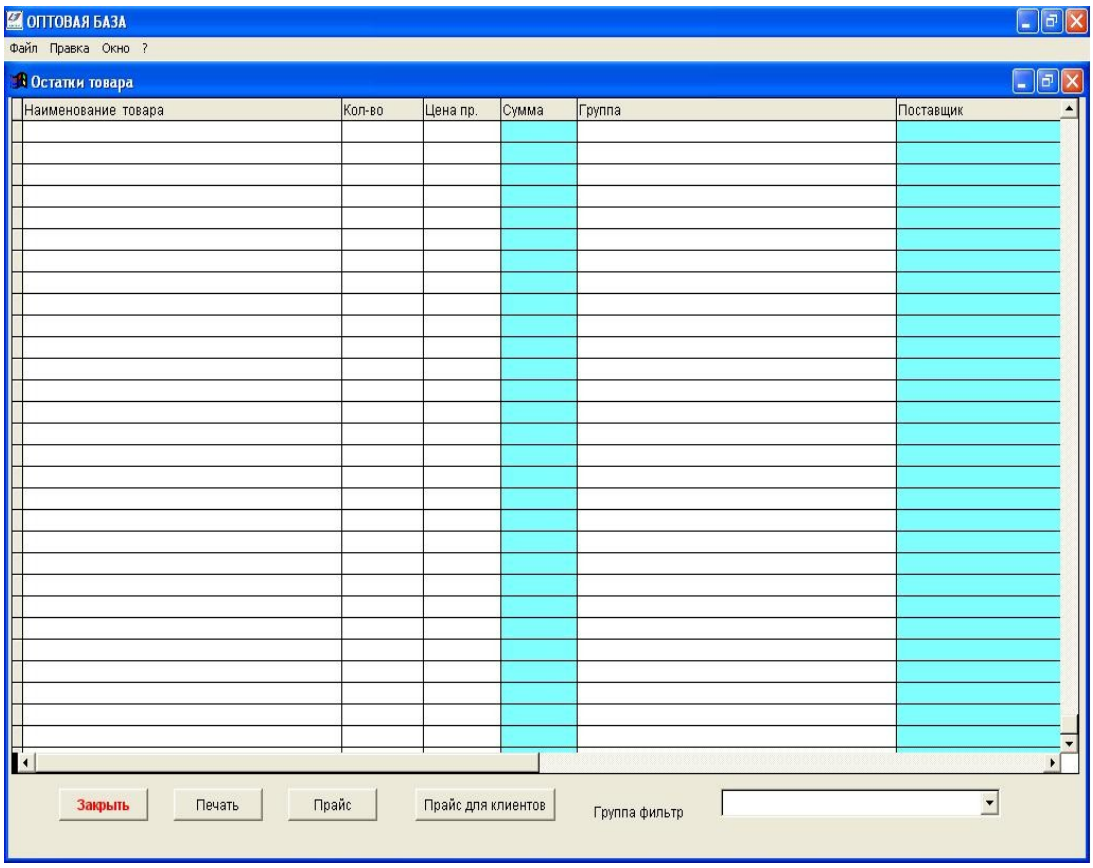


Рисунок 2.3 Окно меню остатка товара.

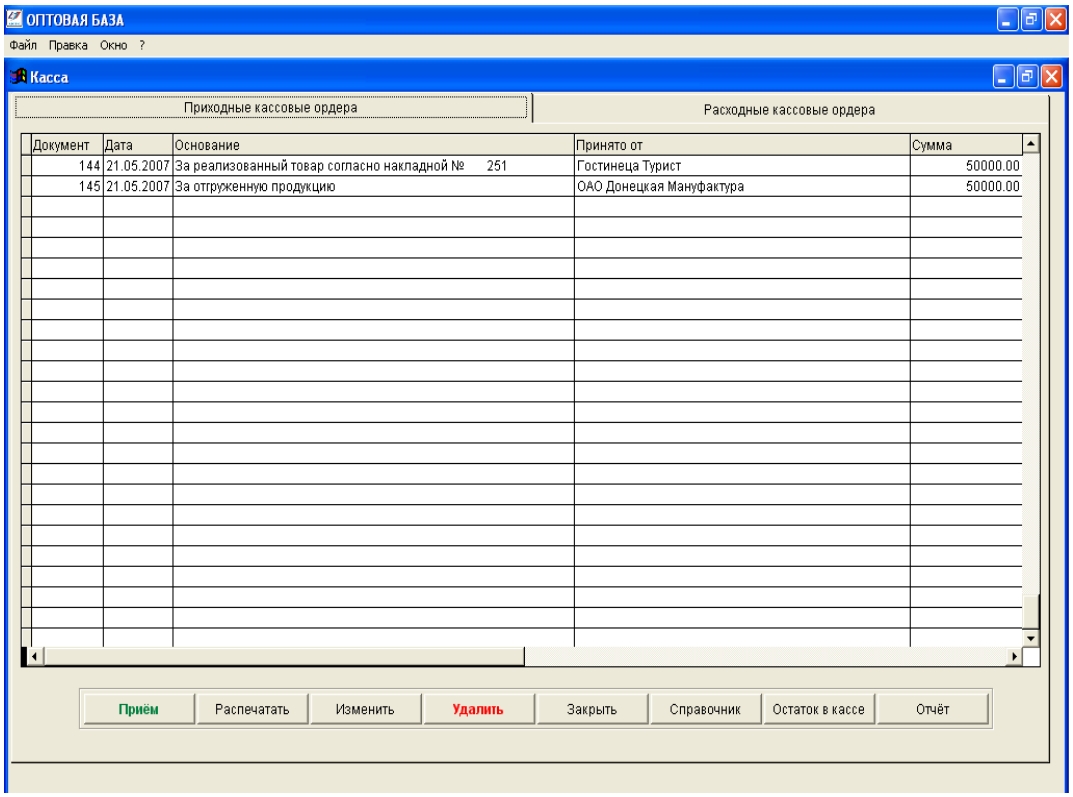


Рисунок 2.4 Окно меню касса.

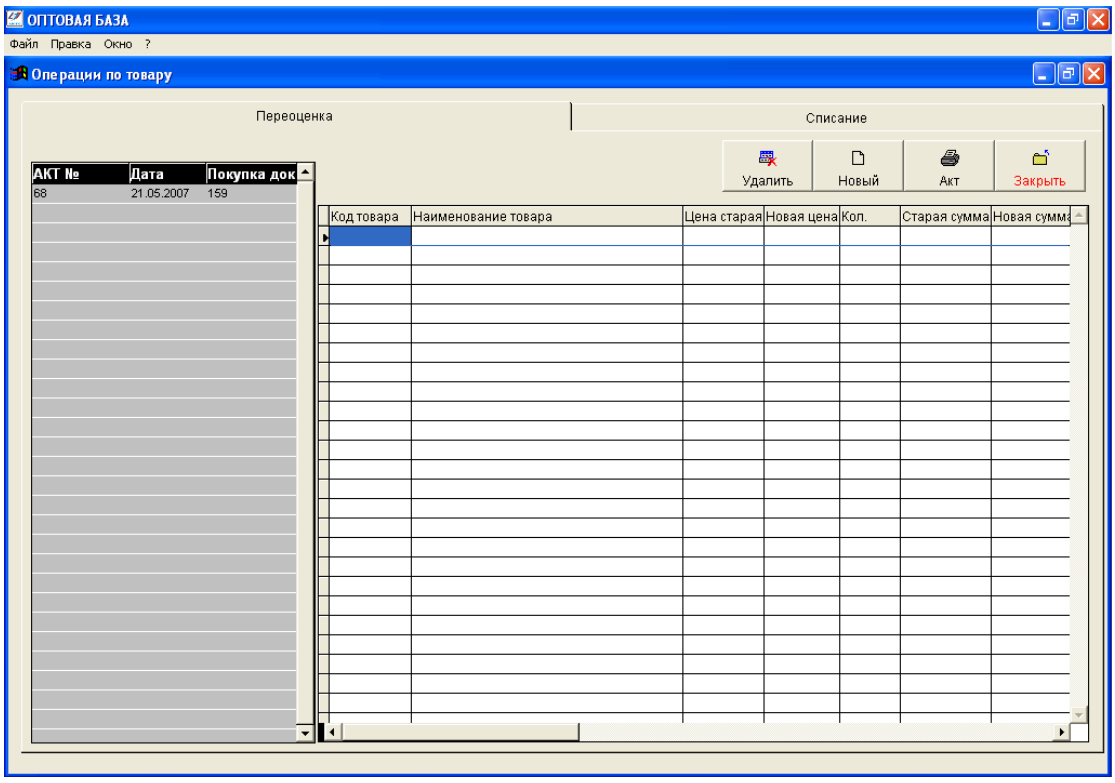


Рисунок 2.4 Окно меню переоценка.

