

Министерство науки и высшего образования Российской Федерации  
федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Кузбасский государственный технический университет  
имени Т.Ф. Горбачева»

Институт профессионального образования  
Кафедра информатики и информационных систем

Галина Алексеевна Алексеева

## **ВНЕДРЕНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

Методические материалы к практическим занятиям,  
лабораторным и самостоятельным работам

Рекомендовано цикловой методической комиссией  
специальности СПО 09.02.07 «Информационные системы и про-  
граммирование» в качестве электронного издания для использо-  
вания в образовательном процессе

Кемерово 2024

Рецензенты: С. А. Асанов – преподаватель кафедры информатики и информационных систем, старший преподаватель кафедры информационных и автоматизированных производственных систем ФГБОУ ВО «Кузбасский государственный технический университет имени Т. Ф. Горбачева»

**Алексеева, Г.А. Внедрение информационной системы:** методические материалы к практическим занятиям, лабораторным и самостоятельным работам для обучающихся специальности СПО 09.02.07 «Информационные системы и программирование» / сост. Г.А. Алексеева, Кузбасский государственный технический университет имени Т. Ф. Горбачева. – Кемерово, 2024. – Текст : электронный.

Методические материалы для дисциплины «Внедрение информационной системы» описывают содержание практических занятий, лабораторных и самостоятельных работ, материал, необходимый для успешного изучения дисциплины.

© Кузбасский государственный  
технический университет  
имени Т. Ф. Горбачева, 2024  
© Алексеева Г.А.  
составление, 2024

## ОГЛАВЛЕНИЕ

1 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА СЦЕНАРИЯ ВНЕДРЕНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ РАБОЧЕГО МЕСТА.....	5
2 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ НА ВНЕДРЕНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ .....	47
3 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА ГРАФИКА РАЗРАБОТКИ И ВНЕДРЕНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ .....	55
4 ЛАБОРАТОРНАЯ РАБОТА. СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОЛОГИЙ ПРОЕКТИРОВАНИЯ.....	81
5 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. АНАЛИЗ БИЗНЕС-ПРОЦЕССОВ ПОДРАЗДЕЛЕНИЯ .....	87
6 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА И ОФОРМЛЕНИЕ ПРЕДЛОЖЕНИЙ ПО РАСШИРЕНИЮ ФУНКЦИОНАЛЬНОСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ .....	99
7 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА ПЕРЕЧНЯ ОБУЧАЮЩЕЙ ДОКУМЕНТАЦИИ НА ИНФОРМАЦИОННУЮ СИСТЕМУ .....	109
8 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА РУКОВОДСТВА ОПЕРАТОРА.....	115
9 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА МОДЕЛЕЙ ИНТЕРФЕЙСОВ ПОЛЬЗОВАТЕЛЕЙ .....	120
10 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. НАСТРОЙКА ДОСТУПА К СЕТЕВЫМ УСТРОЙСТВАМ .....	124
11 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. НАСТРОЙКА ПОЛИТИКИ БЕЗОПАСНОСТИ .....	131
12 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. ВЫПОЛНЕНИЕ ЗАДАЧ ТЕСТИРОВАНИЯ В ПРОЦЕССЕ ВНЕДРЕНИЯ. ....	153
13 САМОСТОЯТЕЛЬНАЯ РАБОТА. ОСНОВНЫЕ ЭТАПЫ И МЕТОДОЛОГИИ В ПРОЕКТИРОВАНИИ И ВНЕДРЕНИИ ИНФОРМАЦИОННЫХ СИСТЕМ .....	162

14 САМОСТОЯТЕЛЬНАЯ РАБОТА. ОРГАНИЗАЦИЯ И ДОКУМЕНТАЦИЯ ПРОЦЕССА ВНЕДРЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ .....	172
15 САМОСТОЯТЕЛЬНАЯ РАБОТА. ИНСТРУМЕНТЫ И ТЕХНОЛОГИИ ВНЕДРЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ .....	184
ЛИТЕРАТУРА .....	194

# 1 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА СЦЕНАРИЯ ВНЕДРЕНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ РАБОЧЕГО МЕСТА

## 1.1 ЦЕЛЬ РАБОТЫ

Цель работы – изучить принципы создания и внедрения информационной системы.

## 1.2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

### 1.2.1 Принципы создания информационной системы

Многие пользователи компьютерной техники и программного обеспечения неоднократно сталкивались с ситуацией, когда программное обеспечение, хорошо работающее на одном компьютере, не работает на другом таком же устройстве. Или системные блоки одного вычислительного устройства не стыкуются с аппаратной частью другого. Или информационная система другой компании упорно не желает обрабатывать данные, которые вы подготовили в информационной системе у себя на рабочем месте. Эта проблема называется проблемой совместимости вычислительных, телекоммуникационных и информационных устройств.

Развитие систем и средств вычислительной техники, расширенное их внедрение во все сферы науки, техники, сферы обслуживания и быта привели к необходимости объединения конкретных вычислительных устройств и реализованных на их основе информационных систем в единые информационно-вычислительные системы (ИВС) и среды. При этом разработчики ИВС столкнулись с рядом проблем.

Например, разнородность технических средств вычислительной техники с точки зрения организации вычислительного процесса, архитектуры, системы команд, разрядности процессора и шины данных и т. д. потребовала создания физических интерфейсов, реализующих, как правило, взаимную совместимость устройств. При увеличении числа типов интегрируемых устройств сложность организации физического интерфейса меж-

ду ними существенно возрастала. Разнородность программируемых сред, реализуемых в конкретных вычислительных устройствах и системах, с точки зрения многообразия операционных систем, различия в разрядности и прочих особенностей привела к созданию программных интерфейсов между устройствами и системами. При этом необходимо отметить, что достигнуть полной совместимости программных продуктов, разработанных для конкретной программной среды, в другой среде удавалось не всегда. Разнородность интерфейсов общения в системе «человек-компьютер» требовала постоянного согласования программно-аппаратного обеспечения и переобучения кадров.

#### 1) Принцип «открытости» информационной системы.

Решение проблем совместимости привело к разработке большого числа международных стандартов и соглашений в сфере применения информационных технологий и разработки информационных систем. Основопологающим понятием стало понятие открытые системы.

Термин «открытая система» сегодня можно определить как «исчерпывающий и согласованный набор международных стандартов на информационные технологии и профили функциональных стандартов, которые специфицируют интерфейсы, службы и поддерживающие их форматы, чтобы обеспечить взаимодействие и мобильность программных приложений, данных и персонала».

Это определение, сформулированное специалистами института IEEE (Institute of Electrical and Electronic Engineers), унифицирует содержание среды, которую предоставляет открытая система для широкого использования. В настоящее время общепризнанным координационным центром по разработке и согласованию стандартов открытых систем является OASIS (Organization for the Advancement of Structured Information Standards).

Общие свойства открытых информационных систем можно сформулировать следующим образом:

- расширяемость/масштабируемость: обеспечение возможности добавления новых функций ИС или изменения некоторых уже имеющихся при неизменных остальных функциональных частях ИС;
- мобильность/переносимость: обеспечение возможности переноса программ, данных при модернизации или замене

аппаратных платформ ИС и возможности работы с ними специалистов, пользующихся ИТ, без их переподготовки при изменениях ИС;

- взаимодействие: способность к взаимодействию с другими ИС (технические средства, на которых реализована информационная система, объединяются сетью или сетями различного уровня: от локальной до глобальной);

- стандартизуемость: ИС проектируются и разрабатываются на основе согласованных международных стандартов и предложений, реализация открытости осуществляется на базе функциональных стандартов (профилей) в области информационных технологий;

- дружелюбность к пользователю: развитые унифицированные интерфейсы в процессах взаимодействия в системе «человек-машина», позволяющие работать пользователю, не имеющему специальной «компьютерной» подготовки.

Новый взгляд на открытые системы определяется тем, что эти черты рассматриваются в совокупности, как взаимосвязанные, и реализуются в комплексе, что вполне естественно, поскольку все указанные выше свойства дополняют друг друга. Только в совокупности возможности открытых систем позволяют решать проблемы проектирования, разработки и внедрения современных информационных систем.

Обобщенная структура любой ИС может быть представлена двумя взаимодействующими частями:

- функциональной части, включающей прикладные программы, которые реализуют функции прикладной области;

- среды или системной части, обеспечивающей исполнение прикладных программ.

- С этим разделением тесно связаны две группы вопросов стандартизации:

- стандарты интерфейсов взаимодействия прикладных программ со средой ИС, прикладной программный интерфейс (Application Program Interface – API);

- стандарты интерфейсов взаимодействия самой ИС с внешней для нее средой (External Environment Interface – EEI).

Эти две группы интерфейсов определяют спецификации внешнего описания среды ИС – архитектуру, с точки зрения ко-

нечного пользователя, проектировщика ИС, прикладного программиста, разрабатывающего функциональные части ИС.

Спецификации внешних интерфейсов среды ИС и, как будет видно далее, спецификации интерфейсов взаимодействия между компонентами самой среды, – это точные описания всех необходимых функций, служб и форматов определенного интерфейса. Совокупность таких описаний составляет эталонную модель открытых систем (Reference Open System Model).

Эта модель используется более 20 лет и определяется системной сетевой архитектурой (SNA), предложенной IBM в 1974 году. Она основана на разбиении вычислительной среды на семь уровней, взаимодействие между которыми описывается соответствующими стандартами, и обеспечивает связь уровней вне зависимости от построения уровня в каждой конкретной реализации (рисунок 1.1). Основным достоинством этой модели является детальное описание связей в среде с точки зрения технических устройств и коммуникационных взаимодействий. Вместе с тем она не принимает в расчет взаимосвязь с учетом мобильности прикладного программного обеспечения.

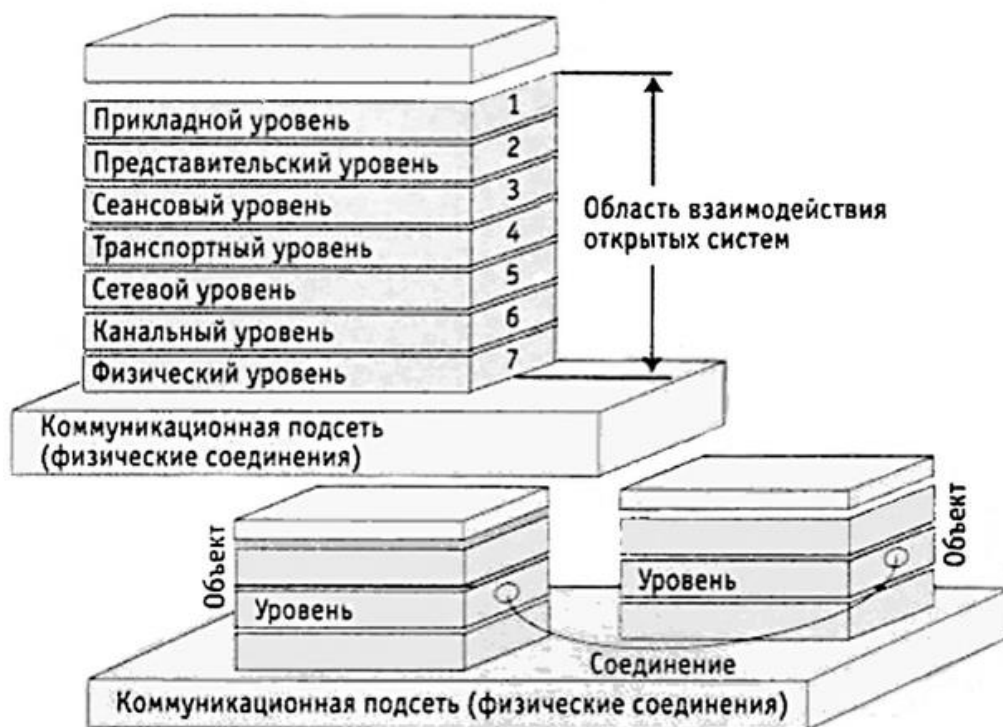


Рисунок 1.1 – Семиуровневая модель взаимодействия информационных систем



Эталонная модель среды открытых систем (OSE/RM) определяет разделение любой информационной системы на приложения (прикладные программы и программные комплексы) и среду, в которой эти приложения функционируют. Между приложениями и средой определяются стандартизованные интерфейсы (API), которые являются необходимой частью профилей любой открытой системы. Кроме того, в профилях ИС могут быть определены унифицированные интерфейсы взаимодействия функциональных частей друг с другом и интерфейсы взаимодействия между компонентами среды ИС.

Методологически важно наряду с рассмотренными моделями среды ИС предложить модель создания ИС, которая имела бы те же аспекты функциональных групп компонентов (пользователи, функции, данные, коммуникации). Такой подход обеспечит сквозной процесс проектирования и сопровождения на всех стадиях эксплуатации ИС и возможность обоснованного выбора стандартов на разработку систем и документирование проектов.

Компания является сложной онтологической (понятийной) структурой, состоящий из определенной совокупности сущностей и взаимосвязей (рисунок 1.2).



Рисунок 1.2 – Онтологическое поле современной компании

Взаимодействия между её элементами, определяемые бизнес-логикой и закреплённые в наборе бизнес-правил, и является деятельностью компании. Информационная система «отражает» логику и правила, организуя и преобразуя информационные потоки, автоматизирует процессы работы с данными и информацией и визуализирует результаты в виде наборов отчетных форм. Поэтому для начала следует создать бизнес-модель предприятия, которая является отображением предприятия и его информационно-управляющей системы.

При создании модели формируется «язык общения» руководителей предприятия, консультантов, разработчиков и будущих пользователей, позволяющий выработать единое представление о том, ЧТО и КАК должна делать система управления предприятием (корпоративная система управления). Такая бизнес-модель – осязаемый результат, с помощью которого можно максимально конкретизировать цели внедрения ИС и определиться со следующими параметрами проекта:

- основные цели бизнеса, которые можно достичь посредством автоматизации процессов;
- перечень участков и последовательность внедрения модулей ИС;
- фактическая потребность в объемах закупаемого программного и аппаратного обеспечения;
- реальные оценки сроков развертывания и запуска ИСУ;
- ключевых пользователей ИС и уточненный список членов команды внедрения;
- степень соответствия выбранного вами прикладного программного обеспечения специфике бизнеса вашей компании.

В основе модели всегда лежат бизнес-цели предприятия, полностью определяющие состав всех базовых компонентов модели:

- бизнес-функции, описывающие ЧТО делает бизнес;
- основные, вспомогательные и управленческие процессы, описывающие КАК предприятие выполняет свои бизнес-функции;
- организационно-функциональную структуру, определяющую ГДЕ исполняются бизнес-функции и бизнес-процессы;

- фазы, определяющие КОГДА (в какой последовательности) должны быть внедрены те или иные бизнес-функции;
- роли, определяющие КТО исполняет бизнес-функции и КТО является «хозяином» бизнес-процессов;
- правила, определяющие связь и взаимодействие между всеми ЧТО, КАК, ГДЕ, КОГДА и КТО.

После построения бизнес-модели (или параллельно с этим) можно приступить к формированию модели проектирования, реализации и внедрения самой ИС (рисунок 1.3).



Рисунок 1.3 – Модели проектирования, реализации и внедрения

Опыт создания и использования «заказных» ИС позволяет условно выделить следующие основные этапы их жизненного цикла:

- определение требований к системе и их анализ – определение того, что должна делать система;
- проектирование – определение того, как система будет делать то, что она должна делать; проектирование это, прежде всего, спецификация подсистем, функциональных компонентов и способов их взаимодействия в системе;
- разработка – создание функциональных компонентов и отдельных подсистем, соединение подсистем в единое целое;
- тестирование – проверка функционального соответствия системы показателям, определенным на этапе анализа;
- внедрение – установка и ввод системы в действие;
- функционирование – штатный процесс эксплуатации в соответствии с основными целями и задачами ИС;
- сопровождение – обеспечение штатного процесса эксплуатации системы на предприятии заказчика.

Определение требований к системе и анализ является первым этапом создания ИС, на котором требования заказчика уточ-

няются, согласуются, формализуются и документируются. Фактически на этом этапе дается ответ на вопрос: «Для чего предназначена и что должна делать информационная система?». Именно здесь лежит ключ к успеху всего проекта.

Целью системного анализа является преобразование общих, расплывчатых знаний об исходной предметной области (требований заказчика) в точные определения и спецификации для разработчиков, а также генерация функционального описания системы. На этом этапе определяются и специфицируются:

- внешние и внутренние условия работы системы;
- функциональная структура системы;
- распределение функций между человеком и системой, интерфейсы;
- требования к техническим, информационным и программным компонентам системы,
- требования к качеству и безопасности ;
- состав технической и пользовательской документации;
- условия внедрения и эксплуатации.

Разработка перечисленных выше спецификаций при создании ИС, предназначенной для автоматизации управленческих процессов, в общем случае проходит четыре стадии.

Первая стадия анализа – структурный анализ предприятия – начинается с исследования того, как организована система управления предприятием, с обследования функциональной и информационной структуры системы управления, определения существующих и возможных потребителей информации.

По результатам обследования аналитик на первой стадии строит обобщенную логическую модель исходной предметной области, отображающую ее функциональную структуру, особенности основной деятельности и информационное пространство, в котором эта деятельность осуществляется (рисунок 1.4). На этом материале аналитик строит функциональную модель «Как есть» (As Is).