2.3 Проектирование баз данных

Для проектирования базы данных был использован ERwin 4.0 от Computer Associates Int.

ERwin - мощное и простое в использовании средство конструирования баз данных завоевавшее широкое признание и популярность. Оно обеспечивает высочайшую продуктивность труда при разработке и сопровождении приложений с использованием баз данных. На протяжении всего процесса - от логического моделирования требований к информации и бизнес-правил, которые определяют базу данных, до оптимизации физической модели в соответствии с заданными характеристиками - ERwin позволяет наглядно отобразить структуру и основные элементы БД. [20]

ERwin - не только лучший инструмент для проектирования баз данных, но и средство для их быстрого создания. ERwin оптимизирует модель в соответствии с физическими характеристиками целевой базы данных. В отличие от других инструментальных средств, ERwin автоматически

поддерживает согласованность логической и физической схем и осуществляет преобразование логических конструкций, таких как отношения многие-ко-многим, в их реализацию на физическом уровне. Облегчает проектирование баз данных. Для этого достаточно создать графическую E-R модель (объект-отношение), удовлетворяющую всем требованиям к данным и ввести бизнес-правила для создания логической модели, которая отображает все элементы, атрибуты, отношения и группировки. Erwin имеет два уровня представления модели – логический и физический. Логический уровень – это абстрактный взгляд на данные, на нём данные представляются, так как выглядят в реальном мире, и могут называться так, как называются в реальном мире, например «Постоянный клиент», «Отдел» или «Фамилия сотрудника». Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами. Логический уровень модели данных является универсальным и никак не связан с конкретной реализацией СУБД. Различают три подуровня логического уровня модели данных, отличающиеся по глубине представления информации о данных:

- Диаграмма сущность – связь (Entity Relationships Diagram (ERD));
- Модель данных, основанная на ключах (Key Based model (KB));

Полная атрибутивная модель (Fully Attributed model (FA)).

Диаграмма сущность – связь включает сущности и взаимосвязи, отражающие основные бизнес – правила предметной области. Такая диаграмма не слишком детализирована, в неё включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям. Диаграмма сущность – связь может включать связи «многие ко многим» и не включать описание ключей. Как правило, ERD используется для презентаций и обсуждения структуры данных с экспертами предметной области. Модель данных, основанная на ключах, - более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

Логическая модель – наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи (смотри приложение Б). [26]

Физическая модель данных напротив зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физическом уровне модели содержится информация обо всех объектах базы данных. Поскольку стандартов на объекты базы данных не существует (например, нет стандарта на типы данных), физический уровень модели зависит от конкретной реализации СУБД. Следовательно, одному и тому же логическому уровню модели могут соответствовать несколько разных физических уровней различных моделей. Если на логическом уровне модели не имеет большего значения, какой конкретно тип данных у атрибута (хотя и поддерживаются абстрактные типы данных), то на физическом уровне модели важно описать всю информацию о конкретных физических объектах – таблицах, колонках, индексах, процедурах и т.д. Разделение модели данных на логический и физический уровни позволяет решить несколько важных задач. [20]

Физическая модель данных представлена в приложении В.

2.4 Обоснование выбора платформы создания информационной системы

Visual FoxPro — визуальная среда разработки систем управления реляционными базами данных, выпускаемая в настоящее время корпорацией Майкрософт. Последней версией является 9.0. Использует язык программирования FoxPro. Версия системы 7.0 может работать в операционных системах Windows 9x и ядра NT, версии 8.0 и 9.0 - только в Windows XP, 2000, 2003.

FoxPro (Фокс-про?) — один из диалектов языка программирования xBase. Применяется в основном для разработки реляционных СУБД, хотя

возможно применять для разработки и других классов программ.Как уже отмечалось выше, язык VFP это сильно дополненный и расширенный язык xBase. В Visual FoxPro язык программирования, то есть базовой конструкцией языка является понятие класса. Исходный же вариант xBase это чистейший структурный язык, с базовым понятием процедур и функций. Таким образом, современный язык программирования Visual FoxPro допускает совмещать как и программирование "по старинке" описанием массы процедур, так и в стиле ООП, создавая сложную иерархию классов.

Выбрал я этот язык программирования потому что он содержит ряд следующих преимуществ :

– Широко известный формат таблиц баз данных, что позволяет легко организовать обмен информацией с другими приложениями Microsoft Windows.

- Современная организация реляционных баз данных, позволяющая хранить информацию о таблицах базы, их свойствах, индексах и связях, задавать условия соблюдения ссылочной целостности, создавать локальные и удаленные представления (Views), связи с серверами, хранимые процедуры, исполняемые при наступлении более 50 различных видов событий (VFP 7.0-9.0).
- Высокая скорость работы с большими базами данных.
- Высокая наглядность работы с базами данных: многофункциональное окно Data session позволяет видеть список открытых таблиц баз данных, их связи, фильтры, порядок по индексам, режимы буферизации, переходить к режимам модификации структуры, к работе с информацией таблиц и пр.
- Высокая скорость разработки приложений с использованием Мастеров (Wizard), Конструкторов (Designer), Построителей (Builder), режим подсказок IntelliSense при написании текста программ, системы отладки и тестирования программ.
- Возможность разработки приложений, работающих по технологии

"клиент-сервер" с данными, размещенными на серверах баз данных Oracle и Microsoft SQL Server и с другими приложениями Microsoft Windows с использованием ODBC и OLE

- Система VFP предназначена для использования профессиональными программистами, поэтому нет смысла в русификации ее меню и языка - для любого программиста английский синтаксис алгоритмического языка более привычен, чем русский.

2.5 Проектирование модулей

Остановимся подробней на проектировании одного из модулей программы и рассмотрим на его примере шаги необходимые для создания проекта.

В качестве примера мной будет рассмотрено проектирование модуля, реализующего вариант использования «Оформляет заявку на поступление».

Для начала опишем потоки событий, происходящие в данном варианте использования.

Предусловием к варианту использования является поступление заявки от клиента.

5. Вариант использования начинается, когда клиент присылает заявку.
6. Менеджер открывает форму Приход.
7. Менеджер ставит дату заявки.
8. Менеджер ставит наименование товара.
9. Менеджер вносит количество поступаемого товара.
10. Менеджер вносит сумму заявки.
11. Менеджер закрывает форму.
12. Вариант использования заканчивается.

Постусловием к варианту использования является оформление заявки в системе и появление нового клиента в журнале главной формы.

Рассмотрим диаграмму последовательности данного варианта использования. Как видно из этой диаграммы менеджер, открывая форму Приход, вызывает выполнение нескольких действий – автоматически (с точки зрения менеджера) заполняется дата заявки. Список клиентов при оформлении заявки заполняется из базы с первичной информацией. После этого менеджер вносит все необходимые данные и нажимает кнопку «Принять». При этом выполняются следующие действия. Все данные передаются в хранимую процедуру.

3 РЕАЛИЗАЦИЯ И АТТЕСТАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1 Реализация приложения

Реализация приложения по своей сути, является одним из трудоемких этапов для разработчика информационной системы, потому что, те требования, которые выдвигает заказчик, должны быть четко и корректно интегрированы в систему. Пока нет таких программных продуктов, которые могли бы «подстраиваться» под требования так называемого заказчика и выдавать определенный набор функций для реализации системы, которые будут соответствовать этим требованиям. Поэтому каждый разработчик должен выбрать для себя оптимальную среду для разработки системы, но следует заметить, что при реализации приложения никак не обойтись без написания программного кода. Именно при написании программного кода, будут реализовываться некие функции, которые должна выполнять система. В зависимости от выбранной среды реализации системы, программный код будет выглядеть по-разному, в такой среде как Microsoft Visual FoxPro будет один программный код, в Visual Basic другой и т.д.

В данном случае реализация приложения выполнялась в Microsoft Visual FoxPro.

Ниже будут описаны основные функции системы:

1. Стартовая форма системы. Данная форма является кнопочной формой и соответственно каждая кнопка выполняет свою функцию. Кнопка регистрация администратора изображена на рисунке 3.1.Данная кнопка выполнят функцию, которая открывает панель администратора, если пользователь имеет такие права к данной системе

2. Кнопка меню приход. Даная кнопка позволяет провести учет поступающих товаров на склад магазина рис 3.2.

3. В кнопке меню расход ведется учет отпущенного товара со склада рис 3.3.

4. В кнопке меню доступа регулируются права пользования данной программой рис 3.4.

5. В кнопке меню остатки хранится информация о хранящихся материалах на складе магазина рис 3.5.

6. В кнопке меню касса храниться информация о приходных кассовых ордерах и расходных кассовых ордерах рис 3.6.

7. В кнопке меню переоценка проходит изменения цены на новую цену товара рис.3.7.

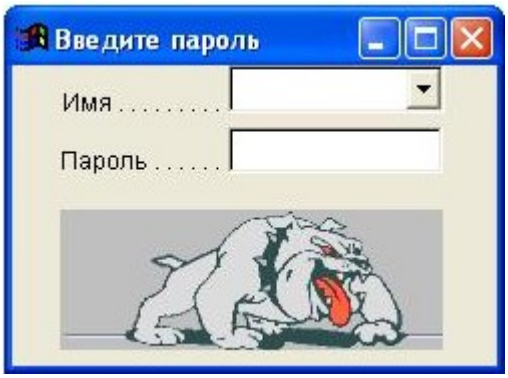


Рисунок 3.1 – Стартовая форма системы

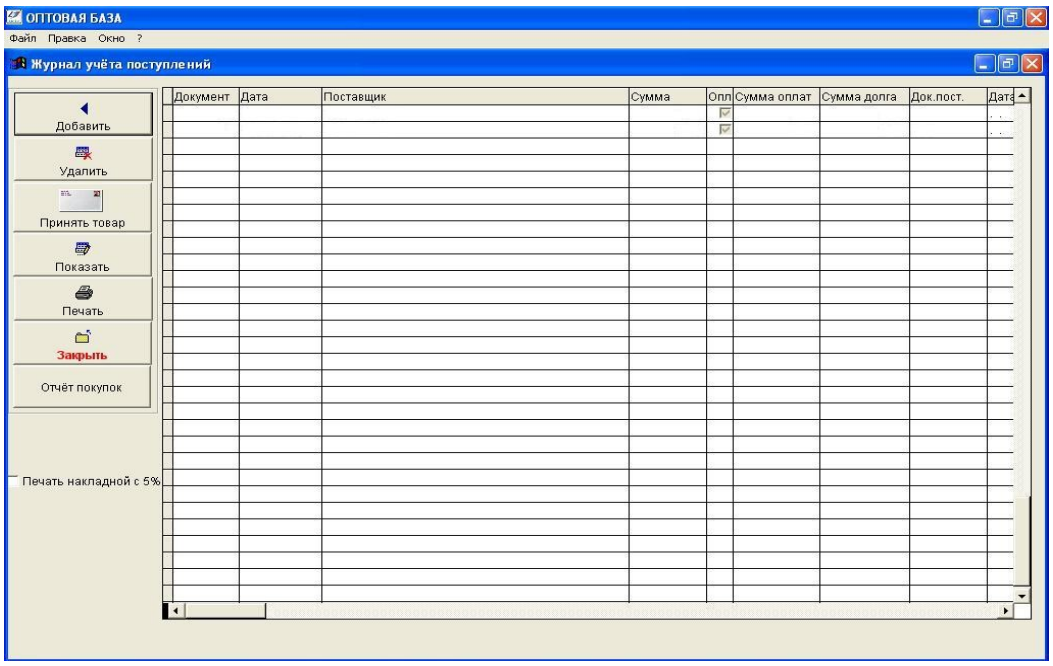


Рисунок 3.2 – Форма учета поступлений материала на склад.

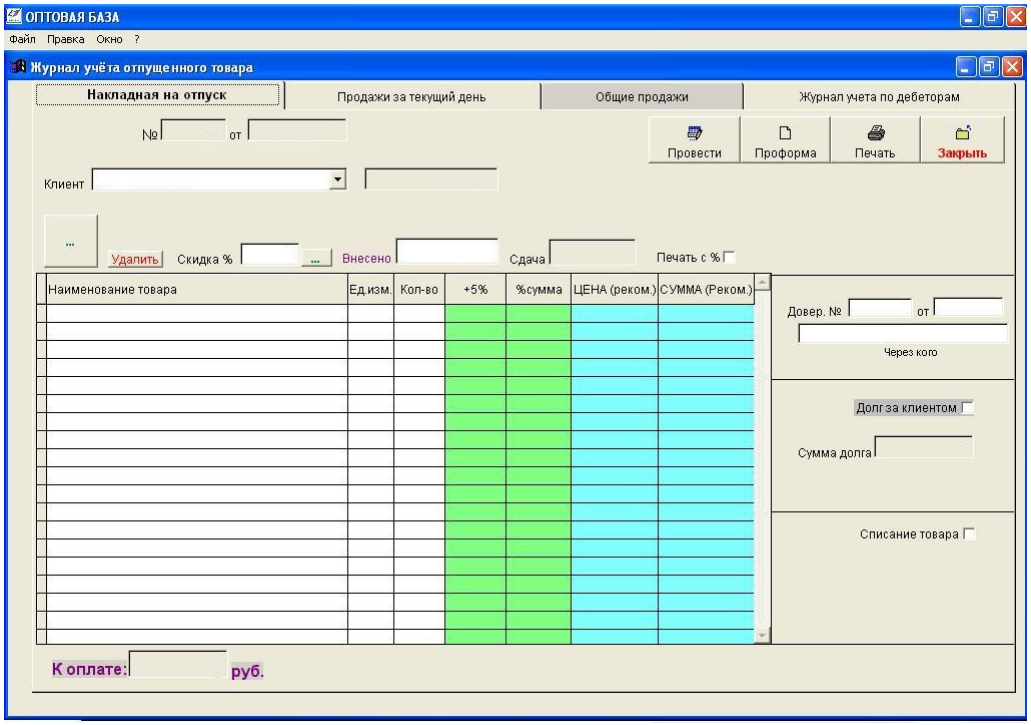


Рисунок 3.3– Форма учета отпущенного товара.

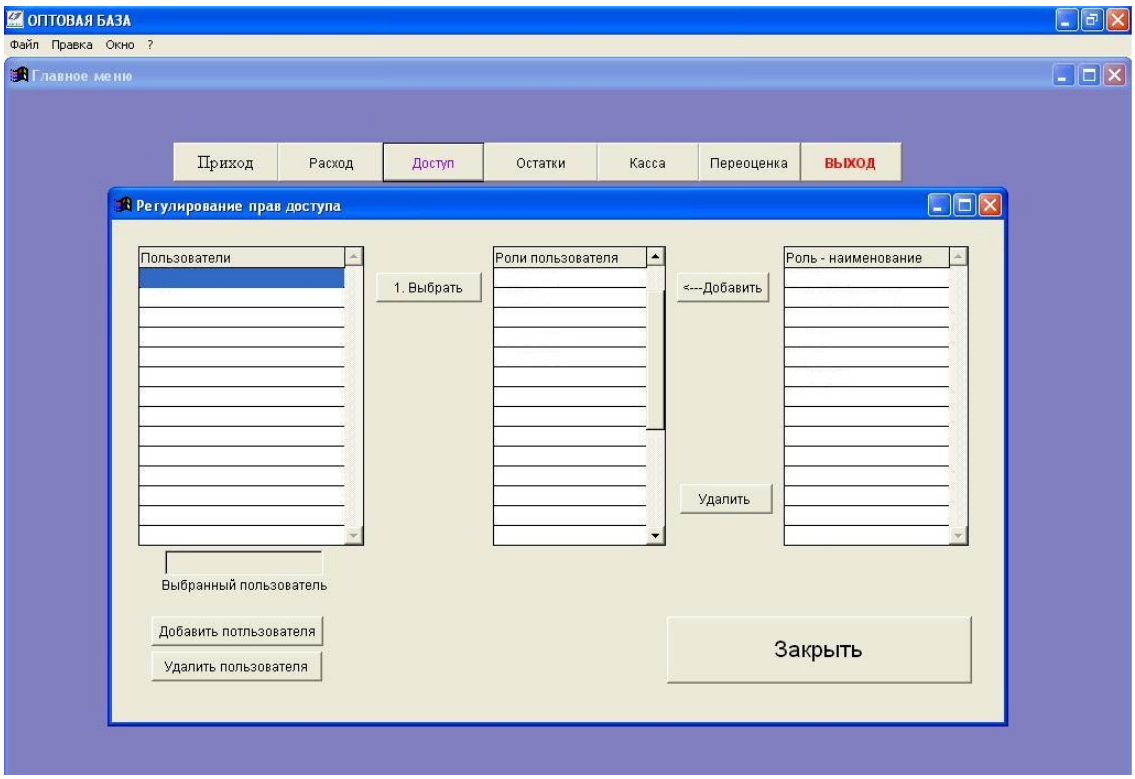


Рисунок 3.4– Форма регулирующая права доступа к программе.