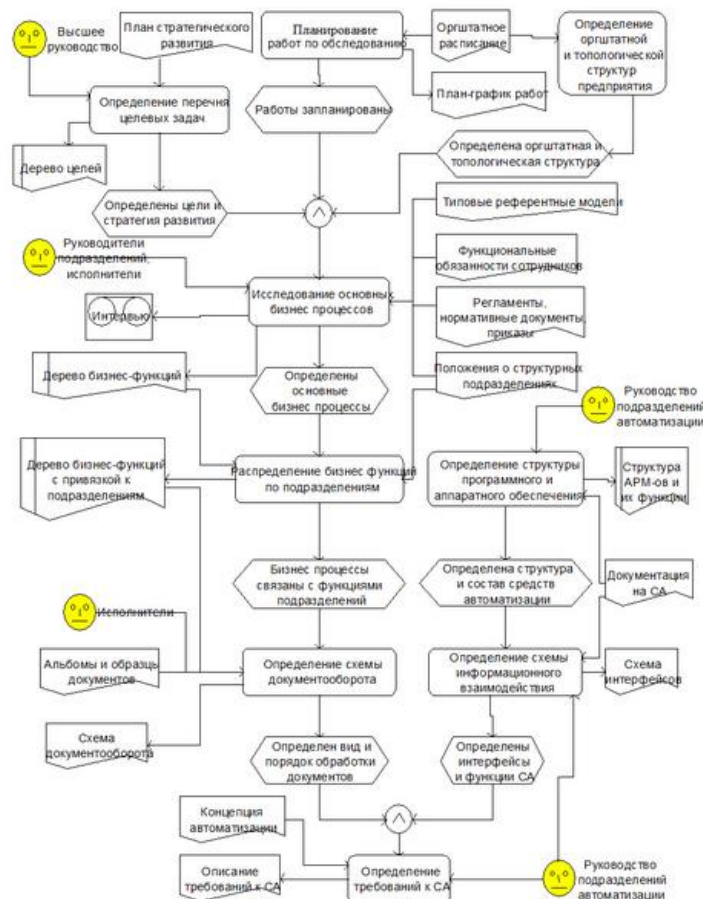


Рисунок 1.4 – Схема обследования предприятия

Вторая стадия работы, к которой обязательно привлекаются заинтересованные представители заказчика, а при необходимости и независимые эксперты, состоит в анализе модели «Как есть», выявлении ее недостатков и узких мест, определение путей совершенствования системы управления на основе выделенных критериев качества.

Третья стадия анализа, содержащая элементы проектирования, – создание усовершенствованной обобщенной логической модели, отображающей реорганизованную предметную область или ее часть, которая подлежит автоматизации – модель «Как должно быть» (As To Be).

Заканчивается процесс (четвертая стадия) разработкой «Карты автоматизации», представляющей собой модель реорганизованной предметной области, на которой обязательно обозначены «границы автоматизации».

В большинстве случаев модель «Как есть» улучшается системным аналитиком за счет устранения очевидных несоответ-

ствий и узких мест, а полученный таким образом вариант модели рассматривается в дальнейшем в качестве предварительной модели «Как должно быть», которая впоследствии дополняется в соответствии со стратегией развития предприятия (рисунок 1.5).



Рисунок 1.5 – Стадии построения модели информационной системы

На стадии анализа требований к проектируемой системе и вводятся:

- классы пользователей и соответствующие диаграммы бизнес-транзакций;
- модели (диаграммы) процессов прикладной деятельности и соответствующие перечни функциональных задач ИС;
- классы объектов предметной области и соответствующие диаграммы «сущность-связь», отражающие информационную модель этой предметной области;
- топология расположения подразделений и пользователей, обслуживаемых данной ИС;
- параметры защиты данных, информации и самой системы.

Основным документом, отражающим результаты работ первого этапа создания ИС, является техническое задание на проект (разработку), содержащее, кроме вышеперечисленных определений и спецификаций, также сведения об очередности создания системы, сведения о выделяемых ресурсах, директивных сроках проведения отдельных этапов работы, организационных процедурах и мероприятиях по приемке этапов, защите проектной информации и т. д.

Следующий этап – проектирование. В реальных условиях проектирование – это поиск, моделирование способа разработки, который удовлетворяет требованиям функциональности системы средствами имеющихся технологий с учетом заданных начальных условий и ограничений. Проектирование информационных систем всегда начинается с определения цели проекта. Основная задача любого успешного проекта заключается в том, чтобы на момент запуска системы и в течение всего времени ее эксплуатации можно было обеспечить:

- требуемую функциональность системы и степень адаптации к изменяющимся условиям ее функционирования;
- требуемую пропускную способность системы и минимальное время реакции системы на запрос;
- безотказную работу системы в требуемом режиме, готовность и доступность системы для обработки запросов пользователей;
- простоту эксплуатации и сопровождения системы;
- необходимую безопасность данных и права доступа пользователей.

Производительность и надёжность являются главными факторами, определяющими эффективность системы. Хорошее проектное решение служит основой высокопроизводительной системы.

Проектирование информационных систем охватывает три основные области:

- проектирование структур данных, которые будут реализованы в базе данных;
- проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;
- проектирование конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры, параллельной обработки, распределенной обработки данных и т. п.

На основе результатов системного анализа на стадии предварительного проекта разрабатываются:

- проект программно-аппаратной реализации, проект пользовательских интерфейсов и технологии работы пользователей в системе;

- архитектура распределенной системы и спецификации телекоммуникационной сети;
- модели (диаграммы) потоков данных;
- функциональные блок-схемы прикладного и системного программного обеспечения (последние – в соответствии с принятыми моделями среды ИС и профилями стандартов).

Стадия предварительного проекта может предусматривать прототипирование фрагментов, важных с точки зрения пользователя для проверки их соответствия требованиям на ранней фазе разработки.

На стадии детального проектирования разрабатываются:

- комплексы функциональных программ ИС и проект реализации среды ИС;
- структуры данных, средства ведения баз данных;
- сетевые адреса, протоколы телекоммуникаций и другие компоненты среды обмена информацией, включаемые в состав проектируемой ИС;
- правила разграничения доступа пользователей и средства их реализации.

Стадия реализации ИС предусматривает разработку и тестирование компонентов и комплексное тестирование системы.

Стадия эксплуатации и сопровождения предусматривает контроль функционирования ИС, внесение требуемых изменений в информационную базу в процессе текущей работы и модернизацию функций ИС силами прикладных специалистов с помощью инструментальных средств, встроенных в систему.

Этапы разработки, тестирования, внедрения, эксплуатации и сопровождения ИС объединяются термином – реализация. Реализация ИС является чрезвычайно сложным многоаспектным процессом, осуществляемым на базе совокупностей (профилей) гармонизированных международных стандартов, спецификаций и соглашений. Такая практика является залогом того, что создаваемая информационная система будет реализована как «открытая система». Иными словами такая ИС будет масштабируема, мобильна, переносима, обладать дружественными интерфейсами и т. д.

Жизненный цикл ИС формируется в соответствии с принципом нисходящего проектирования и, как правило, носит спи-



рально-итерационный характер. Реализованные этапы, начиная с самых ранних, циклически повторяются в соответствии с изменениями требований и внешних условий, введением дополнительных ограничений и т. п. На каждом этапе жизненного цикла порождается определенный набор технических решений и документов, при этом для каждого этапа исходными являются документы и решения, принятые на предыдущем этапе. Жизненный цикл ИС заканчивается, когда прекращается её программное и техническое сопровождение.

### 1.2.2 Реинжиниринг бизнес-процессов

Внедрение информационных технологий и реализованных на их основе информационных систем в повседневную деятельность предприятия дает ему тактические и долгосрочные преимущества в бизнесе. Стремление руководства к использованию ИТ может остаться лишь благими намерениями, если оно не будет следовать сложившимся требованиям и правилам разработки, проектирования и внедрения ИТ. Выше говорилось о базовых требованиях к стандартизации объектов и функциональных задач, без которых реализуемая система не будет являться открытой системой, что приведет впоследствии к многочисленным проблемам при ее внедрении и эксплуатации.

Следование требованиям стандартов при разработке ИС автоматически приводит к тому, чтобы само предприятие – внешняя среда для ИС – также отвечало необходимым требованиям: определение и стандартизация классов пользователей и объектов, топология потоков данных и работ, архитектура наследуемых и разрабатываемых подсистем, состояние бизнес-процессов и т. д.

Бизнес-процесс представляет собой систему последовательных, целенаправленных и регламентированных видов деятельности, в которой посредством управляющего воздействия и с помощью определенных ресурсов за определенное время входы процесса преобразуются в выходы – в результаты, представляющие ценность для потребителя и приносящие прибыль изготовителю.

Стандартный бизнес-процесс в масштабах предприятия реализуется в виде сети основных, вспомогательных, поддерживающих и управленческих процессов (рисунок 1.6).

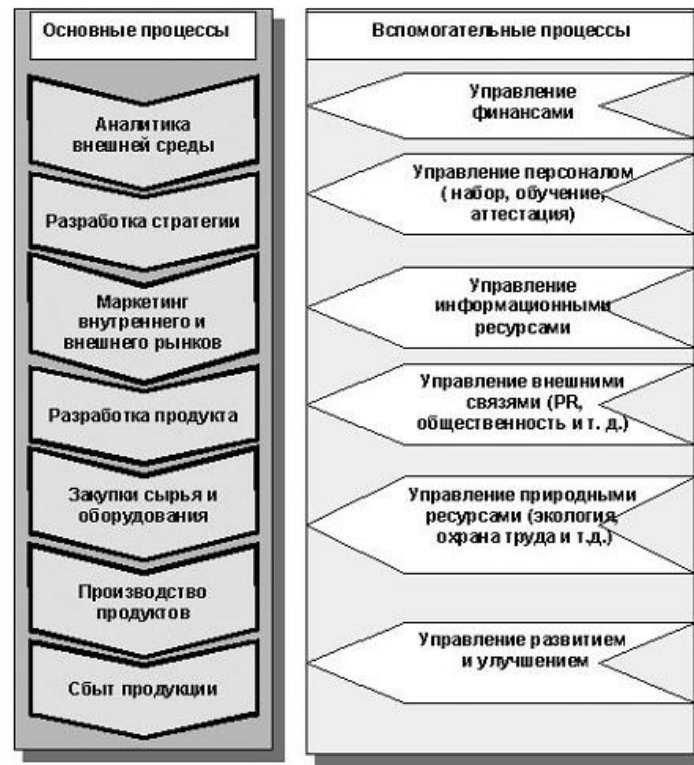


Рисунок 1.6 – Содержание стандартного бизнес-процесса предприятия

При этом разделение на основные и вспомогательные процессы в определяющей степени зависит от предметной области и направления деятельности предприятия: для производственной компании, например, деятельность юридического отдела является вспомогательной, а для юридической или консалтинговой фирмы – основной. Идентификация процессов является обязательным условием, без реализации которого невозможна информатизация деятельности.

Руководители предприятия, решившиеся на внедрение ИТ, должны твердо усвоить – начало работ по проектированию информационной системы чаще всего влечет за собой обязательный реинжиниринг бизнес-процессов! Реинжиниринг представляет собой множество методик и рекомендаций, среди них нужно выбрать те, которые наилучшим образом удовлетворяют поставленным целям.

Реинжиниринг бизнес-процессов – это совокупность методов и действий, служащих для перепроектирования процессов в соответствии с изменившимися условиями внешней и внутренней среды и/или целями бизнеса.

Существует несколько базовых правил, которых следует придерживаться в процессе проведения реинжиниринга:

- разработка последовательных пошаговых процедур для перепроектирования процессов;
- использование в проектировании стандартных языков и нотаций;
- наличие эвристических и прагматических показателей, позволяющих оценить или измерить степень соответствия перепроектированного процесса или функциональности заданным целям;
- подход к решению частных задач и к их совокупности должен быть системным;
- даже небольшое улучшение должно давать быстрый положительный эффект.

Реинжиниринг деловых процессов и функций начинается с пересмотра целей предприятия, его структуры, анализа потребностей внутренних пользователей и рынка, производимых продуктов и услуг (рисунок 1.7).

Перепланирование целей и задач предполагает пересмотр политики предприятия и ответа на следующие вопросы:

Какие новые вызовы предъявляют нам изменившиеся условия бизнеса?

Что представляет предприятие сейчас, и что мы хотим от него в будущем?

Каких именно потребителей мы обслуживаем, насколько мы удовлетворяем их требования и ожидания, и что нужно сделать для привлечения новых?

Какие именно показатели определяют эффективность деятельности предприятия, производительность труда и качество продукта, является ли это определение полным и адекватным?

Какие именно информационные технологии и средства помогут нам в этом?

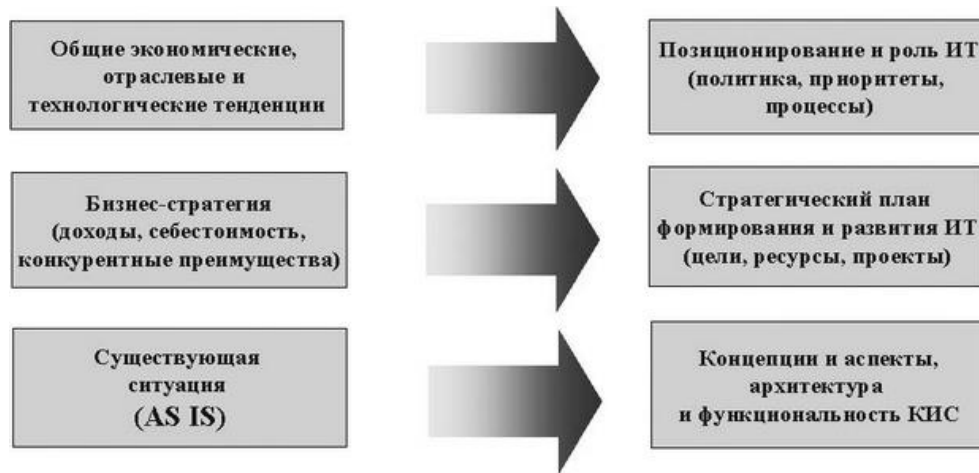


Рисунок 1.7 – Системный подход к реинжинирингу процессов

Для ответа на эти ключевые вопросы необходимо в первую очередь провести детальное описание бизнес-архитектуры предприятия, его бизнес-логики, построить функциональную модель взаимодействия бизнес-процессов, ресурсов и персонала и отразить её в архитектуре ИС, содержании модулей информационных подсистем и визуализации форм представления информации. Необходимо также иметь методики и инструменты реорганизации процессов, решения прикладных задач и управления проектом реинжиниринга (рисунок 1.8). Описание бизнес-архитектуры предприятия позволяет:

- построить схему основных потоков данных, работ, движения финансов и документов;
- понять, как информация распределяется между подразделениями, и кто является конечным пользователем в том или ином бизнес-процессе;
- описать взаимодействие процессов и модулей информационной системы;
- определить критическую важность видов информации для конкретных уровней управления предприятием;
- выявить дублированные структуры и связи.

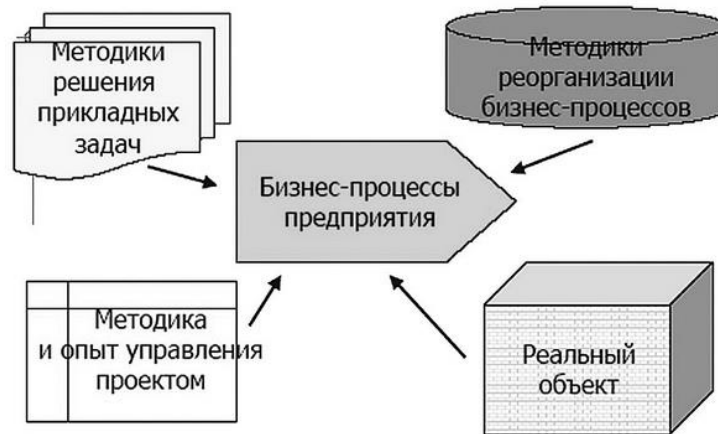


Рисунок 1.8 – Базовая основа улучшения процесса

Результатом такого описания является:

- уточненная карта сети процессов;
- матрица взаимосвязей процессов и подразделений, вовлеченных в эти процессы;
- информация о том, какие системы автоматизации существуют, при выполнении каких операций используются, где и какие данные используются, какие системы автоматизации и информатизации необходимо разработать;
- функциональные схемы потоков данных (Data Flow), работ (Work Flow), финансовых потоков (Cash Flow), потоков управленческих воздействий (Control Flow) и документооборота (Doc Flow).

Функциональная модель поможет составить точные спецификации всех операций, процедур и взаимосвязей между ними. Такая модель, если она построена правильно, обеспечивает исчерпывающее описание о функционирующем процессе и обо всех имеющихся в нем потоках информации. Эта модель описывает состояние «Как есть» (As Is). По результатам анализа возможных путей улучшения от реальной модели нужно перейти к модели, характеризующей улучшения – модель «Как будет» (As To Be), вариант – «Как должно быть» (рисунок 1.9).



Рисунок 1.9 – Схема реинжиниринга бизнес-процесса

Функциональное моделирование является достаточно серьезной проблемой, полнота и соответствие построенной модели зависят как от средств моделирования, так и от квалификации специалистов, выполняющих это моделирование.

Реинжиниринг бизнес-процессов является сложным и многоаспектным проектом, требующим тщательного планирования и проработки деталей. В таблице 1.1 показаны основные этапы реинжиниринга.

Таблица 1.1 – Основные этапы реинжиниринга

Этап	Мероприятия
Планирование начало работ	<p>Выявление главных причин проведения реформы на предприятии и оценка последствий отказа от такой реформы</p> <p>Выявление важнейших процессов, требующих реинжиниринга</p> <p>Выявление единомышленников среди руководства и создание рабочей группы из представителей администрации</p> <p>Обеспечение поддержки проекта руководством</p> <p>Подготовка плана проекта: определение объема, обозначение измеримых целей, выбор методологии, составление подробного графика</p>

Продолжение таблицы 1.1

Этап	Мероприятия
	<p>Согласование целей и объемов проекта с руководством</p> <p>Формирование группы реинжиниринга</p> <p>Выбор консультантов или внешних экспертов</p> <p>Проведение вводного совещания</p> <p>Доведение целей проекта до руководителей низшего звена; начальное информирование всей организации</p> <p>Обучение группы реинжиниринга</p> <p>Подготовка плана и начало работ</p>
Исследования	<p>Аналитическое исследование опыта компаний с подобными процессами</p> <p>Опрос клиентов и контрольных групп для выявления существующих и будущих требований</p> <p>Опрос служащих и руководителей для выявления вопросов; мозговой штурм</p> <p>Поиск в литературе и прессе данных о тенденциях в отрасли и о чужом опыте</p> <p>Оформление подробных документов на исходные процессы и сбор рабочих данных; выявление недоработок</p> <p>Обзор изменений и вариантов технологий</p> <p>Опрос владельцев и представителей руководства</p> <p>Посещение кружков и семинаров</p> <p>Сбор данных от внешних экспертов и консультантов</p>
Проектирование	<p>Мозговой штурм и выработка новаторских идей; упражнения по творческому мышлению, чтобы «снять шоры»</p>

Продолжение таблицы 1.1

Этап	Мероприятия
	<p>Проработка сценариев «а что, если?» и применение «шаблонов успеха» других компаний</p> <p>Создание при помощи специалистов 3-5 моделей; разработка комплексных моделей, в которых собрано лучшее от каждой из предыдущих</p> <p>Создание картины идеального процесса</p> <p>Определение моделей нового процесса и их графическое представление</p> <p>Разработка организационной модели в сочетании с новым процессом</p> <p>Определение технологических требований; выбор платформы для новых процессов</p> <p>Выделение краткосрочных и долгосрочных мер</p>
Утверждение	<p>Анализ затрат и преимуществ; расчет прибыли на капитал</p> <p>Оценка влияния на клиентов и служащих; оценка влияния на конкурентоспособность</p> <p>Подготовка официального документа для высшего руководства</p> <p>Проведение обзорных совещаний для ознакомления и утверждения деталей проекта оргкомитетом и высшим руководством</p>
Внедрение	<p>Завершение подробной разработки процессов и организационных моделей; определение новых рабочих обязанностей</p> <p>Разработка систем поддержки</p>



Продолжение таблицы 1.1

Этап	Мероприятия
	Реализация предварительных вариантов и первичные испытания Ознакомление работников с новым вариантом; разработка и осуществление плана реформы Разработка поэтапного плана; внедрение как таковое Разработка плана обучения; обучение работников новым процессам и системам
Последующие мероприятия	Разработка мероприятий по периодической оценке; определение итогов нового процесса; внедрение программы непрерывного совершенствования нового процесса Предоставление окончательного отчета оргкомитету и администрации

### 1.2.3 Отображение и моделирование процессов

На сегодняшний день получили распространение три основные методологии функционального моделирования (и сопутствующий им инструментарий): IDEF (Integrated DEFinition), UML (Unified Modeling Language) и ARIS (Architecture of Integrated Information Systems). Для каждой из них существуют определенные программные продукты, которые помимо разработки позволяют проводить преобразования и операции для последующей работы с полученными моделями. Наибольшее распространение сегодня получили методологии IDEF и программный продукт BPWin, содержащий методологии IDEF0, IDEF3, DFD (Data Flow Diagrams) и ERWin (IDEF1x) от компании Computer Associates.

История методологии IDEF начинается с 70-х годов XX века с методологии SADT (Structured Analysis and Design Technique), разработанной Дугласом Россом (Softtech INC). Изначально

SADT применялось Министерством Обороны США для практического моделирования процессов в рамках программы ICAM (Integrated Computer Aided Manufacturing). Принципиальным требованием при разработке рассматриваемого семейства методологий была возможность эффективного обмена информацией между всеми специалистами – участниками программы ICAM (Icam DEFinition). В последующем эта методология была трансформирована в стандарт IDEF0 (Function Modeling, FIPS № 183). Семейство IDEF включает уже упомянутые IDEF3 (Process Description Capture) и IDEF1x (Data Modeling, FIPS № 184).

После опубликования стандартов они были успешно применены в самых различных областях бизнеса, показав себя эффективным средством анализа, конструирования и отображения бизнес-процессов (к слову сказать, он активно применяется и в отечественных госструктурах, например в Государственной налоговой инспекции). Более того, собственно с широким применением IDEF (и предшествующей методологии SADT) и связано возникновение основных идей популярного ныне понятия «реинжиниринг бизнес-процессов» (Business Process Reengineering – BPR).

Информационный процесс – это устойчивый процесс (последовательность работ и действий с данными и информацией), относящийся к сопровождению производственно-хозяйственной деятельности компании и обычно ориентированный на информационное обслуживание создания новой стоимости. Бизнес-процесс включает в себя иерархию взаимосвязанных функциональных действий, реализующих одну (или несколько) бизнес-целей компании и отражающий результаты в информационной системе, например, информационное обеспечение управления и анализа выпуска продукции или ресурсное обеспечение выпуска продукции (под продукцией здесь понимают товары, услуги, решения, документы).

Работа с использованием метода IDEF начинается с постановки цели моделирования. Мировой опыт свидетельствует, что ошибки при постановке цели приводят в среднем к 50 % неудач в процессе моделирования. Формулирование цели изначально направляет работу в заданном направлении, а значит, ограничивает круг вопросов для анализа. Практическая работа начинается с определения контекста (Context, Context Diagram), то есть верх-

него уровня системы, в нашем случае – предприятия. После формулировки цели необходимо очертить область моделирования (Score), которая в последующем будет определять общие направления движения и глубину детализации (Decomposition). Собственно, сама методология IDEF определяет стандартизированные объекты для работы и отображения. Например, к таковым относятся функция (Activity), интерфейсная дуга (Arrow), заметка (Note) а также способ их расположения и трактования (Semantics).

В последнее время на российском рынке появился программный продукт Business Studio, который специально создан для работы с методами IDEF и обладает интуитивным и дружелюбным интерфейсом (User-friendly Interface).

В основе нотации и методологии IDEF0 лежит понятие «блока», то есть прямоугольника, который выражает некоторую функцию бизнеса (рисунок 1.10). В соответствии со стандартом функция должна быть выражена глагольным оборотом В IDEF0 роли сторон прямоугольника (функциональные значения) различны: верхняя сторона имеет значение «управление», левая – «вход», правая – «выход», нижняя – «механизм исполнения».

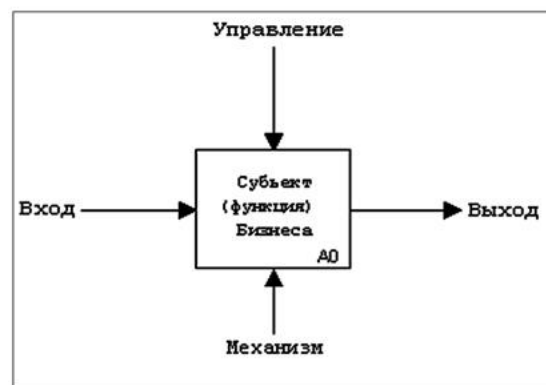


Рисунок 1.10 – Базовый блок методологии IDEF0

Вторым элементом методологии и нотации является «поток», называемый в стандарте «интерфейсная дуга». Это элемент, описывающий данные, неформальное управление, или что-либо другое, оказывающее влияние на функцию, изображенную блоком. Потoki обозначаются оборотом существительного.

В зависимости от того, к какой стороне блока направлен поток, он, соответственно, носит название «входной», «выходной»,

«управляющий». Изобразительным элементом, представляющим поток, является стрелка. Поток можно интерпретировать как представление объекта, под которым понимается как информационный объект, так и реальный физический объект.

Важным фактором является то, что «источником» и «приемником» потоков (то есть, началом и концом стрелки) могут быть, как правило, только блоки. При этом источником может являться только выходная сторона блока, приемником – любая из трех оставшихся. Если же необходимо подчеркнуть внешний характер потока, то может быть применен метод «туннелирования» – скрывание или появление интерфейсной дуги из «туннеля».

И, наконец, «третьим китом» методологии IDEF0 является принцип функциональной декомпозиции блоков, который представляет собой модельную интерпретацию той практической ситуации, что любое действие (тем более такое сложное, как бизнес-процесс) может быть разбито (декомпозировано) на более простые операции (действия, бизнес-функции). Или, другими словами, действие может быть представлено как совокупность элементарных функций.

Пример функциональной модели процесса отгрузки и доставки продукции показан на рисунок 1.11.

Степень формализации описания бизнес-процессов может быть различной в зависимости от решаемых при этом задач. Для описания информационных процессов разработан специализированный язык BPEL (Business Process Execution Language). BPEL создан на основе XML для формального описания бизнес-процессов и протоколов их взаимодействия между собой. BPEL расширяет модель взаимодействия Web-служб и включает в эту модель поддержку транзакций.

В настоящее время активно развивается методология BPMS (Business Process Management System) – класс программного обеспечения для управления бизнес-процессами и административными регламентами. (Употребляются также термины BPM-система и просто BPM). Использование BPMS позволяет организовать эффективное взаимодействие между управленцами и ИТ-специалистами, лучше использовать существующие подсистемы и ускорить разработку новых.

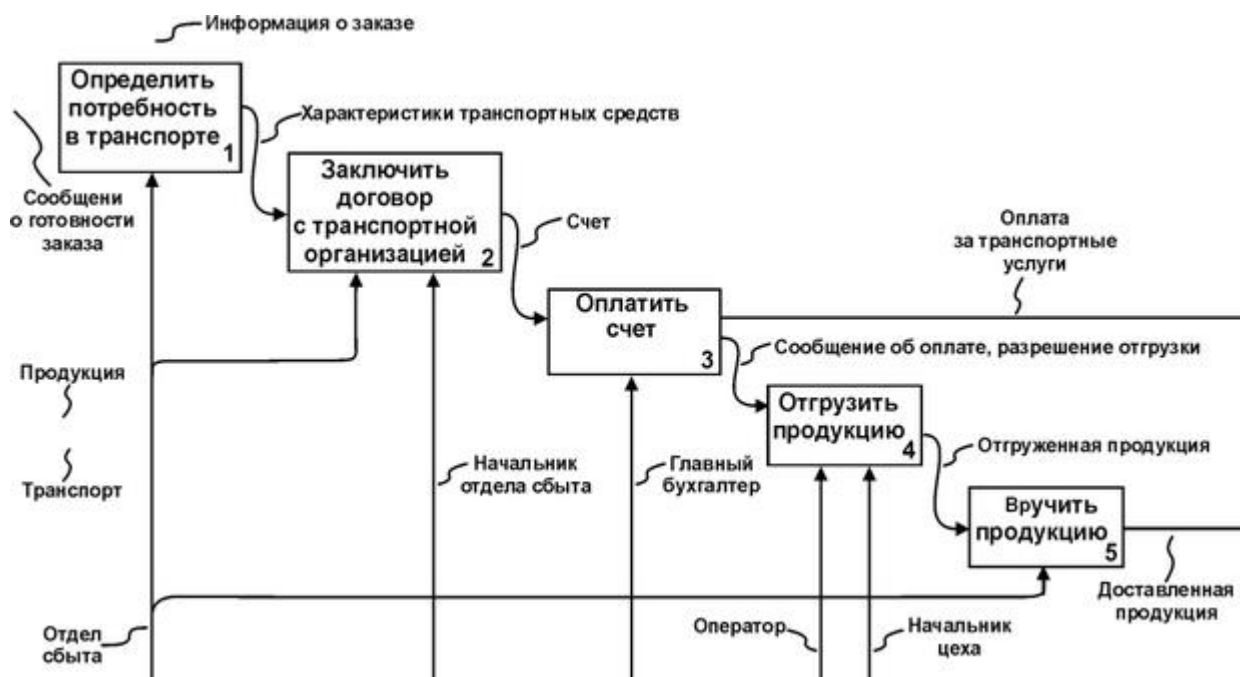


Рисунок 1.11 – Пример функциональной модели процесса отгрузки и доставки

Основные функции BPMS – моделирование, исполнение и мониторинг бизнес-процессов. Основываясь на данных мониторинга, предприятия выявляют узкие места и усовершенствуют свои бизнес-процессы. Цикл управления замыкается, когда при помощи BPMS измененные бизнес-процессы оперативно внедряются в эксплуатацию.

Современные методы разработки и развития программного обеспечения ИС в полной мере стараются ориентироваться на возможности автоматизированного оперативного внесения изменений. Наиболее сложным оказался процесс стандартизации языка BPEL для унификации использования одних и тех же конструкций программным обеспечением разных производителей. Фирмы IBM и Microsoft определили два довольно-таки схожих языка: WSFL (Web Services Flow Language) и Xlang, соответственно.

Рост популярности BPML и открытое движение BPMS к пользователям привело корпорации Intalio Inc., IBM и Microsoft к решению объединить эти языки в новый язык BPEL4WS. В апреле 2003 года корпорации BEA Systems, IBM, Microsoft, SAP и Siebel Systems передали BPEL4WS версии 1.1 в OASIS для стандартизации в Web Services BPEL Technical Committee. Хотя BPEL4WS появился в версиях 1.0 и 1.1, технический комитет

WS-BPEL OASIS проголосовал 14 сентября 2004 за то, чтобы назвать спецификацию WS-BPEL 2.0. Это изменение было сделано, чтобы согласовать BPEL с другими стандартами Web-сервисов, которые на основании «Соглашения об именовании» начинаются сочетаниями букв «WS-».

Корпорации Active Endpoints, Adobe, BEA, IBM, Oracle и SAP опубликовали согласованные спецификации BPEL4 People и WS-HumanTask, в которых описывалось, как может быть реализовано в системе и нотациях BPEL взаимодействие процессов с людьми. Предполагается добавление в BPEL семантики в форме WS-HumanTask и других разнообразных дополнений.

#### 1.2.4 Обеспечение процесса анализа и проектирования ИС возможностями CASE-технологий

Термин CASE (Computer Aided Software/System Engineering) используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения (ПО), в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных ИС в целом.

Теперь под термином CASE-средства понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного программного обеспечения (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. Таким образом, современные CASE-средства вместе с системным программным обеспечением и техническими средствами поддержки образуют полную среду разработки ИС.

Появлению CASE-технологии и CASE-средств предшествовали исследования в области методологии программирования. Программирование обрело черты системного подхода с разработкой и внедрением языков высокого уровня, методов структурного и модульного программирования, средств визуального моделирования и проектирования на базе языка UML (Unified Modeling Language), средств их поддержки, формальных и не-

формальных языков описаний системных требований и спецификаций и т. д. Кроме того, появлению CASE-технологии способствовали и такие факторы, как:

- подготовка аналитиков и программистов, восприимчивых к концепциям модульного и структурного программирования;
- широкое внедрение и постоянный рост производительности компьютеров, позволившие использовать эффективные графические средства и автоматизировать большинство этапов проектирования;
- внедрение сетевой технологии, предоставившей возможность объединения усилий отдельных исполнителей в единый процесс проектирования путем использования разделяемой базы данных, содержащей необходимую информацию о проекте.

CASE-технология представляет собой методологию проектирования ИС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

CASE-средства позволяют создавать не только продукт, практически готовый к использованию, но и обеспечить «правильный» процесс его разработки. Основная цель технологии — отделить проектирование программного обеспечения от его кодирования, сборки, тестирования и максимально «скрыть» от будущих пользователей все детали разработки и функционирования ПО. При этом значительно повышается эффективность работы проектировщика: сокращается время разработки, уменьшается число программных ошибок, программные модули можно использовать при следующих разработках.

Большинство CASE-средств основано на парадигме «методология/метод/нотация/структура/средство».

Методология задает руководящие указания для оценки и выбора проекта разработки ПО, этапы и последовательность работ, правила применения тех или иных методов.

Метод – систематическая процедура или технология генерации описаний компонент ПО (например, описание потоков и структур данных).

Нотации предназначены для описания системы в целом, ее элементов: графы, диаграммы, таблица, блок схемы, алгоритмы, формальные языки и языки программирования.

Структуры являются средством для реализации структурного анализа и построения структуры конкретной системы.

Средства – технологические и программные инструменты для поддержки и усиления методов.

CASE-технологии обладают следующими основными достоинствами, которые позволяют широко использовать их при разработке информационных систем:

- ускоряют процесс коллективного проектирования и разработки;
- позволяют за короткий срок создать прототип заказанной системы с заданными свойствами;
- освобождают разработчика от рутинной работы, оставляя время для творчества;
- обеспечивают эффективность и качество разрабатываемого ПО за счет автоматизации контроля всего процесса разработки;
- поддерживают сопровождение и развитие системы на высоком уровне.

Следует отметить, что, несмотря на все потенциальные возможности CASE-средств, существует достаточно много примеров их неудачного внедрения, в результате которых CASE-средства становятся «полочным» ПО (Shelfware).

В связи с этим необходимо учитывать следующее:

- CASE-средства не обязательно дают немедленный эффект, он может быть получен только спустя какое-то время;
- реальные затраты на внедрение CASE-средств обычно намного превышают затраты на их приобретение;
- CASE-средства обеспечивают возможности для получения существенной выгоды только после успешного завершения



процесса их внедрения, эффективного обучения пользователей и регулярного применения.

Можно также перечислить следующие факторы, усложняющие определение возможного эффекта от использования CASE-средств:

- широкое разнообразие качества и возможностей CASE-средств;
- относительно небольшое время использования CASE-средств в различных организациях и недостаток опыта их применения;
- широкое разнообразие в практике внедрения различных организаций;
- отсутствие детальных метрик и данных для уже выполненных и текущих проектов;
- широкий диапазон предметных областей проектов;
- различная степень интеграции CASE-средств в различных проектах.

Некоторые аналитики считают, что реальная выгода от использования некоторых типов CASE-средств может быть получена только после одно- или двухлетнего опыта. Другие полагают, что воздействие может реально проявиться в фазе эксплуатации жизненного цикла ИС, когда технологические улучшения могут привести к снижению эксплуатационных затрат.

Ниже перечислены основные виды и последовательность работ, рекомендуемые при построении логических моделей предметной области в рамках CASE-технологии анализа системы управления предприятием.

Проведение функционального и информационного обследования системы управления (административно-управленческой деятельности) предприятия:

- определение организационно-штатной структуры предприятия;
- определение функциональной структуры предприятия;
- определение перечня целевых функций структурных элементов (подразделений и должностных лиц);
- определение круга и очередности обследования структурных элементов системы управления согласно сформулированным целевым функциям;

- обследование деятельности выделенных структурных элементов;
- построение FD-диаграммы системы управления с указанием структурных элементов и функций, реализация которых будет моделироваться на DFD-уровне.

Разработка моделей деятельности структурных элементов и системы управления в целом:

- выделение множества внешних объектов, оказывающих существенное влияние на деятельность структурного элемента;
- спецификация входных и выходных информационных потоков;
- выявление основных процессов, определяющих деятельность структурного элемента и обеспечивающих реализацию его целевых функций;
- спецификация информационных потоков между основными процессами деятельности, уточнение связей между процессами и внешними объектами;
- оценка объемов, интенсивности и других необходимых характеристик информационных потоков;
- разработка иерархии диаграмм потоков данных, образующих функциональную модель деятельности структурного элемента;
- объединение DFD-моделей структурных элементов в единую модель системы управления предприятием.

Разработка информационных моделей структурных элементов и модели информационного пространства системы управления:

- определение сущностей модели и их атрибутов;
- проведение атрибутивного анализа и оптимизация сущностей;
- идентификация отношений между сущностями и определение типов отношений;
- анализ и оптимизация информационной модели;
- объединение информационных моделей в единую модель информационного пространства.