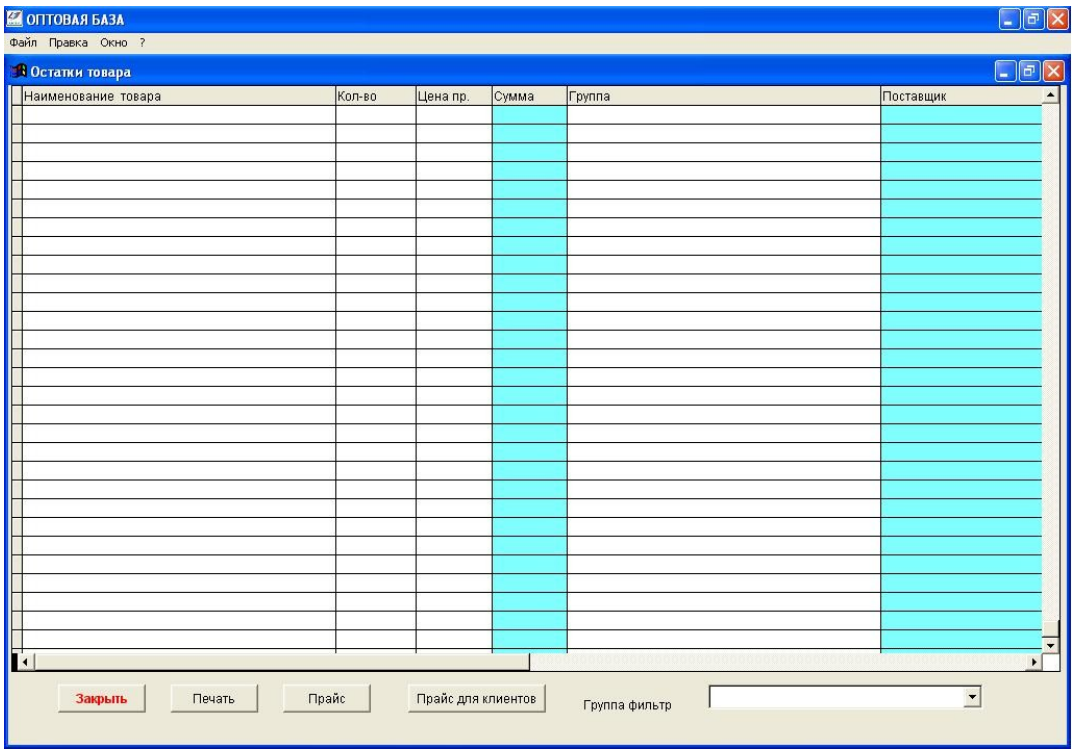


Рисунок 3.5– Форма остатков товара на складе.

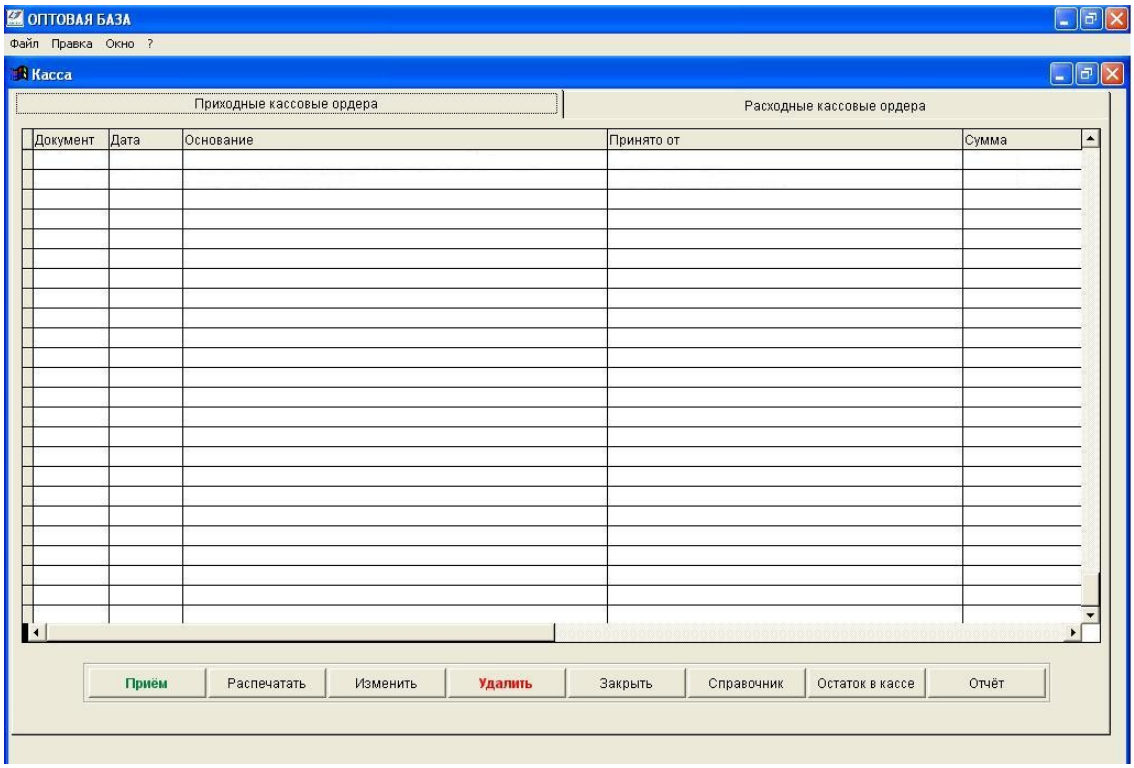


Рисунок 3.5–Форма о приходных кассовых ордерах и расходных кассовых ордерах.

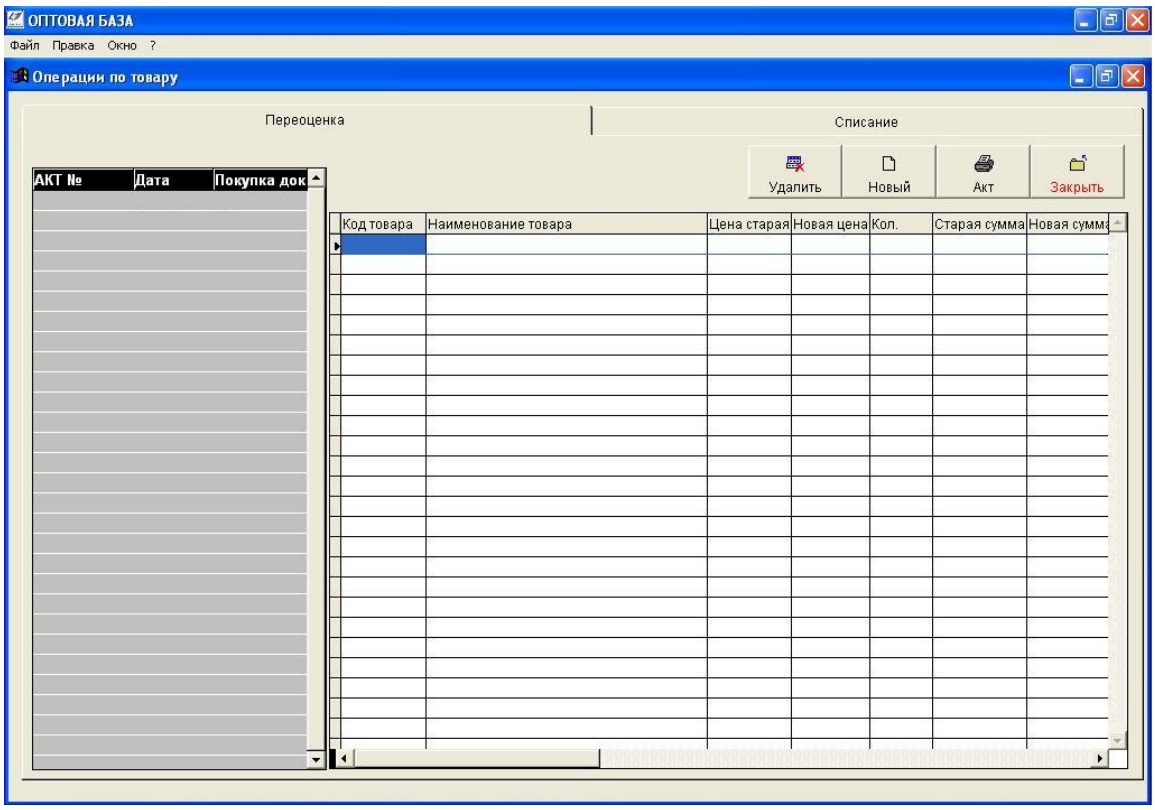


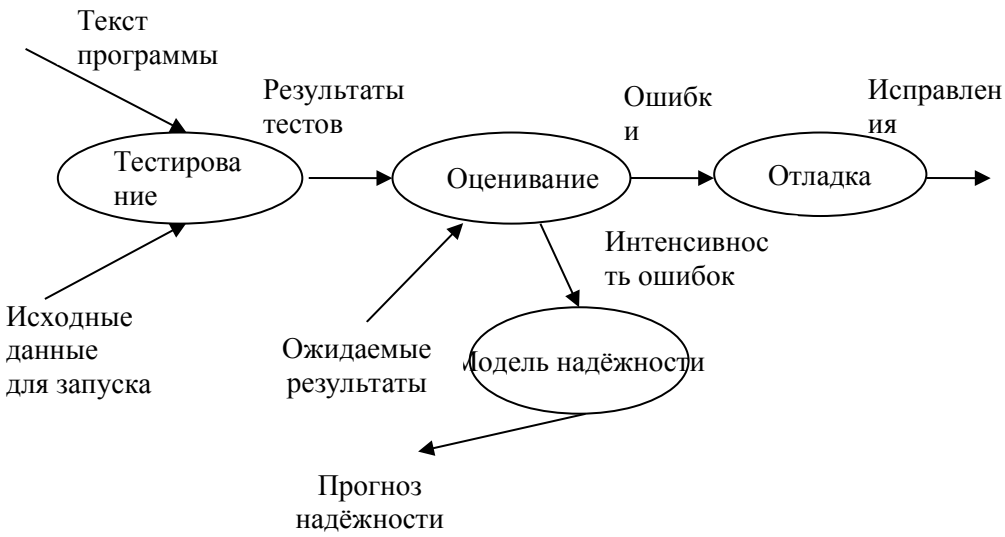
Рисунок 3.6–Форма операций по товару.