Разработка предложений по автоматизации системы управления предприятия:

- определение границ автоматизации – составление перечня автоматизируемых структурных элементов, разбиение процессов основной деятельности на автоматические, автоматизированные и ручные;
- составление перечня подсистем и логических АРМов (автоматизированных рабочих мест), определение способов их взаимодействия;
- разработка предложений по очередности проектирования и реализации подсистем и отдельных логических АРМов, входящих в состав ИС;
- разработка требований к средствам базового технического обеспечения ИС;
- разработка требований к средствам базового программного обеспечения ИС.

Логическая модель, отображающая деятельность системы управления предприятия и информационное пространство, в котором эта деятельность протекает, представляют собой «снимок» положения дел (функциональная структура, роли должностных лиц, взаимодействие подразделений, принятые технологии обработки управленческой информации, автоматизированные и неавтоматизированные процессы и т. д.) на момент обследования. Эта модель позволяет понять, что делает и как функционирует предприятие с позиций системного анализа, сформулировать предложения по улучшению ситуации.

Развитие логической модели предметной области, ее последовательное превращение в модель целевой ИС, позволит интегрировать перспективные предложения руководства и ведущих сотрудников предприятия, экспертов и системных аналитиков, сформировать видение новой, реорганизованной и автоматизированной деятельности предприятия (рисунок 1.12).

Построенная модель является законченным результатом по следующим причинам.

Она включает в себя модель существующей неавтоматизированной технологии, принятой на предприятии. Формальный анализ этой модели позволяет выявить узкие места в управлении предприятием и сформулировать рекомендации по его улучшению (независимо от того, предполагается ли дальнейшая разработка автоматизированной системы или нет).



Рисунок 1.12 – Модель системы в технологическом CASE-решении

Она независима и отделяема от конкретных разработчиков, не требует сопровождения и может быть безболезненно передана другим лицам. Более того, если по каким-либо причинам предприятие не готово к реализации проекта в данный момент времени, модель может быть «положена на полку» до тех пор, пока в ней не возникнет необходимость.

Она позволяет осуществлять эффективное обучение новых работников конкретным направлениям деятельности предприятия, так как соответствующие технологии содержатся в модели.

С ее помощью можно осуществлять предварительное моделирование перспективных направлений деятельности предприятия с целью выявления новых потоков данных, взаимодействующих процессов и структурных элементов.

Она обеспечивает распространение накопленного опыта на других предприятиях, дает возможность унифицировать административно-управленческую и финансовую деятельность этих предприятий.

Модель является не просто реализацией начальных этапов работы и основанием для формирования технического задания на ее последующие этапы. Она представляет собой самостоятельный результат, имеющий большое практическое значение, так как

позволяет дальнейшее применение CASE-технологий для реального проектирования и разработки ИС.

Современные CASE-пакеты имеют широкие возможности инструментального расширения за счёт использования стандартных программных средств, что делает их чрезвычайно удобными при разработке программных и информационных систем (рисунки 1.13 и 1.14).

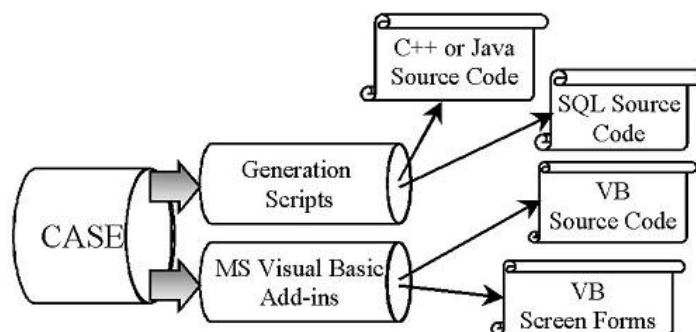


Рисунок 1.13

Для успешного внедрения CASE-средств организация должна обладать следующими качествами:

**Культура.** Готовность к внедрению новых процессов и взаимоотношений между разработчиками и пользователями, ИТ/ИС-управленцами и пользователями.

**Управление.** Четкое руководство и организованность по отношению к наиболее важным этапам и процессам внедрения.

**Технология.** Понимание ограниченности существующих возможностей и способность принять новую технологию.

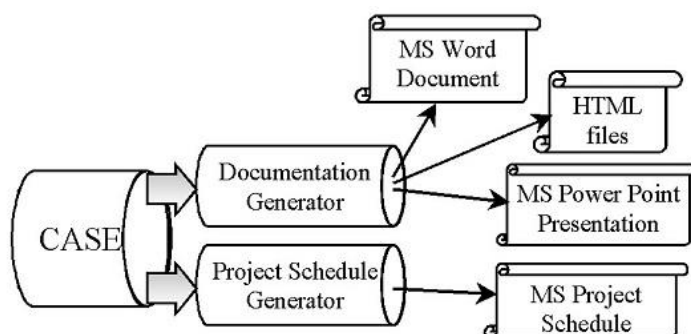


Рисунок 1.14

Если организация не обладает хотя бы одним из перечисленных качеств, то внедрение CASE-средств может закончиться

неудачей независимо от степени тщательности следования различным рекомендациям по внедрению.

В качестве примеров популярных CASE-средств укажем программные средства компании Computer Associates, IBM-Rational Software и Oracle:

- BPwin – моделирование бизнес-процессов;
- ERwin – моделирование баз данных и хранилищ данных;
- ERwin Examiner – проверка структуры СУБД и моделей, созданных в Erwin;
- ModelMart – среда для командной работы проектировщиков;
- Paradigm Plus – моделирование приложений и генерация объектного кода;
- Rational Rose – моделирование бизнес-процессов и компонентов приложений
- Rational Suite AnalystStudio – пакет для аналитиков данных;
- Oracle Designer (входит в Oracle9i Developer Suite) – высоко функциональное средство проектирования программных систем и баз данных, реализующее технологию CASE и собственную методологию Oracle – CDM. Позволяет команде разработчиков полностью провести проект, начиная от анализа бизнес-процессов через моделирование к генерации кода и получению прототипа, а в дальнейшем и окончательного продукта. Сложное CASE-средство, имеет смысл использовать при ориентации на линейку продуктов Oracle.

Самым мощным из указанных программных пакетов является пакет Rational Rose (RR) компании IBM-Rational, с помощью которого можно спроектировать и сопровождать весь жизненный цикл разработки программного продукта (рисунок 1.15).

Пакет включает набор средств моделирования объектно-ориентированных информационных систем, базирующихся на языке моделирования UML. RR способен решать практически любые задачи в проектировании информационных систем: от анализа бизнес-процессов до кодогенерации на определенном языке программирования, позволяет разрабатывать как высокоуровневые, так и низкоуровневые модели, осуществляя тем са-

мым абстрактное либо логическое проектирование (рисунок 1.16).



Рисунок 1.15 – Состав CASE-средства IBM-Rational

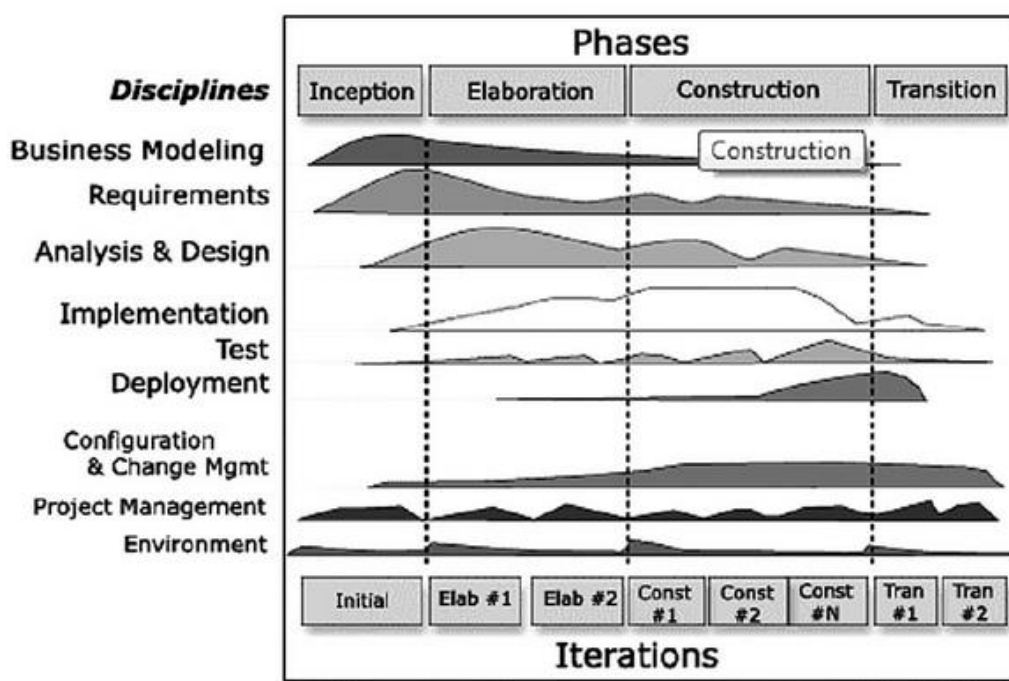


Рисунок 1.16 – Сопровождение ЖЦ программного продукта с RR

### 1.2.5 Внедрение информационных систем

Внедрение корпоративной ИС, разработанной самостоятельно или приобретенной у поставщика, зачастую сопровождается ломкой (перепроектированием) существующих на предприятии бизнес-процессов. Приходится перестраивать их под требования стандартов и логику внедряемой системы. Отметим сразу, что внедрение ИС решает ряд управленческих и технических

проблем, однако порождает проблемы, связанные с человеческим фактором.

Внедрение информационной системы, как правило, значительно облегчает управление деятельностью предприятия, оптимизирует внутренние и внешние потоки информации, ликвидирует узкие места в управлении. Однако после того как система успешно установлена, «обкатана» в работе и показала свою эффективность, у части сотрудников выявляется нежелание использовать ИС в работе. В результате проведённого реинжиниринга становится ясно, что некоторые сотрудники в большой степени дублируют работу других или вовсе не нужна. Кроме того, внедрение КИС сопровождается обязательным обучением, но, как показывает российский опыт, желающих переучиваться не так много. Ломка старых навыков и прививание новых – долгий и трудный процесс!

Надо четко понимать, что корпоративная ИС призвана упростить управление организацией, улучшить процессы, усилить контроль и обеспечить этим конкурентные выгоды. Только с такой точки зрения можно оценивать пользу от её внедрения.

Следуя этой логике, становится понятно, что хотя корпоративная ИС предназначена в целом для обеспечения всех пользователей необходимой информацией, управление разработкой и внедрением КИС является прерогативой высшего руководства компании! Понимают ли это руководители?

Здесь тоже приходится бороться с живучими стереотипами. «Зачем мне корпоративная система, если дела на предприятии и так идут хорошо?». «Зачем, что-то ломать, если все работает?». Но ведь ломать-то чаще всего и не надо. На первом этапе нужно лишь грамотно и корректно формализовать и перенести идентифицированные процессы, в рамках которых живет предприятие, в корпоративную ИС. Подобная формализация лишь отточит, отшлифует удачные маркетинговые и производственные находки, оптимизирует процесс управления и контроля и позволит в дальнейшем проводить целенаправленные изменения.

Внедрение новой ИС – сложный процесс, длящийся от нескольких месяцев для небольших ИС до нескольких лет для ИС больших распределенных компаний с широкой номенклатурой продуктов и большим количеством поставщиков. Успех проекта



по разработке (приобретению) и внедрению ИС во многом зависит от готовности предприятия к ведению проекта, личной заинтересованности и воли руководства, реальной программы действий, наличия ресурсов, обученного персонала, способности к преодолению сопротивления на всех уровнях сложившейся организации.

К настоящему времени сложился стандартный набор приемов внедрения ИС. Основное правило: выполнять обязательные фазы последовательно и не пропускать ни одной из них.

Критически важными для внедрения являются следующие факторы:

- наличие четко сформулированных целей проекта и требований к ИС;
- наличие стратегии внедрения и использования ИС;
- проведение предпроектного обследования предприятия и построения моделей «Как есть» и «Как будет»;
- планирование работ, ресурсов и контроль выполнения плана внедрения;
- участие высшего руководства во внедрении системы;
- проведение работ по внедрению ИС специалистами по интегрированию систем совместно со специалистами предприятия;
- регулярный мониторинг качества выполняемых работ;
- быстрое получение положительных результатов хотя бы в части внедренных модулей ИС или в процессе её опытной эксплуатации.

Перед началом разработки проекта внедрения необходимо:

- максимально формализовать цели проекта внедрения ИС;
- оценить минимально необходимые затраты и статьи расхода;
- установить высокий приоритет проекта внедрения перед остальными текущими проектами;
- наделить руководителя проекта максимально возможными полномочиями;
- провести массовую просветительскую работу с персоналом предприятия с целью довести до каждого важность и необходимость предстоящих преобразований;

- разработать организационные меры для применения новых информационных технологий;
- распределить персональную ответственность по всем этапам внедрения и опытной эксплуатации.

Необходимо также определить функциональные сферы внедрения модулей информационной системы:

- организационное управление;
- организационно-административное обеспечение;
- управление бизнес-процессами;
- управленческий, планово-финансовый и бухгалтерский учет;
- управление персоналом;
- управление документацией;
- управление материально-техническим обеспечением;
- управление связями с клиентами и внешней средой.

Кроме того, что перечислено выше, надо задать технологические требования к внедрению ИС:

- системная платформа: внедрение и адаптация готового решения от производителя или разработка на заказ в соответствии с техническим заданием заказчика.
- интегрируемость: данные хранятся и обрабатываются в едином информационном пространстве – это обеспечивает их полноту, непротиворечивость, достоверность и возможность многократного использования; система может включать в себя вновь разработанные и уже используемые технологии и приложения.
- адаптируемость: система настраивается в соответствии с требованиями заказчика и на особенности информационного поля заказчика.
- распределенность: система может эффективно функционировать в территориально удаленных подразделениях и филиалах предприятия.
- масштабируемость: система может выполняться в виде каркаса, содержащего базовые модули, и дополняться в соответствии с требованиями изменяющейся внешней и внутренней среды.

Основные фазы внедрения информационной системы

Фаза «Предварительные работы по подготовке проекта внедрения ИС». В ходе предпроектного обследования предприятия собирается подробная информация о структурном построении организации, функциональных связях, системе управления, об основных бизнес-процессах, о потоках внутри предприятия (Control Flow, Doc Flow, Data Flow, Work Flow, Cash Flow), необходимая для построения соответствующих моделей и выбора объектов для автоматизации. Оцениваются сроки, ресурсы, виды и объемы работ, номенклатура и стоимость программно-аппаратных и телекоммуникационных средств, стоимость обучения персонала и т. д.

Фаза «Подготовка проекта». После завершения первой фазы осуществляется предварительное планирование и формирование процедур запуска проекта:

- формирование проектной и экспертной групп;
- распределение полномочий и ответственности;
- определение организационно-технических требований к процессу внедрения;
- уточнение спецификаций и ожиданий заказчика;
- обучение группы внедрения, состоящей из специалистов предприятия-заказчика.

Последний очень важный момент почему-то часто пропускается при составлении плана внедрения. А ведь от него в огромной степени зависит успех всего проекта! После начала финансирования проект считается запущенным к исполнению.

Фаза «Концептуальная проработка проекта». В течение этой фазы:

- формируется и утверждается концептуальный проект;
- достигается обязательное однозначное понимание намерений всех участников проекта относительно внедряемой ИС;
- уточняются и конкретизируются цели и задачи проекта;
- определяются размеры прототипа системы;
- согласуются укрупненный план работы, последовательность этапов и условия опытной эксплуатации, планово-финансовые и отчетные показатели.

При этом все указанные действия в обязательном порядке документируются, согласуются и утверждаются всеми заинтересованными и ответственными сторонами.

Фаза «Реализация проекта». Во время проведения основных работ по внедрению создается, устанавливается и конфигурируется системная среда, определяются процедуры системного администрирования, устанавливаются основные программно-аппаратные комплексы и приложения. В системе настраиваются организационно-штатные и организационно-функциональные структуры предприятия с использованием таких организационных единиц, как филиал, департамент, отдел, рабочая группа и т. д.

Осуществляется установка, конфигурирование и настройка сетевых и телекоммуникационных средств, производится перенос данных из прежних локальных систем и формирование интерфейсов с унаследованными и внешними системами. При этом все создаваемые модели, планы, рабочие программные продукты, документация помещаются в сквозной репозиторий проекта внедрения (рисунок 1.17). Важной частью этого репозитория является система документации, формируемая в рамках проекта (рисунок 1.18).



Рисунок 1.17 – Примерное содержание репозитория проекта внедрения



Рисунок 1.18 – Примерный состав документации по процессу внедрения ИС

Отрабатываются системные вопросы безопасности работы системы в многопользовательском режиме. Создаются приложения, шаблоны, отчеты, клиентские формы доступа, распределяются полномочия пользователей. Проводится «прогонка» всех систем в «боевом режиме» с участием всех заинтересованных сторон. После окончания фазы реализации проект внедрения считается законченным. Информационная система передается в эксплуатацию.

Осуществляется установка, конфигурирование и настройка сетевых и телекоммуникационных средств, производится перенос данных из прежних локальных систем и формирование интерфейсов с унаследованными и внешними системами. При этом все создаваемые модели, планы, рабочие программные продукты, документация помещаются в сквозной репозиторий проекта внедрения (рисунок 1.17). Важной частью этого репозитория является система документации, формируемая в рамках проекта (рисунок 1.18).

Отрабатываются системные вопросы безопасности работы системы в многопользовательском режиме. Создаются приложения, шаблоны, отчеты, клиентские формы доступа, распределяются полномочия пользователей. Проводится «прогонка» всех систем в «боевом режиме» с участием всех заинтересованных сторон.

После окончания фазы реализации проект внедрения считается законченным. Информационная система передается в эксплуатацию.

### 1.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Данное практическое занятие предполагает выполнение следующих этапов:

- изучить методические указания;
- ответить на контрольные вопросы.

### 1.4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое «открытая информационная система»?
2. Перечислите основные свойства открытых систем.
3. Охарактеризуйте суть современного процессного подхода к управлению деятельностью предприятия и использования этого подхода при разработке ИС.
4. Что включает в себя понятие «Реинжиниринг бизнес-процессов»?
5. Какие модели и каким образом используются при проектировании информационных систем?
6. Какие программные средства используются для моделирования процессов при разработке информационных систем?
7. На основании, каких данных и информации разрабатываются модели состояния AS IS и AS TO BE?
8. Кто в компании занимается вопросами разработки, внедрения и развития ИС? Кто участвует в подготовке технического задания на разработку ИС?
9. Назовите основные этапы проектирования информационных технологий.
10. Перечислите этапы жизненного цикла информационной системы.
11. На каком этапе разработки и внедрения ИС производится обучение персонала компании?
12. Перечислите основные фазы внедрения ИС.

## 2 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ НА ВНЕДРЕНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

### 2.1 ЦЕЛЬ РАБОТЫ

Цель работы – изучение теоретических основ, связанных с разработкой технического задания и получение практических навыков по разработке технического задания.

### 2.2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Техническое задание (ТЗ) в дальнейшем является основным документом, по которому ведут разработку проекта. Любые изменения ТЗ на систему должны быть согласованы и заверены.

Техническое задание состоит из трех частей:

- содержание задания, в котором перечисляются все основные составные этапы выполнения проекта;
- исходные данные;
- календарный план выполнения работ.

Часть 1 – Содержание задания. Данная часть ТЗ фиксирована, в ней перечислены основные составные этапы выполнения проекта. При разработке программной используются две основные методологии: объектно-ориентированного анализа и проектирования (ООАП) и методология структурного проектирования.

ООАП (Object-Oriented Analysis/Design) – технология разработки программных систем, в основу которых положена объектно-ориентированная методология представления предметной области в виде объектов, являющихся экземплярами соответствующих классов. Методология ООАП тесно связана с концепцией автоматизированной разработки программного обеспечения (Computer Aided Software Engineering, CASE) и языком моделирования UML (Unified Modeling Language).

Методология структурного проектирования применяется на первой фазе проектирования при разделении системы на подсистемы, здесь используется принцип проектирования «сверху вниз».

Часть 2 – Исходные данные включает в себя следующие подразделы:

- характеристики объекта автоматизации (или управления);
- требования к информационному обеспечению;
- требования к техническому обеспечению;
- требования к программному обеспечению;
- общие требования к проектируемой систем;
- перечень дополнительных работ (если необходимо).

Характеристики объекта автоматизации. Здесь указываются общие характеристики объекта автоматизации, характерные для рассматриваемой предметной области:

- полное название объекта;
- условия его функционирования;
- количественные и качественные показатели объекта, которые являются ограничениями процесса функционирования.

В качестве примера рассмотрим проект «Автоматизированная система составления и разгадывания линейного кроссворда по выбранной теме».

Понятие «объект автоматизации» в явном виде в ГОСТ 34.602-89 нигде не определено, но в пункте 2.4.1 указано: «В подразделе Назначение системы указывают вид автоматизируемой деятельности (управление, проектирование и т. п.) и перечень объектов автоматизации (объектов), на которых предполагается ее использовать». Из него следует, что под «объектами автоматизации» авторы понимали вовсе не процессы. Будем считать, что объектом автоматизации может быть только материальный (интеллектуальный) объект организация, магазин, цех, отдел, и так далее.

Из названия темы следует, что в данном случае объект автоматизации – это линейный кроссворд, а виды автоматизируемой деятельности – это процессы:

- составления/ генерирования кроссворда;
- разгадывания кроссворда;
- работы со словарем понятий.

Составление кроссворда отличается от генерирования: в первом случае пользователь вручную составляет кроссворд (добавляет понятия, удаляет понятия и т.п.), во втором (генерирование) кроссворд составляется системой автоматически в соответствии с теми настройками, которые выполнил пользователь. Про-



цесс составления во многом дублирует функции, которые будет выполнять пользователь при редактировании кроссворда, поэтому процесс редактирования кроссворда не нужно выделять отдельно.

Для каждого процесса в ТЗ должны быть указаны количественные показатели-ограничения, для того, чтобы их правильно выбрать, необходимо знать начальные сведения о структуре самого объекта, каким образом будет проходить его построение и разгадывание. Отправной точкой является определение линейного кроссворда (ЛК). Итак, ЛК – это «цепочка слов, которая строится методом стыкования, где последняя буква первого слова является первой буквой второго и т. д. В чайнворде, как и в кроссворде, используются только имена существительные в именительном падеже и единственном числе». Так как ЛК строится из слов, то необходимо указать ограничение на длину слова: минимальную длину слова можно определить равную 3, а максимальную – 15, потому что чаще всего кроссворды будут состояться на бытовые темы и сам ЛК не должен быть очень простым. Чтобы ЛК было интересно разгадывать, он не должен быть очень коротким, поэтому минимальное количество слов, например, 5, а максимальное – 15. Идем дальше: «слова в чайнворде не пересекаются, а только стыкуются друг с другом. Иногда цепочку слов изгибают для придания сетке причудливой формы. Длинная изогнутая цепочка может неоднократно пересекать саму себя, как слова в кроссворде, такая головоломка обычно называется кросс чайнвордом». Исходя из этого, можно задать следующее ограничение – на форму отображения ЛК: обычная (линейная), спираль, змейка, W-образная. «В линейных кроссвордах слова могут перекрываться не только одной, но и двумя или тремя буквами, поэтому их длина указывается в скобках при определении к слову», эта часть описания ЛК дает еще одно ограничение: количество букв в пересечении от 1 до 3. В качестве пожелания, заказчик отметил, что ЛК необходимо строить в двух режимах: ручном и автоматическом (генерация кроссворда), из этого следует еще одно ограничение – составление кроссворда осуществляется с привязкой к словарю понятий. Для того чтобы системы была более универсальной (необходимо обеспечить создание тематических кроссвордов или на общие области знаний), заказчик предложил

загружать в систему внешние словари понятий и обеспечить ему возможность редактировать их содержимое. При разгадывании кроссворда могут возникнуть затруднения, поэтому (в соответствии с пожеланиями заказчика) в системе должна быть организована система подсказок, количество которых можно связать с количеством слов – не менее 1 и не более 10% от количества слов.

Требования к информационному обеспечению. Разработка информационного обеспечения (ИО) – наиболее важная часть проекта, она может оказать существенное влияние на весь процесс разработки, поэтому уже на стадии разработки ТЗ необходимо определить:

1. На основании каких документов разрабатывается методическое и информационное обеспечение системы (нормативные и другие документы).

2. Перечень исходных данных:

- какие массивы данных используются и в каких форматах;

- на каких носителях эти данные будут поставляться в систему.

3. Перечень выходных данных:

- какие массивы данных будут являться результатом работы системы;

- какие документы будут представлены пользователю и в каком виде (указывается вид носителя) и с какой периодичностью;

- какие требования по целостности данных и их защите должны быть выполнены в проектируемой системе.

Особо должны быть выделены файл-серверные и клиент-серверные части информационного обеспечения, если таковые имеются.

Для разработки данной системы никаких нормативных документов не требуется (стандартов, инструкций и т.п.), поэтому необходимо только сослаться на информацию, где определена структура и свойства линейного кроссворда, а также требования к его построению. Также необходимо определить требования по входным и выходным данным. Большинство параметров линейного кроссворда будут задаваться пользователем в режиме диало-

га: он обязательно должен подключить словарь понятий, из которого будут формироваться задания для кроссворда. В данном случае «словарь понятий» – это текстовый файл определенной структуры (каждая строка файла – это понятие и его расшифровка), который должен загружаться в систему из внешней памяти (с любого логического диска), с которым пользователь должен иметь возможность работать дополнительно. Обязательное условие заказчика – возможность создания коллекции ЛК, поэтому в системе должна быть предусмотрена возможность сохранения наиболее интересных кроссвордов в файл. На данном этапе еще трудно определить структуру этого файла (она должна учитывать и возможность дальнейшего разгадывания кроссворда с помощью данной системы), поэтому можно написать, что «структура файла определяется в процессе проектирования». Обязательным условием составления любого типа кроссвордов является его целостность (для данного случая это отсутствие пустых клеток в середине ЛК), поэтому это обязательно должно быть записано в требованиях.

Требования к техническому обеспечению. Здесь формулируются ограничения по составу технических средств автоматизации с указанием конкретных типов оборудования и ЭВМ или их составляющих, используемых в проекте, если они заранее известны. Иначе в этом разделе указывается, что состав комплекса технических средств системы определяется в процессе проектирования системы.

Требования к программному обеспечению. Приводится перечень используемых системных и прикладных программных средств, включая операционную систему, систему программирования, систему управления базами данных (в случае необходимости) и другие инструментальные средства (например, среда проектирования) с точным наименованием версий, если они заранее известны. Иначе указывается, что состав программного обеспечения определяется в процессе проектирования системы. Дополнительно могут быть указаны требования по совместимости разрабатываемого программного обеспечения с существующими системами.

Общие требования к проектируемой системе. В данной части ТЗ отдельно выделяется подраздел «Функции, реализуемые

системой». В нем приводится подробный перечень функций, которые должна выполнять проектируемая система (или подсистема) в процессе ее эксплуатации. Отдельно должны быть выделены функции ввода данных, их обработки, передачи, хранения, а также формирования отчетов с выдачей на экран или печатающие устройства, функции управления, работа со справочниками и различные сервисные (обслуживающие систему) функции. Формулировка функций должна быть однозначной и конкретной, так как именно она является основой приемки проекта руководителем и проверки на полноту и качество реализованной системы или подсистемы.

Функции, которые должна выполнять данная система, определяются в первую очередь видами автоматизируемой деятельности. Среди неявных функций, про которые не должны забывать разработчики, это:

- визуализация процессов работы с кроссвордом;
- выдача сведений о системе (справочные данные о системе и о том, как с ней работать).

Нужно отметить, что многие перечисленные в ТЗ функции, не раскрываются подробно, их необходимо будут детализировать при разработке функциональной спецификации.

В других подразделах оговариваются специальные технические требования, предъявляемые к системе:

- по быстродействию (времени реакции на выполнение наиболее важных функций);
- по режиму работы (диалоговый/интерактивный, автоматический);
- по точности (в случае, если в системе производятся математические расчеты, требующие минимизации вычислительных погрешностей, или используются внешние информационные источники (датчики, измерители и т.п.));
- по достоверности;
- по условиям функционирования (диапазон температур, относительная влажность, давление, наличие в атмосфере пыли, вредных примесей и т.д.),
- другие количественные и качественные показатели, определяющие эффективность функционирования системы.

Кроме того, в данном разделе указываются санитарные правила и нормы и ГОСТы, требования которых необходимо учитывать при разработке такого класса систем, с учетом того, что системы разворачиваются на средствах вычислительной техники.

Часть 3 – Календарный план выполнения работ. Технология RAD требует жесткого следования плану-графику работ, поэтому в ТЗ оговариваются ключевые задания, по которым преподаватель должен проводить обязательный контроль. Каждый из перечисленных этапов должен завершаться полностью готовой документацией, согласованной с заказчиком (руководителем). Невыполнение в срок какого-либо из этапов может привести либо к сдвигу «контрольных точек» по оставшимся этапам, либо к не завершению проекта в срок.

В заключение хотелось бы отметить, что процесс составления ТЗ на внедрение системы:

- требует от разработчиков коллективных обсуждений и принятия ответственных решений;
- позволяет выявить наиболее «узкие» места проекта и оценить возможные риски;
- дает возможность команде разработчиков распределить между собой все виды выполняемых работ, сосредоточив в дальнейшем усилия на концептуальных аспектах проекта;
- определить наиболее приоритетные функции, которые будут составлять каркас системы.

## 2.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Данное практическое занятие предполагает выполнение следующих этапов:

- изучение теоретических положений;
- разработка технического задания по выбранной системе;
- ответы на контрольные вопросы.

## 2.4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Из каких частей состоит техническое задание?
2. Поясните содержание задания.
3. Подразделы исходных данных к проекту.

4. Что указывается в характеристиках объекта автоматизации?
5. Что необходимо определить в требованиях к информационному обеспечению?
6. Что необходимо определить в требованиях к техническому обеспечению?
7. Что необходимо определить в требованиях к программному обеспечению?
8. Что содержат общие требования к проектируемой системе?
9. Специальные технические требования, предъявляемые к системе.
10. Календарный план выполнения работ.

### 3 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА ГРАФИКА РАЗРАБОТКИ И ВНЕДРЕНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

#### 3.1 ЦЕЛЬ РАБОТЫ

Цель работы – изучение методологии управления проектами, получение навыков по применению данных методологий для разработки графика разработки и внедрения проекта.

#### 3.2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Проблемы управления программными проектами впервые проявились в 60-х — начале 70-х годов. Руководители программных проектов выполняют такую же работу, что и руководители технических проектов. Вместе с тем процесс разработки программного обеспечения (ПО) существенно отличается от процессов реализации технических проектов, что порождает определенные сложности в управлении программными проектами. Ниже приведён краткий список этих отличий.

1. Программный продукт нематериален. Менеджер технического проекта видит результаты выполнения своего проекта. Если реализация проекта отстает от графика, это также видно воочию. В противоположность этому программное обеспечение нематериально. Его нельзя увидеть или потрогать. Менеджер программного проекта не видит процесс «роста» разрабатываемого ПО. Он может полагаться только на документацию, которая фиксирует процесс разработки программного продукта.

2. Не существует стандартных процессов разработки ПО. На сегодняшний день не существует четкой зависимости между процессом создания ПО и типом создаваемого программного продукта. Другие технические дисциплины имеют длительную историю, процессы разработки технических изделий многократно опробованы и проверены. Процессы создания большинства технических систем хорошо изучены. Изучением же процессов создания ПО специалисты занимаются только несколько последних лет. Поэтому пока нельзя точно предсказать, на каком этапе процесса разработки ПО могут возникнуть проблемы, угрожающие всему программному проекту.

3. Большие программные проекты - это часто «одноразовые» проекты. Большие программные проекты, как правило, значительно отличаются от проектов, реализованных ранее. Поэтому, чтобы уменьшить неопределенность в планировании проекта, руководители проектов должны обладать очень большим практическим опытом. Но постоянные технологические изменения в компьютерной технике и коммуникационном оборудовании обесценивают предыдущий опыт. Знания и навыки, накопленные опытом, могут не востребоваться в новом проекте.

Перечисленное выше может привести к тому, что реализация проекта выйдет из временного графика или превысит бюджетные ассигнования. Программные системы зачастую оказываются новинками как в «идеологическом», так и в техническом плане. Технические проекты, которые являются инновационными (например, новая транспортная система), также часто нарушают временные графики работ. Поэтому, предвидя возможные проблемы в реализации программного проекта, следует всегда помнить, что многим из них свойственно выходить за рамки временных и бюджетных ограничений.

### 3.2.1 Планирование проекта

Эффективное управление программным проектом напрямую зависит от правильного планирования работ, необходимых для его выполнения. План помогает менеджеру предвидеть проблемы, которые могут возникнуть на каких-либо этапах создания ПО, и разработать превентивные меры для их предупреждения или решения. План, разработанный на начальном этапе проекта, рассматривается всеми его участниками как руководящий документ, выполнение которого должно привести к успешному завершению проекта. Этот первоначальный план должен максимально подробно описывать все этапы реализации проекта.

Кроме разработки плана проекта, на менеджера ложится обязанность разработки других видов планов. Эти виды планов кратко описаны в таблице 3.1.



Таблица 3.1 – Виды планов

План	Описание
План качества	Описывает стандарты и мероприятия по поддержке качества разрабатываемого ПО
План аттестации	Описывает способы, ресурсы и перечень работ, необходимых для аттестации программной системы
План управления конфигурацией	Описывает структуру и процессы управления конфигурацией
План сопровождения ПО	Предлагает план мероприятий, требующихся для сопровождения ПО в процессе его эксплуатации, а также расчет стоимости сопровождения и необходимые для этого ресурсы
План по управлению персоналом	Описывает мероприятия, направленные на повышение квалификации членов команды разработчиков

В примере 1 показан процесс планирования создания ПО в виде псевдокода. Здесь сделан акцент на том, что планирование – это итерационный процесс. Поскольку в процессе выполнения проекта постоянно поступает новая информация, план должен регулярно пересматриваться. Важными факторами, которые должны учитываться при разработке плана, являются финансовые и деловые обязательства организации. Если они изменяются, эти изменения также должны найти отражение в плане работ.

Пример 1. Процесс планирования проекта.

Определение проектных ограничений

Первоначальная оценка параметров проекта

Определение этапов выполнения проекта и контрольных отметок

While loop (Пока проект не завершится или не будет остановлен)

Составление графика работ

Начало выполнения работ

Ожидание окончания очередного этапа работ

Отслеживание хода выполнения работ

```

    Пересмотр оценок параметров проекта
    Изменение графика работ
    Пересмотр проектных ограничений
    If (возникла проблема) then
        Пересмотр технических или организационных параметров
        проекта
    End if
    End loop

```

Процесс планирования начинается с определения проектных ограничений (временные ограничения, возможности наличного персонала, бюджетные ограничения и т.д.). Эти ограничения должны определяться параллельно с оцениванием проектных параметров, таких как структура и размер проекта, а также распределением функций среди исполнителей. Затем определяются этапы разработки и то, какие результаты (документация, прототипы, подсистемы или версии программного продукта) должны быть получены по окончании этих этапов. Далее начинается циклическая часть планирования. Сначала разрабатывается график работ по выполнению проекта или дается разрешение на продолжение использования ранее созданного графика. После этого (обычно через 2-3 недели) проводится контроль выполнения работ и отмечаются расхождения между реальным и плановым ходом работ.

Далее, по мере поступления новой информации о ходе выполнения проекта, возможен пересмотр первоначальных оценок параметров проекта. Это, в свою очередь, может привести к изменению графика работ. Если в результате этих изменений нарушаются сроки завершения проекта, должны быть пересмотрены (и согласованы с заказчиком ПО) проектные ограничения.

Конечно, большинство менеджеров проектов не думают, что реализация их проектов пройдет гладко, без всяких проблем. Желательно описать возможные проблемы еще до того, как они проявят себя в ходе выполнения проекта. Поэтому лучше составлять «пессимистические» графики работ, чем «оптимистические». Но, конечно, невозможно построить план, учитывающий все, в том числе случайные, проблемы и задержки выполнения проекта, поэтому и возникает необходимость периодического пересмотра

проектных ограничений и этапов создания программного продукта.

### 3.2.2 План проекта

План проекта должен четко показать ресурсы, необходимые для реализации проекта, разделение работ на этапы и временной график выполнения этих этапов. В некоторых организациях план проекта составляется как единый документ, содержащий все виды планов, описанных выше. В других случаях план проекта описывает только технологический процесс создания ПО. В таком плане обязательно присутствуют ссылки на планы других видов, но они разрабатываются отдельно от плана проекта.