Тестирование приложения

Тестирование — процесс выполнения программы с целью обнаружения ошибок. Тестирование обеспечивает:

- обнаружение ошибок;
- демонстрацию соответствия функций программы ее назначению;
- демонстрацию реализации требований к характеристикам программы;
- отображение надежности как индикатора качества программы.

На рисунке 3.2 представлены информационные потоки процесса тестирования.



На входе процесса тестирования три потока:

- текст программы;
- исходные данные для запуска программы;
- ожидаемые результаты.

Выполняются тесты, все полученные результаты оцениваются. Это значит, что реальные результаты тестов сравниваются с ожидаемыми результатами. Когда обнаруживается несовпадение, фиксируется ошибка – начинается отладка.

После сбора и оценивания результатов тестирования начинается отображение качества и надежности ПО. Если регулярно встречаются серьезные ошибки, требующие проектных изменений, то качество и надежность ПО подозрительны, констатируется необходимость усиления тестирования.

Результаты, накопленные в ходе тестирования, могут оцениваться и более формальным способом. Для этого используют модели надежности ПО, выполняющие прогноз надежности по реальным данным об интенсивности ошибок.

Существуют 2 принципа тестирования программы:

- функциональное тестирование (тестирование «черного ящика»);
- структурное тестирование (тестирование «белого ящика»).

При тестировании методом «белого ящика» известна внутренняя структура программы. Объектом тестирования здесь является не внешнее, а внутреннее поведение программы. Проверяется корректность построения всех элементов программы и правильность их взаимодействия друг с другом.

Тестирование «черного ящика» (функциональное тестирование) позволяет получить комбинации входных данных, обеспечивающих полную проверку всех функциональных требований к программе /Error: Reference source not found/. Программное изделие здесь рассматривается как «черный ящик», чье поведение можно определить только исследованием его входов и соответствующих выходов.

Принцип «черного ящика» не альтернативен принципу «белого ящика». Скорее это дополняющий подход, который обнаруживает другой класс ошибок.

Тестирование «черного ящика» обеспечивает поиск следующих категорий ошибок:

- некорректных или отсутствующих функций;
- ошибок интерфейса;
- ошибок во внешних структурах данных или в доступе к внешней базе данных;
- ошибок характеристик (необходимая емкость памяти и т. д.);
- ошибок инициализации и завершения.

Подобные категории ошибок способами «белого ящика» не выявляются.

В отличие от тестирования «белого ящика», которое выполняется на ранней стадии процесса тестирования, тестирование «черного ящика» применяют на поздних стадиях тестирования. При тестировании «черного ящика» пренебрегают управляющей структурой программы. Здесь внимание концентрируется на информационной области определения программной системы. При тестировании на этом этапе основное внимание уделяется пригодности решения для работы в условиях живого производства. Основное

внимание уделяется исправлению ошибок и определению их важности, а также подготовки продукта к выпуску.

На этапе тестирования решают две основные задачи:

- Тестирование решения – выполняются планы тестирования, созданные на этапе планирования и расширенные и опробованные на этапе разработки;
- Пилотная эксплуатация – развертывание решения в тестовой среде и тестирование с привлечением будущих пользователей и реализацией реальных сценариев использования системы. Эта задача выполняется до начала этапа развёртывания.

Цель этапа тестирования – снижение риска, возникающего при вводе решения в промышленную эксплуатацию.

Для успеха этапа тестирования необходимо, чтобы произошла смена отношения к проекту и разработчик переключился с разработки новых функций на обеспечение должного качества решения.

На данной стадии разработки информационной системы необходимо провести следующие типы тестирования:

- Базовое тестирование – низкоуровневое техническое тестирование. Проводится самим разработчиком в процессе написания программного кода. Применяется метод «белого ящика», высокий риск ошибок.
- Тестирование на пригодность к использованию – высокоуровневое тестирование, выполняется тестировщиком и будущими пользователями продукта. Применяется метод «чёрного ящика».
- Альфа- и бета-тестирование – в терминах MSF альфа-код – это в основном все исходные тексты, созданные на этапе разработки модели процессов MSF, а бета-код – код, прошедший тестирование на этапе тестирования. Поэтому на этапе разработки модели процесса MSF тестируется альфа-код, а на этапе тестирования – бета-код.
- Тестирование совместимости – от разрабатываемого решения требуется возможность интеграции и способность к взаимодействию с

существующими системами и программными решениями. Данная форма тестирования ориентирована на проверку интегрируемости и способности разрабатываемого решения взаимодействовать с существующими системами. В данном конкретном случае будет проверятся корректность работы приложения на оборудовании пользователя и используемым пользователем программным обеспечением.

- Тестирование производительности – ориентировано на проверку того, удовлетворяет ли приложение требованиям по производительности и уровню комфортности работы по скорости.
- Тестирование документации и справочной системы – тестируются все разработанные сопровождающие документы и справочные системы.

Пилотная эксплуатация – это тестирование решения в промышленной среде. Основная задача пилотной эксплуатации – продемонстрировать, что решение способно стабильно работать условиях промышленной эксплуатации и удовлетворяет требованиям бизнеса. В процессе пилотной эксплуатации решение испытывается в реальных условиях. Пилотная эксплуатация дает возможность пользователям высказать свое мнение о работе продукта. Руководствуясь этим мнением разработчиком устраняются все возможные неполадки или создается план действий на случай непредвиденных обстоятельств. В конечном итоге, пилотная эксплуатация позволяет принять решение, стоит ли начинать полномасштабное развертывание или отложить до устранения неполадок, способных сорвать развертывание.

План процесса пилотной эксплуатации для разрабатываемой информационной системы приведен в таблице 3.2.

Таблица 3.2 – План пилотной эксплуатации

Действие	Описание
1. Выбор критериев успеха	Разработчик и участники опытного тестирования определяют критерии успешности и согласовывают их
2. Выбор пользователей и места	Формируется команда участников опытного тестирования

установки	со стороны пользователей и разработчиков. Определяется место развертывания пилотного процесса.
3. Подготовка пользователей и места установки	Проводится обучение пользователей – участников испытания. Подготавливается место установки.
4. Развёртывание опытной версии	Устанавливается опытная версия и включается в работу.
5. Поддержка и мониторинг опытной версии	Контроль работы пользователей и системы, оказание помощи в эксплуатации, сбор сведений о работе системы
6. Обратная связь с пользователями и оценка результатов	Пользователи высказывают своё мнение о работе системы, указывают на недочёты и ошибки.
7. Внесение изменений и дополнений	Исправляются обнаруженные ошибки, вносятся изменения в дизайн или процесс. Исправленные результаты предоставляются для работы и оценки пользователям.
8. Решения о развертывании	Если результаты работы опытного тестирования удовлетворяют пользователей принимается решение о развертывании системы.

3.2 Методика развертывания приложения

На этом этапе разработчик (или команда) развёртывает необходимые для решения технологии и компоненты, проект переходит на стадию сопровождения и поддержки, а заказчик окончательно утверждает его. После развертывания команда проводит оценку проекта и опрос пользователей, чтобы выяснить степень их удовлетворенности.

Цели этапа развертывания:

- ~ перенести решение в промышленную среду;
- ~ признание заказчиком факта завершения проекта.

Развертывание компонентов, характерных для конкретного места установки, состоит из нескольких стадий: подготовки, установки, обучения и формального одобрения.

Результатами этапа развертывания системы являются системы сопровождения и поддержки, хранилище документов, где размещаются все версии документов и кода, разработанных в течение проекта.