План, представленный ниже, принадлежит именно к последнему типу планов. Детализация планов проектов очень разнится в зависимости от типа разрабатываемого программного продукта и организации-разработчика. Но в любом случае большинство планов содержат следующие разделы.

1. Введение. Краткое описание целей проекта и проектных ограничений (бюджетных, временных и т.д.), которые важны для управления проектом.

2. Организация выполнения проекта. Описание способа подбора команды разработчиков и распределение обязанностей между членами команды.

3. Анализ рисков. Описание возможных проектных рисков, вероятности их проявления и стратегий, направленных на их уменьшение. Тема управления рисками рассмотрена ниже.

4. Аппаратные и программные ресурсы, необходимые для реализации проекта. Перечень аппаратных средств и программного обеспечения, необходимого для разработки программного продукта. Если аппаратные средства требуется закупать, приводится их стоимость совместно с графиком закупки и поставки.

5. Разбиение работ на этапы. Процесс реализации проекта разбивается на отдельные процессы, определяются этапы выполнения проекта, приводится описание результатов («выходов») каждого этапа и контрольные отметки. Эта тема представлена ниже.

6. График работ. В этом графике отображаются зависимости между отдельными процессами (этапами) разработки ПО, оценки времени их выполнения и распределение членов команды разработчиков по отдельным этапам.

7. Механизмы мониторинга и контроля за ходом выполнения проекта. Описываются предоставляемые менеджером отчеты о ходе выполнения работ, сроки их предоставления, а также механизмы мониторинга всего проекта.

План должен регулярно пересматриваться в процессе реализации проекта. Одни части плана, например, график работ, изменяются часто, другие более стабильны. Для внесения изменений в план требуется специальная организация документопотока, позволяющая отслеживать эти изменения.

### 3.2.3 Контрольные отметки этапов работ

Менеджеру для организации процесса создания ПО и управления им необходима информация. Поскольку само программное обеспечение неосвязаемо, эта управленческая информация может быть получена только в виде документов, отображающих выполнение очередного этапа разработки программного продукта. Без этой информации нельзя судить о степени готовности создаваемого продукта, невозможно оценить произведенные затраты или изменить график работ.

При планировании процесса определяются контрольные отметки – вехи, отмечающие окончание определенного этапа работ. Для каждой контрольной отметки создается отчет, который предоставляется руководству проекта. Эти отчеты не должны быть большими объемными документами; они должны подводить краткие итоги окончания отдельного логически завершенного этапа проекта. Этапом не может быть, например, «Написание 80% кода программ», поскольку невозможно проверить завершение такого «этапа»; кроме того, подобная информация практически бесполезна для управления, поскольку здесь не отображается связь этого «этапа» с другими этапами создания ПО.

Обычно по завершении основных больших этапов, таких как разработка спецификации, проектирование и т.п., заказчику ПО предоставляются результаты их выполнения, так называемые

контрольные проектные элементы. Это может быть документация, прототип программного продукта, законченные подсистемы ПО и т.д. Контрольные проектные элементы, предоставляемые заказчику ПО, могут совпадать с контрольными отметками (точнее, с результатами выполнения какого-либо этапа). Но обратное утверждение неверно. Контрольные отметки – это внутренние проектные результаты, которые используются для контроля за ходом выполнения проекта, и они, как правило, не предоставляются заказчику ПО.

Для определения контрольных отметок весь процесс создания ПО должен быть разбит на отдельные этапы с указанным «выходом» (результатом) каждого этапа. Например, на рисунке 3.1 показаны этапы разработки спецификации требований в случае, когда для ее проверки используется прототип системы, а также представлены выходные результаты (контрольные отметки) каждого этапа. Здесь контрольными проектными элементами являются требования и спецификация требований.



Рисунок 3.1 – Этапы процесса разработки спецификации

### 3.2.4 График работ

Составление графика – одна из самых ответственных работ, выполняемых менеджером проекта. Здесь менеджер оценивает длительность проекта, определяет ресурсы, необходимые для реализации отдельных этапов работ, и представляет их (этапы) в виде согласованной последовательности. Если данный проект подобен ранее реализованному, то график работ последнего проекта можно взять за основу для данного проекта. Но затем следует учесть, что на отдельных этапах нового проекта могут использоваться методы и подходы, отличные от использованных ранее.

Если проект является инновационным, первоначальные оценки длительности и требуемых ресурсов наверняка будут слишком оптимистичными, даже если менеджер попытается

предусмотреть все возможные неожиданности. С этой точки зрения проекты создания ПО не отличаются от больших инновационных технических проектов. Новые аэропорты, мосты и даже новые автомобили, как правило, появляются позже первоначально объявленных сроков их сдачи или поступления на рынок, чему причиной являются неожиданно возникшие проблемы и трудности. Именно поэтому графики работ необходимо постоянно обновлять по мере поступления новой информации о ходе выполнения проекта.

В процессе составления графика (рисунок 3.2) весь массив работ, необходимых для реализации проекта, разбивается на отдельные этапы и оценивается время, требующееся для выполнения каждого этапа. Обычно многие этапы выполняются параллельно. График работ должен предусматривать это и распределять производственные ресурсы между ними наиболее оптимальным образом. Нехватка ресурсов для выполнения какого-либо критического этапа – частая причина задержки выполнения всего проекта.

Длительность этапов обычно должна быть не меньше недели. Если она будет меньше, то окажется ниже точности временных оценок этапов, что может привести к частому пересмотру графика работ. Также целесообразно (в аспекте управления проектом) установить максимальную длительность этапов, не превышающую 8 или 10 недель. Если есть этапы, имеющие большую длительность, их следует разбить на этапы меньшей длительности.

При расчете длительности этапов менеджер должен учитывать, что выполнение любого этапа не обойдется без больших или маленьких проблем и задержек. Разработчики могут допускать ошибки или задерживать свою работу, техника может выйти из строя либо аппаратные или программные средства поддержки процесса разработки могут поступить с опозданием. Если проект инновационный и технически сложный, это становится дополнительным фактором появления непредвиденных проблем и увеличения длительности реализации проекта по сравнению с первоначальными оценками.



Рисунок 3.2 – Процесс составления графика работ

Кроме временных затрат, менеджер должен рассчитать другие ресурсы, необходимые для успешного выполнения каждого этапа. Особый вид ресурсов – это команда разработчиков, привлеченная к выполнению проекта. Другими видами ресурсов могут быть необходимое свободное дисковое пространство на сервере, время использования какого-либо специального оборудования и бюджетные средства на командировочные расходы персонала, работающего над проектом.

Существует хорошее эмпирическое правило: оценивать временные затраты так, как будто ничего непредвиденного и «плохого» не может случиться, затем увеличить эти оценки для учета возможных проблем. Возможные, но трудно прогнозируемые проблемы существенно зависят от типа и параметров проекта, а также от квалификации и опыта членов команды разработчиков. К исходным расчетным оценкам рекомендуется добавлять 30% на возможные проблемы и затем еще 20%, чтобы быть готовым к тому, что невозможно предвидеть.

График работ по проекту обычно представляется в виде набора диаграмм и графиков, показывающих разбиение проектных работ на этапы, зависимости между этапами и распределение разработчиков по этапам.

Временные и сетевые диаграммы полезны для представления графика работ. Временная диаграмма показывает время начала и окончания каждого этапа и его длительность. Сетевая диаграмма отображает зависимости между различными этапами проекта. Эти диаграммы можно создать автоматически с помощью программных средств поддержки управления на основе информации, заложенной в базе данных проекта.

Рассмотрим этапы некоего проекта, представленные в таблице 3.2, из которой, в частности, видно, что этап Т3 зависит от этапа Т1. Это значит, что этап Т1 должен завершиться прежде, чем начнется этап Т3. Например, на этапе Т1 проводится компо-

нентный анализ создаваемого программного продукта, а на этапе Т3 – проектирование системы.

На основе приведенных значений длительности этапов и зависимости между ними строится сетевой график последовательности этапов (рисунок 3.3). На этом графике видно, какие работы могут выполняться параллельно, а какие должны выполняться последовательно друг за другом. Этапы обозначены прямоугольниками. Контрольные отметки и контрольные проектные элементы показаны в виде овалов и обозначены (как и в таблице 3.2) буквой М с соответствующим номером. Даты на данной диаграмме соответствуют началу выполнения этапов. Сетевую диаграмму следует читать слева направо и сверху вниз.

Таблица 3.2 – Этапы проекта

Этап	Длительность (дни)	Зависимость
T1	8	
T2	15	
T3	15	T1 (M1)
T4	10	
T5	10	T2, T4 (M2)
T6	5	T1, T2 (M3)
T7	20	T1 (M1)
T8	25	T4 (M5)
T9	15	T3, T6 (M4)
T10	15	T5, T7 (M7)
T11	7	T9 (M6)
T12	10	T11 (M8)



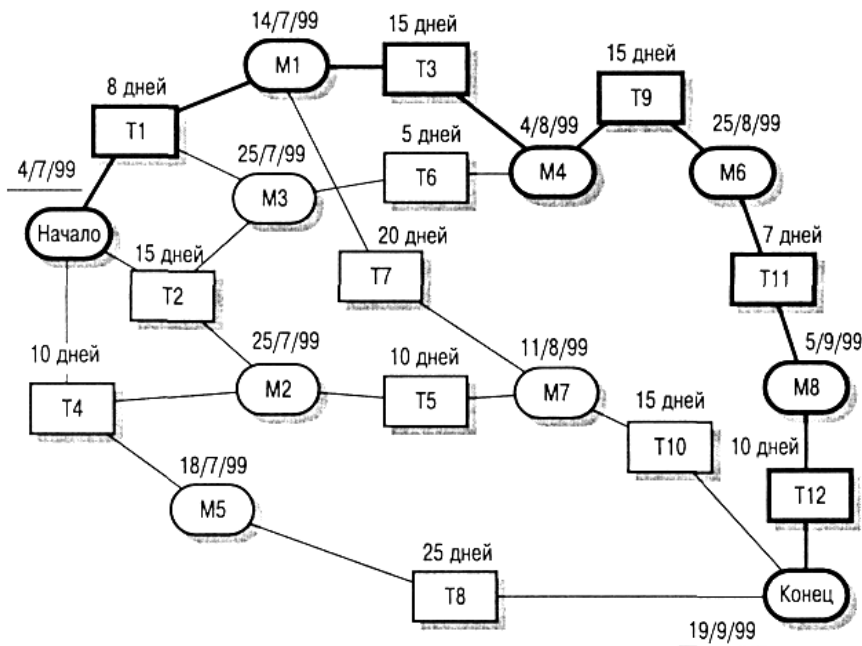


Рисунок 3.3 – Сетевая диаграмма этапов

Если для создания сетевой диаграммы используются программные средства поддержки управления проектом, каждый этап должен заканчиваться контрольной отметкой. Очередной этап может начаться только тогда, когда будет получена контрольная отметка (которая может зависеть от нескольких предшествующих этапов). Поэтому в третьем столбце таблицы 3.2 приведены контрольные отметки; они будут достигнуты только тогда, когда будет завершен этап, в строке которого помещена соответствующая контрольная отметка.

Любой этап не может начаться, пока не выполнены все этапы на всех путях, ведущих от начала проекта к данному этапу. Например, этап T9 не может начаться, пока не будут завершены этапы T3 и T6. Отметим, что в данном случае достижение контрольной отметки M4 говорит о том, что эти этапы завершены.

Минимальное время выполнения всего проекта можно рассчитать, просуммировав в сетевой диаграмме длительности этапов на самом длинном пути (длина пути здесь измеряется не количеством этапов на пути, а суммарной длительностью этих этапов) от начала проекта до его окончания (это так называемый критический путь). В нашем случае продолжительность проекта составляет 11 недель или 55 рабочих дней. На рисунке 3.3 критический путь показан более толстыми линиями, чем остальные пути. Таким образом, общая продолжительность реализации проек-

та зависит от этапов работ, находящихся на критическом пути. Любая задержка в завершении любого этапа на критическом пути приведет к задержке всего проекта.

Задержка в завершении этапов, не входящих в критический путь, не влияет на продолжительность всего проекта до тех пор, пока суммарная длительность этих этапов (с учётом задержек) на каком-нибудь пути не превысит продолжительности работ на критическом пути. Например, задержка этапа Т8 на срок, меньший 20 дней, никак не влияет на общую продолжительность проекта. На рисунке 3.4 представлена временная диаграмма, на которой показаны возможные задержки на каждом этапе.

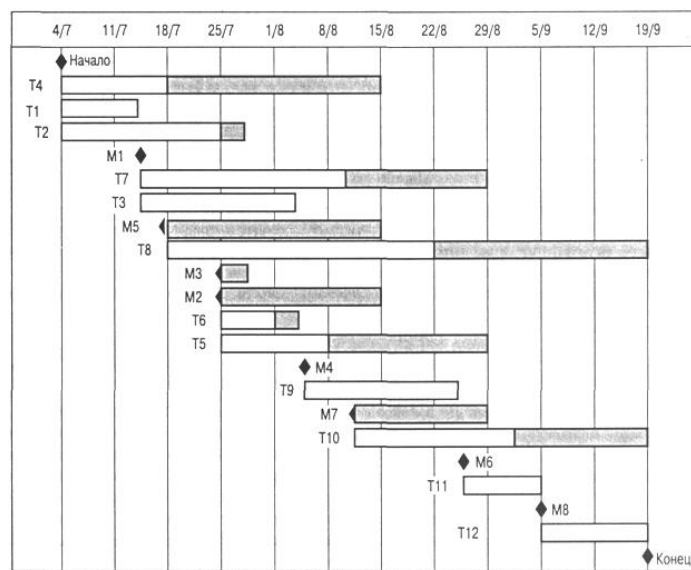


Рисунок 3.4 – Временная диаграмма длительности этапов

Сетевая диаграмма позволяет увидеть в зависимости этапов значимость того или иного этапа для реализации всего проекта. Внимание к этапам критического пути часто позволяет найти способы их изменения с тем, чтобы сократить длительность всего проекта. Менеджеры используют сетевую диаграмму для распределения работ.

На рисунке 3.4 показано другое представление графика работ. Это временная диаграмма (иногда называемая по имени ее изобретателя диаграммой Ганта) может быть построена программными средствами поддержки процесса управления. Она показывает длительность выполнения каждого этапа и возможные их задержки (показаны затененными прямоугольниками), а также даты начала и окончания каждого этапа. Этапы критического пу-

ти не имеют затененных прямоугольников; это означает, что задержка с завершением данных этапов приведет к увеличению длительности всего проекта.

Подобно распределению времени выполнения этапов, менеджер должен рассчитать распределение ресурсов по этапам, в частности назначить исполнителей на каждый этап. В таблице 3.3 приведено распределение разработчиков на каждый этап, представленный на рисунке 3.4.

Таблица 3.3 – Распределение исполнителей по этапам

Этап	Исполнитель
T1	Джейн
T2	Анна
T3	Джейн
T4	Фред
T5	Мэри
T6	Анна
T7	Джим
T8	Фред
T9	Джейн
T10	Анна
T11	Фред
T12	Фред

Приведенная таблица может быть использована программными средствами поддержки процесса управления для построения временной диаграммы занятости сотрудников на определенных этапах работ (рисунок 3.5). Персонал не занят в работе над проектом все время его реализации. В течение периода незанятости сотрудники могут быть в отпуске, работать над другими проектами, проходить обучение и т.д.

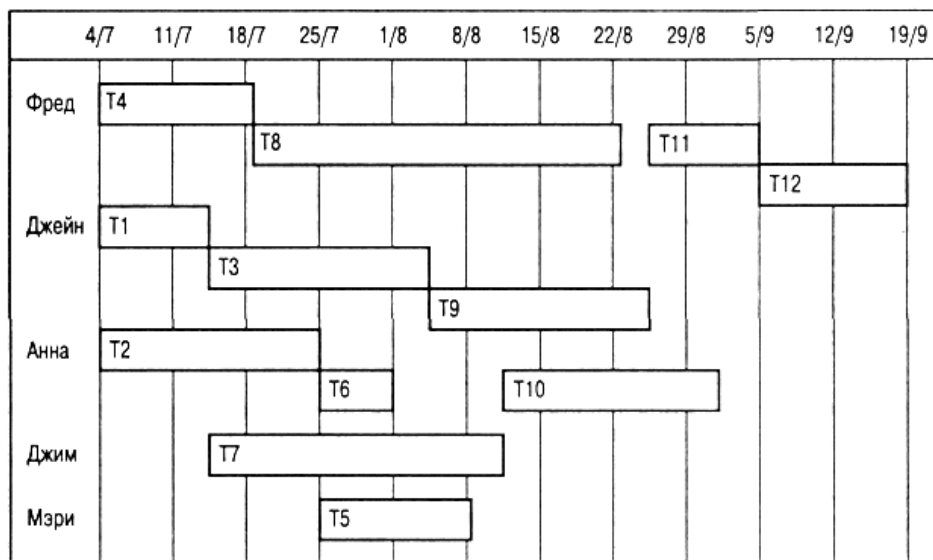


Рисунок 3.5 – Временная диаграмма распределения работников по этапам

В больших организациях обычно работает много специалистов, которые задействуются в проекте по мере необходимости. Конечно, такой подход может создать определенные проблемы для менеджеров проектов. Например, если специалист занят в проекте, который задерживается, это может создать прямые сложности для других проектов, где он также должен участвовать.

Первоначальный график работ неизбежно содержит какие-нибудь ошибки или недоработки. По мере реализации проекта рассчитанные оценки длительности выполнения этапов работ должны сравниваться с реальными сроками выполнения этих этапов. Результаты сравнения должны использоваться в качестве основы для пересмотра графика работ еще не реализованных этапов проекта, в частности для того, чтобы попытаться уменьшить длительность этапов критического пути.

### 3.2.5 Управление рисками

Важной частью работы менеджера проекта является оценка рисков, которые могут повлиять на график работ или на качество создаваемого программного продукта, и разработка мероприятий по предотвращению рисков. Результаты анализа рисков должны быть отражены в плане проекта. Определение рисков и разработка мероприятий по уменьшению их влияния на ход выполнения проекта называется управлением рисками.

Упрощенно риск можно понимать как вероятность проявления каких-либо неблагоприятных обстоятельств, негативно влияющих на реализацию проекта. Риски могут угрожать проекту в целом, создаваемому программному продукту или организации-разработчику. Можно выделить три типа рисков.