Для развертывания разрабатываемой системы был составлен план действий, который приведен в таблице 3.1.

Таблица 3.1 – План развертывания приложения

Действие	Описание действия
1. Резервное копирование	Производится резервное копирование данных пользователя при его участии и согласовании путем переноса информации на сменные носители (CD, DVD)
2. Установка базовых компонентов решения	Применение технологий, обеспечивающих работу решения. В данном случае – установка компонента Visual FoxPro
3. Установка клиентского приложения	Перенос на компьютер пользователя и установка окончательного варианта разработанной ИС и базы данных
4. Обучение	Производится обучение пользователей по работе с системой, разработчик убеждается в правильности и понимании работы ИС клиентами
5. Передача базы знаний проекта клиенту	Заказчику передаётся вся проектная документация
6. Закрытие проекта	Составляется отчёт о закрытии проекта. Заказчик подписывает акт приёмки.

Для нормального функционирования АРМ требуется операционная система Microsoft WindowsXP.

4 УПРАВЛЕНИЕ ИНФОРМАЦИОННЫМ ПРОЕКТОМ

4.1 Выбор жизненного цикла разработки

Одним из базовых понятий методологии проектирования ИС является понятие жизненного цикла ее программного обеспечения (ЖЦ ПО). ЖЦ ПО – это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации.

Основным нормативным документом, регламентирующим ЖЦ ПО, является международный стандарт ISO/IEC 12207 (ISO - International Organization of Standardization – Международная организация по стандартизации, IEC - International Electro technical Commission -

Международная комиссия по электротехнике). Он определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО.

Стандарт ISO/IEC 12207 не предлагает конкретную модель ЖЦ и методы разработки ПО. Под моделью ЖЦ можно понимать структуру, определяющую последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ. Модель ЖЦ зависит от специфики ИС и специфики условий, в которых создается и функционирует.

На сегодняшний день существует много моделей жизненного цикла программного обеспечения, но наиболее популярны и распространены две модели это:

- спиральная модель (смотри рисунок 4.1);
- итерационная модель.

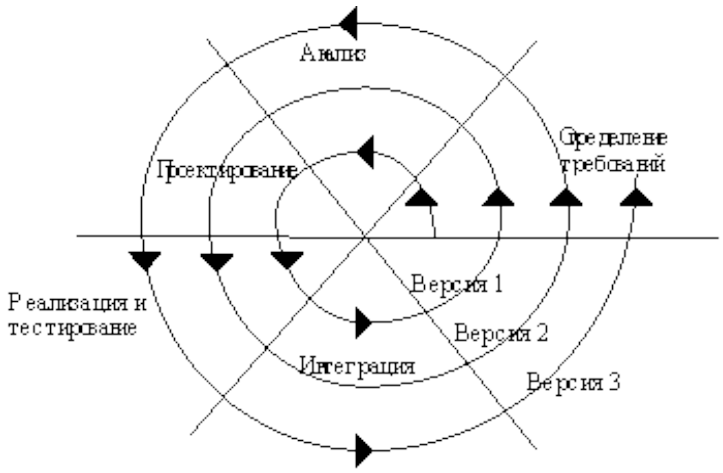


Рисунок 4.1 – Спиральная модель ЖЦ ПО

Для создания информационной системы, т.е. «Автоматизированное рабочее место сотрудника склада оптовая база», была выбрана итерационная. Отличительным свойством итерационной модели можно назвать то, что она представляет собой формальный метод, она состоит из независимых фаз, выполняемых последовательно, и подвержена частому обзору (рисунок 4.2). Итерационный подход хорошо зарекомендовал себя при построении ИС, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования, с тем, чтобы предоставить разработчикам свободу реализовать их как можно лучше с технической точки зрения.

Преимущества итерационной модели:

модель хорошо известна потребителям, не имеющим отношения к разработки ПО, и конечным пользователям.

- удобность и простота применения, т.к. все работы выполняются поэтапно (по фазам модели);
- стабильность требований;
- модель доступна для понимания;
- структурой модели может руководствоваться даже слабо подготовленный в техническом плане персонал (неопытный пользователь);
- модель упорядоченно справляется со сложностями и хорошо срабатывает для тех проектов, которые достаточно понятны;

- модель способствует осуществлению строгого контроля менеджмента проекта;
- облегчает работу менеджеру проекта по составлению плана и комплектации команды разработчиков.

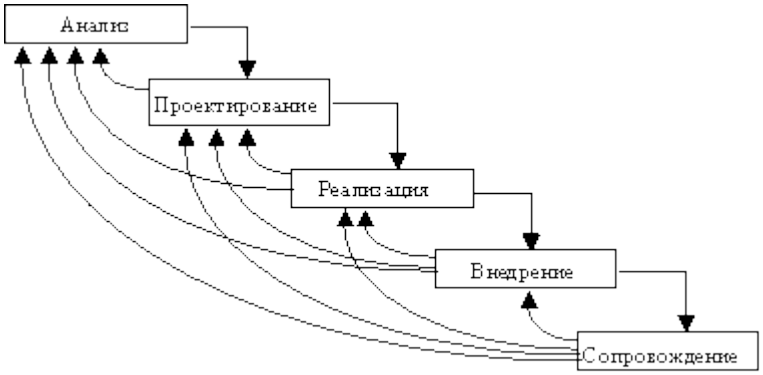


Рисунок 4.2 – Итерационная модель ЖЦ ПО

Фазы модели:

- на стадии анализа определяют функции, которые должна выполнять система, выделяют наиболее приоритетные из них, требующие проработки в первую очередь, описывают информационные потребности;
- на стадии проектирования, более подробно рассматриваются процессы системы. Анализируется и, при необходимости, корректируется функциональная модель. Строятся прототипы системы;
- на стадии реализации идет разработка системы;
- на стадии внедрения, готовый продукт внедряется в уже действующую систему организации. Производится обучение пользователей;
- на стадии сопровождения происходит обслуживание программного продукта (какое-либо добавление или изменение, для более функциональной работы продукта).

Выбор модели жизненного цикла разработки программного обеспечения является важным этапом. Поэтому для проекта выбор модели жизненного цикла разработки программного обеспечения может осуществляться в ходе использования следующих процессов.

- Анализ отличительных категорий проекта, помещённых в таблицах.
- Ответить на вопросы, приведённые для каждой категории, подчеркнув слова «да» и «нет».
- Расположить по степени важности категории или вопросы, относящиеся к каждой категории, относительно проекта, для которого выбирается приемлемая модель.

Команда разработчиков. Исходя из возможностей, отбор персонала в состав команды разработчиков проходит ещё до того момента, как будет выбрана модель жизненного цикла разработки программного обеспечения. Характеристики такой команды (смотри приложение Ж таблица Ж.1) играют важную роль в процессе выбора модели жизненного цикла, это означает, что команда может оказать значительную помощь в выборе модели жизненного цикла программного продукта, поскольку она несёт ответственность за удачное выполнение разработанной модели жизненного цикла.

Коллектив пользователей. На начальных стадиях проекта можно получить полное представление о коллективе пользователей (смотри приложение И таблица И.1), которые будут работать с разработанным программным обеспечением, и его будущей взаимосвязи с командой разработчиков на протяжении всего проекта. Такое представление помогает при выборе подходящей модели, поскольку некоторые модели требуют усиленного участия пользователей в процессе разработки и изучения проекта, так как требования могут незначительно изменяться пользователем в процессе разработки, то разработчику необходимо знать эти изменения и как эти изменения представить в программном обеспечении.

4.2 Определение цели и области действия программного проекта

Разрабатываемый программный продукт по учету товара на складе, позволит автоматизировать процесс поступления, структурирования и хранения данных о товаре на складе, а также упростить процесс выдачи

отчётов.

Целями программного проекта будут являться – создание и развертывание системы по учету товара. Данная система предназначена для внутреннего использования персоналом «Cleonelly» , в большей части сотрудниками склада предприятия.

Для определения области действия программного продукта, ниже будет описан, каким должен быть иле не должен быть программный проект.

Программный проект должен быть:

- для внутреннего использования в организации;
- проектом для осуществления многопользовательского доступа;
- проектом, который имеет возможность занесение, изменение и хранение сведений о товаре предприятия;
- проектом, который имеет возможность занесение, изменение и хранение сведений о пользователях системы;
- проектом, который имеет возможность занесение, изменение и хранение сведений о заказчиках и поставщиках организации, являющихся субъектами заключаемых сделок;
- проектом, который будет осуществлять формирование внешней отчетности.

4.3 Создание структуры пооперационного перечня работ

Для создания уникального продукта или услуги (результата проекта) нужно осуществить некоторую последовательность работ. Задача планирования проекта заключается в том, чтобы достаточно точно оценить сроки исполнения и стоимость этих работ. Чем точнее дана оценка, тем выше качество плана проекта. Чтобы дать точную оценку, нужно хорошо представлять состав работ проекта, то есть знать, какие именно работы нужно выполнить для получения его результата. Только после того, как составлен список проектных работ, оценивается длительность каждой из них,

и выделяются ресурсы, необходимые для их выполнения. И лишь затем можно оценить стоимость и сроки исполнения каждой задачи и, в результате сложения, общую стоимость и срок проекта. Вот почему определение состава работ является первым шагом при планировании проекта. Определение состава проектных работ начинается с определения этапов (или фаз) проекта. Например, в проекте создание системы «Учет товара на складе» могут быть выделены фазы:

- разработка требований к программному обеспечению;
- проектирование информационной системы;
- реализация и аттестация информационной системы;
- внедрение системы.

После того как состав фаз и их результаты определены, нужно определить последовательность этих фаз относительно друг друга и крайние сроки их исполнения. Затем нужно определить, из каких работ состоят фазы, в какой последовательности исполняются эти работы и в какие крайние сроки нужно уложиться при их исполнении.

Пооперационный перечень работ (рисунок 4.3) был спроектирован с помощью программного продукта такого, как MS Project 2003.

		Название задачи	Длительность	Начало	Окончание	Предшес	Названия ресурсов
2		**Разработка требований к программному обеспечению**	**26 дней?**	**Пн 09.01.07**	**Пн 13.02.07**		
3	✓	Анализ существующих решений по автоматизации предметной области	4 дней	Пн 09.01.07	Чт 12.01.07		Разработчик[50%];Бизн - аналитик[50%]
4	✓	Анализ предметной области	3 дней	Пт 13.01.07	Вт 17.01.07	3	Разработчик
5	✓	Сбор требований	5 дней	Ср 18.01.07	Вт 24.01.07	3;4	Разработчик
6	✓	Анализ и моделирование требований	6 дней	Ср 25.01.07	Ср 01.02.07	4;5	Разработчик[60%];Бизн
7	✓	Спецификация требований	3,5 дней?	Вт 31.01.07	Пт 03.02.07	5;6	Разработчик[75%];Бизн
8	✓	Аттестация требований	3,5 дней?	Чт 02.02.07	Вт 07.02.07	7;5	Разработчик[50%];Бизн
9	✓	Выбор методологии проектирования информационной системы	4 дней	Ср 08.02.07	Пн 13.02.07	4;6	Разработчик
10		Разработка завершена	0 дней	Пн 13.02.07	Пн 13.02.07	9	
11		**Проектирование информационной системы**	**26 дней**	**Вт 14.02.07**	**Вт 21.03.07**	**10**	
12	✓	Архитектурное проектирование	5 дней	Вт 14.02.07	Пн 20.02.07	7	Разработчик
13	✓	Проектирование пользовательского интерфейса	8 дней	Пн 20.02.07	Ср 01.03.07	12	Разработчик
14	✓	Проектирование баз данных	10 дней	Ср 01.03.07	Вт 14.03.07	12;13	Разработчик
15	✓	Обоснование выбора платформы создания информационной системы	2 дней	Ср 15.03.07	Чт 16.03.07	12	Разработчик
16	✓	Проектирование модулей (функциональная модель)	3 дней	Пт 17.03.07	Вт 21.03.07	14	Разработчик
17		Проектирование завершено	0 дней	Вт 21.03.07	Вт 21.03.07	16	
18		**Реализация и аттестация информационной системы**	**34 дней**	**Ср 22.03.07**	**Пн 08.05.07**	**17**	
19	✓	Реализация приложения	14 дней	Ср 22.03.07	Пн 10.04.07	13;14	Разработчик
20	✓	Взаимодействие приложения с источниками данных	8 дней	Вт 11.04.07	Чт 20.04.07	19	Разработчик
21	✓	Тестирование приложения	4 дней	Пт 28.04.07	Ср 03.05.07	19;20	Тестер[80%];Разработч
22	✓	Методика развертывания приложения	3 дней	Чт 04.05.07	Пн 08.05.07	19;20;21	Разработчик
23		Реализация завершена	0 дней	Пн 08.05.07	Пн 08.05.07	22	
24		**Внедрение системы**	**9 дней?**	**Вт 09.05.07**	**Пт 19.05.07**	**23**	
25	✓	Выбор окончательного метода развертывания системы	2 дней	Вт 09.05.07	Ср 10.05.07	22	Разработчик[85%];Бизн
26	✓	Создание плана развертывания системы	3 дней	Ср 10.05.07	Пт 12.05.07	25	Разработчик
27	✓	Развертывание системы	1 день?	Сб 13.05.07	Пн 15.05.07	25;26	Разработчик
28	✓	Обучение персонала	4 дней	Вт 16.05.07	Пт 19.05.07	27	Разработчик
29		Внедрение завершено	0 дней	Пт 19.05.07	Пт 19.05.07	28	

Рисунок 4.3 – Пооперационный перечень работ

4.4 Оценка длительности и стоимости разработки ПО

Оценка длительности. Она определяется после построения пооперационного перечня работ (рисунок 4.3, пункт 4.3). Данную оценку длительности можно увидеть при помощи диаграммы Ганта (приложение К).

Диаграммы являются графическим средством отображения содержащейся в проектном файле информации. Из диаграмм можно получить визуальное представление о последовательности задач, их относительной длительности и длительности проекта в целом.

Диаграмма Ганта — это один из наиболее популярных способов графического представления плана проекта, применяемый во многих программах управления проектами. [34]

В MS Project диаграмма Ганта является основным средством визуализации плана проекта. Эта диаграмма представляет собой график, на котором по горизонтали размещена шкала времени, а по вертикали расположен список задач. При этом длина отрезков, обозначающих задачи, пропорциональна длительности задач.

На диаграмме Ганта рядом с отрезками может отображаться дополнительная информация (рядом с задачами отображаются названия задействованных в них ресурсов и их загрузка при выполнении задачи). [34]

Оценка затрат

Проект состоит из **задач**, то есть активностей, направленных на достижение определенного результата. Чтобы задача могла быть выполнена, на нее выделяются **ресурсы**.

Важное свойство ресурсов — стоимость (Cost (Затраты)) их использования в проекте. В MS Project есть два типа стоимости ресурсов: повременная ставка и стоимость за использование.

Повременная ставка (Rate) выражается в стоимости использования ресурса в единицу времени, например 100 рублей в час или 1000 рублей в день. В таком случае стоимость участия ресурса в проекте составит время, в течение которого он работает в проекте, умноженное на почасовую ставку.

В данном случае использовалась повременная ставка (рисунок 4.4) Общие затраты на использование ресурсов можно на рисунке 4.5.

		Название ресурса	Стандартная ставка	Ставка сверхурочных
1		Разработчик	50,00р./ч	0,00р./ч
2		Бизнес - аналитик	45,00р./ч	0,00р./ч
3		Тестер	38,00р./ч	0,00р./ч

Рисунок 4.4 – Повременная ставка в использовании ресурса

На данном рисунке можно увидеть, что разработчик системы при выполнении проекта получает 50 рублей в час; бизнес-аналитик получает 45 рублей в час, тестер 38 рублей в час. Ставка сверхурочных не учитывается.