1. Риски для проекта, которые влияют на график работ или ресурсы, необходимые для выполнения проекта.

2. Риски для разрабатываемого продукта, влияющие на качество или производительность разрабатываемого программного продукта.

3. Бизнес-риски, относящиеся к организации-разработчику или поставщикам.

Конечно, эти типы рисков могут пересекаться. Например, если опытный программист покидает проект, это будет риском для проекта (поскольку задерживается срок сдачи готового продукта), риском для продукта (так как новый программист, заменивший ушедшего, может оказаться не слишком опытным и сделать ошибки в программе) и бизнес-риском (поскольку задержка данного проекта может негативно повлиять на будущие деловые контакты между заказчиком и организацией-разработчиком).

Конкретные типы рисков, которые могут оказать влияние на данный проект, зависят от вида создаваемого программного продукта и от организационного окружения, где реализуется программный проект. Вместе с тем многие типы рисков способны повлиять на любые программные проекты, эти риски приведены в таблице 3.4.

Таблица 3.4 – Возможные риски программных проектов

Риск	Типы риска	Описание риска
Текущность разработчиков	Риск для проекта	Опытные разработчики покидают проект до его завершения
Изменение в управлении организацией	Риск для проекта	Организация меняет свои приоритеты в управлении проектом
Неготовность аппаратных средств	Риск для проекта	Аппаратные средства, которые необходимы для проекта, не поступили вовремя или не готовы к эксплуатации
Изменение требований	Риск для проекта и для разрабатываемого продукта	Появление большого количества непредвиденных изменений в требованиях, предъявляемых к разрабатываемому ПО

## Продолжение таблицы 3.4

Риск	Типы риска	Описание риска
Задержка в разработке спецификации	Риск для проекта и для разрабатываемого продукта	Спецификации основных интерфейсов подсистем не поступили к разработчикам в соответствии с графиком работ
Недооценка размера разрабатываемой системы	Риск для проекта и для разрабатываемого продукта	Размер системы значительно превысил первоначальную оценку
Недостаточная эффективность CASE-средств	Риск для разрабатываемого продукта	CASE-средства, предназначенные для поддержки проекта, оказались менее эффективными, чем ожидалось
Изменения в технологии разработки ПО	Бизнес-риск	Основные технологии построения программной системы заменяются новыми
Появление конкурирующего программного продукта	Бизнес-риск	На рынке программных продуктов до окончания проекта появилась конкурирующая программная система

Схема процесса управления рисками показана на рисунке 3.6. Этот процесс состоит из четырех стадий.

1. Определение рисков. Определяются возможные риски для проекта, для разрабатываемого продукта и бизнес-риски.

2. Анализ рисков. Оценивается вероятность и последовательность появления рисковых ситуаций.

3. Планирование рисков. Планируются мероприятия по предотвращению рисков или минимизации их воздействия на проект.

4. Мониторинг рисков. Постоянное оценивание вероятностей рисков и выполнение мероприятий по смягчению последствий проявления рисков ситуаций.

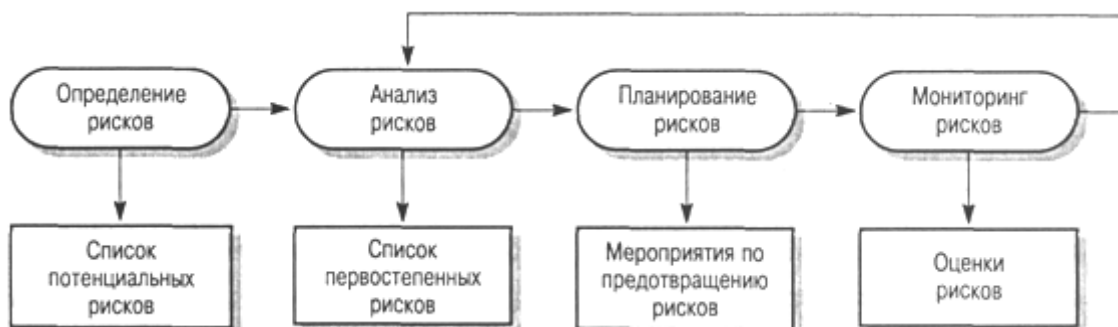


Рисунок 3.6 – Процесс управления рисками

Процесс управления рисками, как и другие процессы планирования, является итерационным, выполняемым в течение всего срока реализации проекта. Сначала разрабатываются планы управления рисками, затем постоянно отслеживается ситуация вокруг реализации проекта. При поступлении новой информации о возможных рисках заново проводится анализ рисков и первостепенное внимание уделяется новым рискам. По мере поступления новой информации также изменяются планы мероприятий по предотвращению и смягчению рисков.

Результаты процесса управления рисками документируются в виде планов управления рисками. Они должны включать описание возможных проектных рисков, их анализ и перечень мероприятий, необходимых для управления рисками.

Определение рисков – первая стадия процесса управления рисками. На этой стадии описываются риски, которые могут проявиться при реализации проекта. В принципе на этой стадии не должна оцениваться вероятность и значимость рисков, но на практике маловероятные риски с незначительными последствиями обычно отбрасываются сразу.

Определение рисков может выполняться в режиме командной работы с использованием подхода «мозговой штурм» либо основываться на опыте менеджера. При определении рисков может помочь приведенный ниже список возможных категорий рисков.



1. Технологические риски. Проистекают из программных и аппаратных технологий, на основе которых разрабатывается система.

2. Риски, связанные с персоналом. Связаны с членами команды разработчиков.

3. Организационные риски. Проистекают из организационного окружения, в котором выполняется проект.

4. Инструментальные риски. Связаны с используемыми CASE-средствами и другими средствами поддержки процесса создания ПО.

5. Риски, связанные с системными требованиями. Проявляются при изменении требований, предъявляемых к разрабатываемой системе.

6. Риски оценивания. Связаны с оцениванием характеристик программной системы и ресурсов, необходимых для реализации проекта.

В таблице 3.5 представлены некоторые примеры, относящиеся к каждой из описанных категорий рисков. Результатом этапа определения рисков будет длинный список возможных рисков, которые могут повлиять на разрабатываемый программный продукт, проект или организацию-разработчика.

Таблица 3.5 – Категории рисков

Категория рисков	Примеры рисков
Технологические риски	База данных, которая используется в программной системе, не обеспечивает обработку ожидаемого объема транзакций. Программные компоненты, которые используются повторно, имеют дефекты, ограничивающие их функциональные возможности
Риски, связанные с персоналом	Невозможно подобрать работников с требуемым профессиональным уровнем. Ведущий разработчик заболел в самое критическое время. Невозможно организовать необходимое обучение персонала

Продолжение таблицы 3.5

Категория рисков	Примеры рисков
Организационные риски	В организации, выполняющей разработку ПО, произошла реорганизация, в результате чего изменились приоритеты в управлении проектом. Финансовые затруднения в организации привели к уменьшению бюджета проекта
Инструментальные риски	Программный код, генерируемый CASE-средствами, не эффективен. CASE-средства невозможно интегрировать с другими средствами поддержки проекта
Риски, связанные с системными требованиями	Изменения требований приводят к значительным повторным темным требованиям к работам по проектированию системы. Первоначальная нечеткая формулировка пользовательских требований привела к значительным изменениям системных требований, проявившихся на поздних стадиях разработки проекта
Риски оценивания	Недооценки времени выполнения проекта. Скорость выявления дефектов в системе ниже ранее запланированной. Размер системы значительно превышает первоначально рассчитанный

При анализе для каждого определенного риска подсчитывается вероятность его проявления и ущерб, который он может нанести. Не существует простых методов выполнения анализа рисков – в значительной мере он основан на мнении и опыте менеджера. Можно привести следующую шкалу вероятностей рисков и их последствий.

1. Вероятность риска считается очень низкой, если она имеет значение менее 10%; низкой, если ее значение от 10 до 25%; средней при значениях от 25 до 50%; высокой, если значение колеблется от 50 до 75%; очень высокой при значениях более 75%.

2. Возможный ущерб от рисков ситуаций можно подразделить на катастрофический, серьезный, терпимый и незначительный.

Результаты анализа рисков должны быть представлены в виде таблицы рисков, упорядоченных по степени возможного ущерба. В таблице 3.6 приведен упорядоченный список рисков, описанных в таблице 3.5, там же указаны вероятности этих рисков. Здесь вероятности рисков и степень ущерба от них указаны произвольно. На практике для их определения необходима подробная информация о проекте, технологии создания ПО, команде разработчиков и о самой организации.

Таблица 3.6 – Список рисков после проведения их анализа

Риск	Вероятность	Степень ущерба
Финансовые затруднения в организации привели к уменьшению бюджета проекта	Низкая	Катастрофическая
Невозможно подобрать работников с требуемым профессиональным уровнем	Высокая	Катастрофическая
Ведущий разработчик заболел в самое критическое время	Средняя	Серьезная
Программные компоненты, используемые повторно, имеют дефекты, ограничивающие их функциональные возможности	Средняя	Серьезная
Изменения требований приводят к значительным повторным работам по проектированию системы	Средняя	Серьезная
В организации, выполняющей разработку ПО, произошла реорганизация, в результате чего изменились приоритеты в управлении проектом	Высокая	Серьезная

Продолжение таблицы 3.6

Риск	Вероятность	Степень ущерба
База данных, которая используется в программной системе, не обеспечивает обработку ожидаемого объема транзакций	Средняя	Серьезная
Недооценки времени выполнения проекта	Высокая	Серьезная
CASE-средства невозможно интегрировать с другими средствами поддержки проекта	Высокая	Терпимая
Первоначальная нечеткая формулировка пользовательских требований привела к значительным изменениям системных требований, проявившихся на поздних стадиях разработки проекта	Средняя	Терпимая
Невозможно организовать необходимое обучение персонала	Средняя	Терпимая
Скорость выявления дефектов в системе ниже ранее спланированной	Средняя	Терпимая
Размер системы значительно превышает первоначально рассчитанный	Высокая	Терпимая
Программный код, генерируемый CASE-средствами, неэффективен	Средняя	Незначительная

Конечно, как вероятность рисков, так и возможный ущерб от них должны пересматриваться при поступлении дополнительной информации об этих рисках и по мере реализации мероприятий по управлению ими. Поэтому подобные таблицы рисков должны переделываться на каждой итерации процесса управления рисками.

После проведения анализа рисков определяются наиболее значимые риски, которые затем отслеживаются на протяжении всего срока выполнения проекта. Определение этих значимых рисков зависит от их вероятностей и возможного ущерба. В общем случае всегда отслеживаются риски с катастрофическими последствиями, а также риски с серьезным ущербом, значение вероятности которых выше среднего.

В некоторых статьях рекомендуется определить и отслеживать «10 верхних» рисков, но это не всегда обоснованная рекомендация. Количество рисков, которые необходимо отслеживать, зависит от конкретного проекта. Это может быть пять рисков, а может — пятнадцать. Но, конечно, количество рисков, по которым проводится мониторинг, должно быть обозримым. Большое количество отслеживаемых рисков потребует огромного количества собираемой информации. Из списка рисков, представленных в табл. 6, для мониторинга следует отобрать те риски, которые могут привести к катастрофическим и серьезным последствиям для вашего проекта.

Планирование рисков заключается в определении стратегии управления каждым значимым риском, отобранным для мониторинга после анализа рисков. Здесь также не существует общепринятых подходов для разработки таких стратегий — многое основывается на «чутье» и опыте менеджера проекта. В таблице 3.7 показаны возможные стратегии управления основными рисками, приведенными в таблице 3.6.

Таблица 3.7 – Стратегии управления рисками

Риск	Стратегия
Финансовые проблемы организации	Подготовить краткий документ для руководства организации, показывающий важность данного проекта для достижения финансовых целей организации
Проблемы неквалифицированного персонала	Предупредить заказчика о потенциальных трудностях и возможной задержке проекта, рассмотреть вопрос о покупке компонентов системы

## Продолжение таблицы 3.7

Риск	Стратегия
Болезни персонала	Реорганизовать работу команды разработчиков таким образом, чтобы обязанности и работа членов команды перекрывали друг друга, вследствие этого разработчики будут знать и понимать задачи, выполняемые другими сотрудниками
Дефектные системные компоненты	Заменить потенциально дефектные системные компоненты покупными компонентами, гарантирующими качество работы
Изменения требований	Попытаться определить требования, наиболее вероятно подверженные изменениям; в структуре системы не отображать детальную информацию
Реорганизация компании-разработчика	Подготовить краткий документ для руководства компании, показывающий важность данного проекта для достижения финансовых целей компании
Недостаточная производительность базы данных	Рассмотреть возможность покупки более производительной базы данных
Недооценки времени выполнения проекта	Рассмотреть вопрос о покупке системных компонентов, исследовать возможность использования генератора программного кода

Существует три категории стратегий управления рисками.

1. Стратегии предотвращения рисков. Согласно этим стратегиям следует проводить мероприятия, снижающие вероятность проявления рисков. Примером может служить стратегия исключения потенциально дефектных компонентов, описанная в таблице 3.7.

2. Минимизационные стратегии. Направлены на уменьшение возможного ущерба от рисков. Примером служит страте-

гия уменьшения ущерба от болезни членов команды разработчиков (таблица 3.7).

3. Планирование «аварийных» ситуаций. Согласно этим стратегиям необходимо иметь план мероприятий, которые следует выполнить в случае проявления рисков ситуации. В таблице 3.7 это стратегия поведения при возникновении финансовых проблем у организации-разработчика.

Мониторинг рисков заключается в регулярном пересчете вероятностей рисков и ущерба, который они могут нанести. Для этого необходимо постоянно отслеживать факторы, которые влияют на вероятность рисков и возможный ущерб. Эти факторы зависят от типов риска. В таблице 3.8 приведены признаки, которые помогают определить тип риска.

Таблица 3.8 – Признаки рисков

Тип риска	Признаки
Технологические риски	Задержки в поставке оборудования или программных средств поддержки процесса создания ПО, многочисленные документированные технологические проблемы
Риски, связанные с персоналом	Низкое моральное состояние персонала, натянутые отношения между членами команды разработчиков, низкое качество выполненной работы
Организационные риски	Разговоры среди персонала о пассивности и недостаточной компетентности высшего руководства организации
Инструментальные риски	Нежелание разработчиков использовать программные средства поддержки, неодобрительные отзывы о CASE-средствах, запросы на более мощные инструментальные средства
Риски, связанные с системными требованиями	Необходимость пересмотра многих системных требований, недовольство заказчика ПО
Риски оценивания	Изменения графика работ, многочисленные отчеты о нарушении графика работ

Мониторинг рисков должен быть непрерывным процессом, отслеживающим ход выполнения мероприятий по управлению рисками, при этом каждый основной риск должен рассматриваться отдельно.

### 3.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Данное практическое занятие предполагает выполнение следующих этапов:

- изучить теоретический материал;
- построить временную и сетевую диаграммы для выбранного проекта;
- построить диаграмму распределения участников группы по этапам;
- построить список возможных рисков с указанием названия риска, его описание и типа;
- провести анализ рисков;
- описать стратегию планирования рисков;
- построить отчёт, включающий все полученные диаграммы и описание стратегии планирования рисков;
- ответить на контрольные вопросы.

### 3.4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Сложности в управлении программными проектами.
2. Перечислите виды планов.
3. Опишите план сопровождения.
4. Перечислите основные разделы плана.
5. Что понимается под контрольными отметками этапов работ?
6. Перечислите этапы разработки спецификации требований.
7. Основные этапы процесса составления графика работ.
8. Для чего используются временные диаграммы?
9. Для чего используются сетевые диаграммы?
10. Типы рисков проекта.
11. Перечислите возможные риски программных проектов.
12. Перечислите категории риски программных проектов.
13. Признаки рисков программных проектов.



## 4 ЛАБОРАТОРНАЯ РАБОТА. СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОЛОГИЙ ПРОЕКТИРОВАНИЯ

### 4.1 ЦЕЛЬ РАБОТЫ

Цель работы – изучить методологии проектирования.

### 4.2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Сравнительный анализ методологий проведем по следующим параметрам:

- адекватность средств рассматриваемой проблеме;
- согласованность с другими средствами структурного анализа;
- интеграция с последующими фазами проектирования системы.

#### 4.2.1 Адекватность

Выбор той или иной методологии и нотации структурного моделирования напрямую зависит от специфики предметной области, для которой создается модель. Общая теория систем рассматривает модель любой предметной области в виде двух основных множеств: объектов (сущностей, процессов) и отношений (связей, зависимостей) между объектами. Элементарный подход к анализу специфики предметной области заключается в выяснении соотношения объектов и связей (чего больше?). Более сложные методологии анализа предметной области предполагают выделение подмножества базовых объектов и их свойств и классификацию отношений. Теория систем, которая изучает прежде всего отношения предметной области, предполагает наличие у инструмента изучения развитых средств классификации отношений, спецификации их семантики. Отсутствие у инструмента моделирования предметной области средств описания семантики отношений, ориентация его в большей степени на спецификацию семантики объектов (процессов) может привести к построению неадекватной модели.

В качестве основных предметных областей будем рассматривать бизнес-процессы и системы обработки информации.

BPR – реорганизация бизнес-процессов. Под реорганизацией понимают основательное переосмысление и перепланирование критических бизнес-процессов, имеющие целью резко улучшить их выполнение в отношении затрат, качества обслуживания и скорости. При этом бизнес-процесс представляет собой некоторую деятельность, получающую входные данные одного или нескольких типов и выдающую результат, имеющий ценность для клиента.

Для моделирования бизнес-процессов традиционно используется методология SADT (точнее, ее подмножество IDEF0). Однако статическая SADT-модель полностью не решает задач реорганизации, для этого необходимо иметь возможность исследования динамических характеристик бизнес-процессов. Одним из решений является использование динамических моделей, основанных на цветных (раскрашенных) сетях Петри (CPN – Color Petri Nets). Фактически SADT и CPN служат компонентами интегрированной методологии реорганизации: SADT-диаграммы автоматически преобразуются в прообраз CPN-модели, который дорабатывается вручную и затем исполняется в различных режимах, чтобы получить соответствующие оценки.

Следует отметить, что не существует принципиальных ограничений для использования традиционных диаграмм потоков данных в качестве средства построения статических моделей бизнес-процессов. Более того, в настоящий момент доступен ряд методологий и продуктов динамического моделирования (INCOME Mobile, CPN-AMI и др.), базирующихся на сетях Петри различного вида и интегрируемых с DFD-моделью, которые позволяют успешно решать подобные задачи.