	Название задачи	Общие затраты	Базовый план	Отклонение	Фактические	Оставшиеся
1	**Система учета товара на складе**	**39 396,47р.**	**15 449,67р.**	**23 946,80р.**	**39 396,47р.**	**0,00р.**
2	**Разработка требований к программному обеспечению**	**13 169,00р.**	**5 099,00р.**	**8 070,00р.**	**13 169,00р.**	**0,00р.**
3	Анализ существующих решений по автоматизации предметной области	2 320,00р.	880,00р.	1 440,00р.	2 320,00р.	0,00р.
4	Анализ предметной области	1 200,00р.	480,00р.	720,00р.	1 200,00р.	0,00р.
5	Сбор требований	2 000,00р.	800,00р.	1 200,00р.	2 000,00р.	0,00р.
6	Анализ и моделирование требований	2 304,00р.	864,00р.	1 440,00р.	2 304,00р.	0,00р.
7	Спецификация требований	1 715,00р.	665,00р.	1 050,00р.	1 715,00р.	0,00р.
8	Аттестация требований	2 030,00р.	770,00р.	1 260,00р.	2 030,00р.	0,00р.
9	Выбор методолоии проектирования информационной системы	1 600,00р.	640,00р.	960,00р.	1 600,00р.	0,00р.
10	Разработка завершена	0,00р.	0,00р.	0,00р.	0,00р.	0,00р.
11	**Проектирование информационной системы**	**11 200,00р.**	**4 480,00р.**	**6 720,00р.**	**11 200,00р.**	**0,00р.**
12	Архитектурное проектирование	2 000,00р.	800,00р.	1 200,00р.	2 000,00р.	0,00р.
13	Проектирование пользовательского интерфейса	3 200,00р.	1 280,00р.	1 920,00р.	3 200,00р.	0,00р.
14	Проектирование баз данных	4 000,00р.	1 600,00р.	2 400,00р.	4 000,00р.	0,00р.
15	Обоснование выбора платформы создания информационной системы	800,00р.	320,00р.	480,00р.	800,00р.	0,00р.
16	Проектирование модулей (функциональная модель)	1 200,00р.	480,00р.	720,00р.	1 200,00р.	0,00р.
17	Проектирование завершено	0,00р.	0,00р.	0,00р.	0,00р.	0,00р.
18	**Реализация и аттестация информационной системы**	**11 039,47р.**	**4 282,67р.**	**6 756,80р.**	**11 039,47р.**	**0,00р.**
19	Реализация приложения	5 600,00р.	2 240,00р.	3 360,00р.	5 600,00р.	0,00р.
20	Взаимодействие приложения с источниками данных	3 200,00р.	1 280,00р.	1 920,00р.	3 200,00р.	0,00р.
21	Тестирование приложения	1 039,47р.	282,67р.	756,80р.	1 039,47р.	0,00р.
22	Методика развертывания приложения	1 200,00р.	480,00р.	720,00р.	1 200,00р.	0,00р.
23	Реализация завершена	0,00р.	0,00р.	0,00р.	0,00р.	0,00р.
24	**Внедрение системы**	**3 988,00р.**	**1 588,00р.**	**2 400,00р.**	**3 988,00р.**	**0,00р.**
25	Выбор окончательного метода развертывания системы	788,00р.	308,00р.	480,00р.	788,00р.	0,00р.
26	Создание плана развертывания системы	1 200,00р.	480,00р.	720,00р.	1 200,00р.	0,00р.
27	Развертывание системы	400,00р.	160,00р.	240,00р.	400,00р.	0,00р.
28	Обучение персонала	1 600,00р.	640,00р.	960,00р.	1 600,00р.	0,00р.

Рисунок 4.5 – Общие затраты на использовании ресурсов проекта

4.5 Распределение ресурсов проекта

Фрагмент распределения ресурсов для системы «Учета товара на складе» можно увидеть на рисунке 4.6

	Название ресурса	Трудозатраты	Подробности	6						06 Фев '06		
				В	С	Ч	П	С	В	П	В	С
1	Разработчик	714,4 ч	Трудозатр.	12,8ч	12,8ч	12ч	12ч			8ч	8ч	8ч
	Анализ существующих решений по автомати	32 ч	Трудозатр.									
	Анализ предметной области	24 ч	Трудозатр.									
	Сбор требований	40 ч	Трудозатр.									
	Анализ и моделирование требований	28,8 ч	Трудозатр.	4,8ч	4,8ч							
	Спецификация требований	28 ч	Трудозатр.	8ч	8ч	8ч	4ч					
	Аттестация требований	28 ч	Трудозатр.			4ч	8ч			8ч	8ч	
	Выбор методологии проектирования информ	32 ч	Трудозатр.									8ч
	Архитектурное проектирование	40 ч	Трудозатр.									
	Проектирование пользовательского интерф	64 ч	Трудозатр.									
	Проектирование баз данных	80 ч	Трудозатр.									
	Обоснование выбора платформы создания информационной системы	16 ч	Трудозатр.									
	Проектирование модулей (функциональная м	24 ч	Трудозатр.									
	Реализация приложения	112 ч	Трудозатр.									
	Взаимодействие приложения с источниками данных	64 ч	Трудозатр.									
	Методика развертывания приложения	24 ч	Трудозатр.									
	Выбор окончательного метода развертыван	13,6 ч	Трудозатр.									
	Создание плана развертывания системы	24 ч	Трудозатр.									
	Развертывание системы	8 ч	Трудозатр.									
	Обучение персонала	32 ч	Трудозатр.									
2	Бизнес - аналитик	58,6 ч	Трудозатр.	5,2ч	5,2ч	4ч	5ч			4ч	4ч	
	Анализ существующих решений по автомати	16 ч	Трудозатр.									
	Анализ и моделирование требований	19,2 ч	Трудозатр.	3,2ч	3,2ч							
	Спецификация требований	7 ч	Трудозатр.	2ч	2ч	2ч	1ч					
	Аттестация требований	14 ч	Трудозатр.			2ч	4ч			4ч	4ч	
	Выбор окончательного метода развертыван	2,4 ч	Трудозатр.									
3	Тестер	25,6 ч	Трудозатр.									
	Тестирование приложения	25,6 ч	Трудозатр.									

Рисунок 4.6 – Фрагмент распределения ресурсов проекта

Для каждой работы выполняемой в проекте сопоставляется ресурс, который будет выполнять данную работу. На рисунке показано общее количество трудозатрат каждого из ресурсов и конкретное количество часов затраченных в определенный день.

4.6 Оценка экономической эффективности проекта

Расчет экономической эффективности проекта является важным этапом. Именно здесь будет рассчитываться экономическая эффективность проекта. Данный расчет покажет, на сколько выгоден проект или проект совсем убыточный. При расчете экономической эффективности проекта, будет необходимо рассчитать и срок окупаемости проекта. Срок окупаемости будет показывать период, за который окупиться проект.

Входные данные.

Дополнительная прибыль от реализации проекта (DP) = 38000 рублей.

Дополнительна прибыль была спрогнозирована экспертами предприятия.

Стартовые инвестиции (IC) = 39396,47 рублей. Стартовые инвестиции соответствуют общим затратам на использование ресурсов проекта (рисунок 4.5 пункта 4.6)

Ставка дисконтирования (i) = 12%.

Срок, на который рассчитан проект (n) = 2 года.

Дополнительная прибыль от реализации проекта (DP) = 38000 рублей.

Ежегодные затраты на реализацию проекта (Z_1) = 15000 рублей.

Ежегодные затраты на реализацию проекта (Z_2) = 10000 рублей.

Годовые денежные поступления можно рассчитать по формуле 4.1.

$$R_k = DP - Z_k \quad (4.1)$$

Годовые денежные поступления (R_1) = 23000 рублей.

Годовые денежные поступления (R_2) = 28000 рублей.

При оценке инвестиционных проектов используется метод расчета чистого приведенного дохода, который предусматривает дисконтирование денежных потоков: все доходы и затраты приводятся к одному моменту времени.

Центральным показателем в рассматриваемом методе является показатель NPV (net present value) – текущая стоимость денежных потоков. Это обобщенный конечный результат инвестиционной деятельности в абсолютном измерении.

Важным моментом является выбор ставки дисконтирования, которая должна отражать ожидаемый усредненный уровень ссудного процента на финансовом рынке.

Чистый приведенный доход (NPV) рассчитывается по формуле 4.2

$$NPV = \sum_{k=1}^{n} \left[\frac{R_k}{(1+i)^k} \right] - IC \quad (4.2)$$

R_k – годовые денежные поступления в течении n лет.

k – количество лет на сколько рассчитан проект.

IC – стартовые инвестиции.

i – ставка дисконтирования.

По расчетам данной формулы **NPV** = 3460,67 руб.

Показатель NPV является абсолютным приростом, поскольку оценивает, на сколько приведенный доход перекрывает приведенные затраты. Так как NPV > 0, то проект следует принять.

Коэффициент возврата инвестиций (ROI) рассчитывается по формуле 4.3

$$ROI = \frac{NPV + IC}{IC} * 100\% \quad (4.3)$$

По расчетам (ROI) = 108,78%

Далее необходимо рассчитать срок окупаемости проекта по упрощенной формуле: **$n_{ок}$ = Число лет до года окупаемости + (Не возмещенная стоимость на начало года окупаемости / Приток наличности в течение года окупаемости)**

Таблица 4.1 ~ Вспомогательная таблица расчёта срока окупаемости проекта

Показатели	**Значения**		
Период (лет)	0	1	2
Денежный поток	-39396,47	23000	28000
Дисконтированный денежный поток	-39396,47	20535,71	22321,43
Накопленный дисконтированный денежный поток	-39396,47	-18860,76	3460,67

$$n_{ok} = 1 - \frac{-18860,76}{22321,43} = 1,84$$

Срок окупаемости n_{ok} = 1,84 года (1 год и 11 месяцев)

Так как ROI = > 100% (а именно = 108,78%) то проект считается прибыльным.

Далее необходимо рассчитать индекс прибыльности (PI). Индекс прибыльности показывает, сколько мы заработаем денег с каждого вложенного рубля, и рассчитывается по формуле 4.4

$$PI = \sum_{k=1}^{n} \left[\frac{R_k}{(1+i)^k} \right] / IC \quad (4.4)$$

Таким образом, индекс прибыльности равен (PI) =1,2