Системы обработки информации. Любой класс систем успешно моделируется при помощи DFD-ориентированных методов: в этом случае вместо реальных объектов рассматриваются отношения, описывающие свойства объектов и правила их поведения. Примерами таких систем служат организационные системы, системы документооборота, управления и другие системы, богатые разнообразными отношениями. В случае, если нотация DFD не адекватна специфике предметной области (например, предметная область с жесткими технологическими процессами: обработка деталей на станках, подготовка ракеты к запуску и т.

п.), то для ее анализа применяются более адекватные средства: технологические карты, диаграммы потоков управления и т. д.

По мнению специалистов в области проектирования информационных систем SADT-диаграммы значительно менее выразительны и удобны для моделирования систем обработки информации. Так, дуги в SADT-диаграммах жестко типизированы (вход, выход, управление, ресурс). В силу общности определения в SADT отсутствуют средства детального описания содержания и структуры дуг (их семантики), за исключением именования. Аналогичная проблема возникает и при описании блоков (процессов).

Однако DFD-диаграммы имеют удобные средства для описания семантики потоков (представляемых дугами диаграммы) и правил преобразования входных данных в выходные (миниспецификации).

Применительно к системам обработки информации стирается смысловое различие между используемыми в SADT входами-выходами, с одной стороны, и управлениями и механизмами, с другой: в системах обработки информации входы, выходы и управления являются потоками данных и правилами их трансформации. Анализ системы при помощи потоков данных и процессов, их преобразующих, является более прозрачным и недвусмысленным.

В SADT отсутствуют выразительные средства для моделирования особенностей систем обработки информации. DFD с самого начала создавались как средство проектирования информационных систем (тогда как SADT – как средство проектирования систем вообще) и имеют более богатый набор элементов, адекватно отражающих специфику таких систем. Например, хранилища данных являются прообразами носителей информации – файлов или баз данных; внешние сущности отражают взаимодействие моделируемой системы с внешним миром.

Наличие миниспецификаций DFD-процессов нижнего уровня позволяет преодолеть логическую незавершенность SADT (а именно, обрыв модели на некотором достаточно низком уровне, когда дальнейшая ее детализация становится бессмысленной) и построить полную функциональную спецификацию разрабатываемой системы. Это позволит расширить возможности примене-

ния созданной модели (например, ее можно будет использовать для автоматизированного и быстрого обучения новых работников конкретному направлению деятельности).

Ограничения SADT, запрещающие использовать более пяти-семи блоков на диаграмме, вынуждают искусственно детализировать систему, что затрудняет понимание модели заказчиком, резко увеличивает объем и, как следствие, ведет к неадекватности модели с реальной картиной.

#### 4.2.2 Согласованность с другими средствами структурного анализа

Речь идет о согласованности со средствами информационного и временного моделирования системы. Поскольку SADT-диаграммы предназначены для моделирования систем общего класса и в них отсутствуют средства описания данных и событий, то согласование модели, например, с ERD- или STD-диаграммами практически невозможно или носит тривиальный характер.

В свою очередь, DFD-, ERD- и STD-диаграммы взаимно дополняют друг друга и по сути являются согласованными представлениями различных аспектов одной и той же модели.

Отметим, что интеграция DFD – STD осуществляется за счет расширения классической DFD специальными средствами проектирования систем реального времени (управляющими процессами, потоками, хранилищами данных) и STD является детализацией управляющего процесса, согласованной по управляющим потокам и хранилищам.

#### 4.2.3 Интеграция с последующими фазами проектирования системы

Важная характеристика методологии – ее совместимость с последующими этапами применения результатов анализа. Если речь идет о системах обработки информации, то результаты анализа используются затем на фазах проектирования, реализации и тестирования системы. Здесь рассматриваются все методы и средства, применяемые на последующих фазах. Однако отметим потенциальную совместимость методологий анализа с наиболее

широко распространенными методами проектирования, реализации и т. д.

Неизвестны формальные методы преобразования SADT-диаграмм в проектные решения системы обработки информации. В то же время DFD-диаграммы могут быть легко преобразованы при проектировании системы. Известен ряд алгоритмов автоматического преобразования иерархии DFD в структурные схемы различных видов, что обеспечивает логичный и безболезненный переход от этапа анализа требований к проектированию системы.

### 4.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Данное практическое занятие предполагает выполнение следующих этапов:

- изучить методические указания;
- ответить на контрольные вопросы.

### 4.4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что лежит в основе Case-средств?
2. Назовите и охарактеризуйте компоненты Case- средств.
3. Какие положения лежат в основе Case- средств?
4. Какими свойствами должна обладать формализованная модель, представленная Case- средством?
5. Дайте характеристику диаграммам, представленным с помощью средств моделирования.
6. На какие группы с точки зрения функционального моделирования принято делить все разновидности структурного анализа?
7. Охарактеризовать методологию SADT.
8. На чём основывается модель с точки зрения SADT?
9. Как представляются дуальные модели системы?
10. Дать характеристику основного элемента моделирования.
11. Как на диаграммах отражаются связи и отношения элементов модели?
12. В чём преимущество методологии SADT?
13. Охарактеризовать методологию Гейни-Сарсона.

14. Сколько этапов логического моделирования выделяют по методологии Гейни-Сарсона?
15. Дать характеристику 1 и 2-му этапу логического моделирования по методологии Гейни-Сарсона.
16. Охарактеризовать методологию Йодана.
17. Перечислите составляющие и этапы структурного моделирования.
18. На чём основывается методология моделирования данных?
19. Назовите критерии сравнения методологий проектирования.
20. Что предполагает реорганизация бизнес-процессов?

## 5 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. АНАЛИЗ БИЗНЕС-ПРОЦЕССОВ ПОДРАЗДЕЛЕНИЯ

### 5.1 ЦЕЛЬ РАБОТЫ

Цель работы – изучение процесса функционального моделирования для заданной предметной области с помощью инструментальной среды BPwin.

### 5.2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Наиболее удобным языком моделирования бизнес-процессов является IDEF0, предложенный более 40 лет назад Дугласом Россом (SoftTech, Inc.) и называвшийся первоначально SADT – Structured Analysis and Design Technique<sup>1</sup>. В начале 70-х годов XX века вооруженные силы США применили подмножество SADT, касающееся моделирования процессов, для реализации проектов в рамках программы ICAM (Integrated Computer-Aided Manufacturing). В дальнейшем это подмножество SADT было принято в качестве федерального стандарта США под наименованием IDEF0.

В IDEF0 система представляется как совокупность взаимодействующих работ или функций. Такая чисто функциональная ориентация является принципиальной – функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации.

Под моделью в IDEF0 понимают описание системы (текстовое и графическое), которое должно дать ответ на некоторые заранее определенные вопросы.

Моделируемая система рассматривается как произвольное подмножество Вселенной. Произвольное потому, что, во-первых, мы сами умозрительно определяем, будет ли некий объект компонентом системы, или мы будем его рассматривать как внешнее воздействие, и, во-вторых, оно зависит от точки зрения на систему. Система имеет границу, которая отделяет ее от остальной Вселенной. Взаимодействие системы с окружающим миром описывается как вход (нечто, что перерабатывается системой), выход (результат деятельности системы), управление (стратегии и про-

цедуры, под управлением которых производится работа) и механизм (ресурсы, необходимые для проведения работы). Находясь под управлением, система преобразует входы в выходы, используя механизмы.

Процесс моделирования какой-либо системы в IDEF0 начинается с определения контекста, т. е. наиболее абстрактного уровня описания системы в целом. В контекст входит определение субъекта моделирования, цели и точки зрения на модель.

Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами, другими словами, мы должны определить, что мы будем в дальнейшем рассматривать как компоненты системы, а что как внешнее воздействие. На определение субъекта системы будет существенно влиять позиция, с которой рассматривается система, и цель моделирования – вопросы, на которые построенная модель должна дать ответ, другими словами, первоначально необходимо определить область моделирования. Описание области как системы в целом, так и ее компонентов является основой построения модели. Хотя предполагается, что в течение моделирования область может корректироваться, она должна быть в основном сформулирована изначально, поскольку именно область определяет направление моделирования и когда должна быть закончена модель. При формулировании области необходимо учитывать два компонента – широту и глубину. Широта подразумевает определение границ модели – мы определяем, что будет рассматриваться внутри системы, а что снаружи. Глубина определяет, на каком Уровне детализации модель является завершенной. При определении глубины системы необходимо не забывать об ограничениях времени: трудоемкость построения модели растет в геометрической прогрессии от глубины декомпозиции. После определения границ модели предполагается, что новые объекты не должны вноситься в моделируемую систему; поскольку все объекты модели взаимосвязаны, внесение нового объекта может быть не просто арифметической добавкой, но в состоянии изменить существующие взаимосвязи. Внесение таких изменений в готовую модель является, как правило, очень трудоемким процессом (так называемая проблема «плавающей области»).



Цель моделирования (Purpose). Модель не может быть построена без четко сформулированной цели. Цель должна отвечать на следующие вопросы:

- Почему этот процесс должен быть замоделирован?
- Что должна показывать модель?
- Что может получить читатель?

Формулировка цели позволяет команде аналитиков сфокусировать усилия в нужном направлении. Примерами формулирования цели могут быть следующие утверждения: «Идентифицировать и определить текущие проблемы, сделать возможным анализ потенциальных улучшений», «Идентифицировать роли и ответственность служащих для написания должностных инструкций», «Описать функциональность предприятия с целью написания спецификаций информационной системы» и т. д.

Точка зрения (Viewpoint). Хотя при построении модели учитываются мнения различных людей, модель должна строиться с единой точки зрения. Точку зрения можно представить как взгляд человека, который видит систему в нужном для моделирования аспекте. Точка зрения должна соответствовать цели моделирования. Очевидно, что описание работы предприятия с точки зрения финансиста и технолога будет выглядеть совершенно по-разному, поэтому в течение моделирования важно оставаться на выбранной точке зрения. Как правило, выбирается точка зрения человека, ответственного за моделируемую работу в целом.

Общий порядок разработки функциональной модели можно представить следующим образом:

1. Выделение функциональных блоков (функций процесса).
2. Выделение связей между функциями.

Начнем построение функциональной модели с описания первоначальной глобальной функции – разработки плана привлечения и размещения ресурсов банка и ее связей с внешним миром (рисунок 5.1).

Далее декомпозируем эту функцию на более мелкие функции, описывающие нужный нам процесс. Следующий уровень проектируемой функциональной модели будет состоять из 5 блоков (рисунок 5.2):

- консолидировать показатели планов ресурсов отделений;

- проверить показатели полученного сводного плана ресурсов;
- при наличии ошибки скорректировать показатели сводного плана ресурсов на основе данных сводного балансового отчета;
- если ошибок нет, то составить сводный план ресурсов банка;
- на основе сводного плана ресурсов банка составить окончательный вариант плана ресурсов отделений банка.

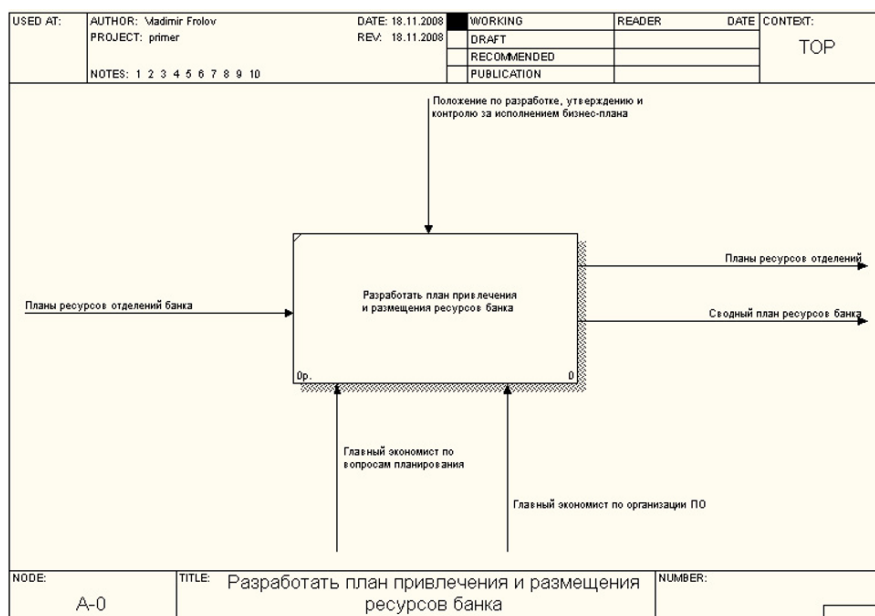


Рисунок 5.1 – Первый уровень функциональной модели

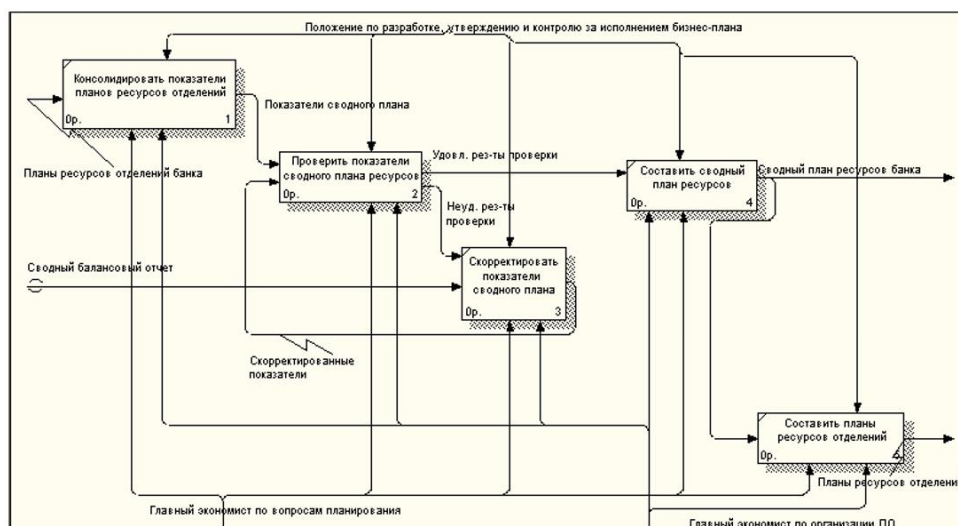


Рисунок 5.2 – Второй уровень функциональной модели

Продекомпозируем следующий блок функциональной модели – «проверить показатели сводного плана ресурсов». Следу-

ющий уровень декомпозиции будет состоять из трех функциональных блоков (рисунок 5.3):

- рассчитать соотношение привлеченных и размещенных ресурсов (размещенные ресурсы должны составлять не менее 85% от привлеченных ресурсов);
- рассчитать соотношение основных показателей сводного плана ресурсов (долю физических, юридических лиц, а также долю банка в привлечении и размещении ресурсов);
- проанализировать результаты проверки (проверить соотношение между привлекаемыми и размещаемыми ресурсами и т.д.).

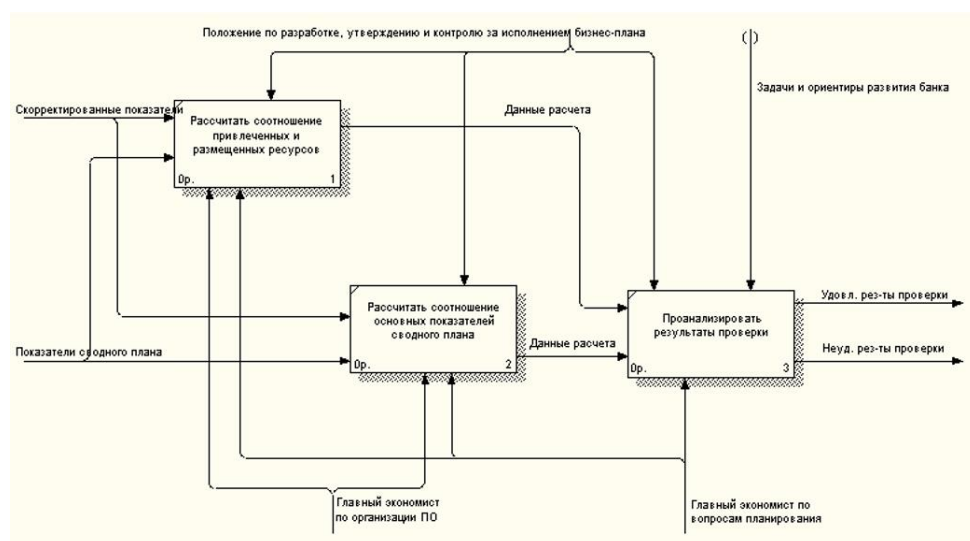


Рисунок 5.3 – Третий уровень функциональной модели

Функциональная модель заданной предметной области построена. Теперь следует проверить синтаксис полученной модели. Программа выдала список синтаксических ошибок (рисунок 5.4), показывающий, что на уровне декомпозиции диаграммы A0 имеется одна неразрешенная стрелка с названием «сводный балансовый отчет», на уровне декомпозиции диаграммы A2 также имеется неразрешенная стрелка с названием «задачи и ориентиры развития банка».

Данные стрелки следует сделать туннельными, так как они свойственны только для указанных уровней диаграммы и не должны появиться на верхних.

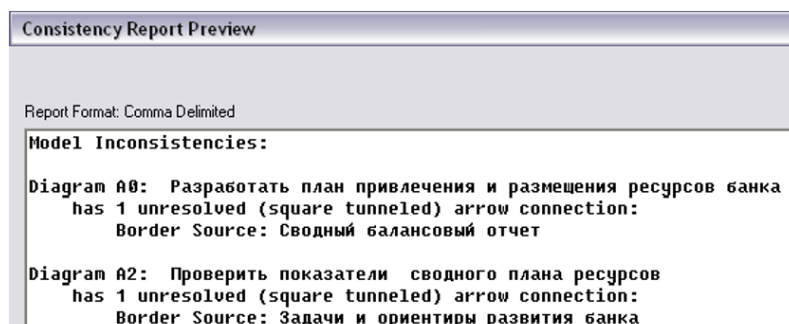


Рисунок 5.4 – Отчет по синтаксическим ошибкам модели

Отчет Node Tree (рисунок 5.5). На сформированном отчете Node Tree наглядно видно количество уровней декомпозиции построенной функциональной модели и отношение между родительскими и дочерними диаграммами.

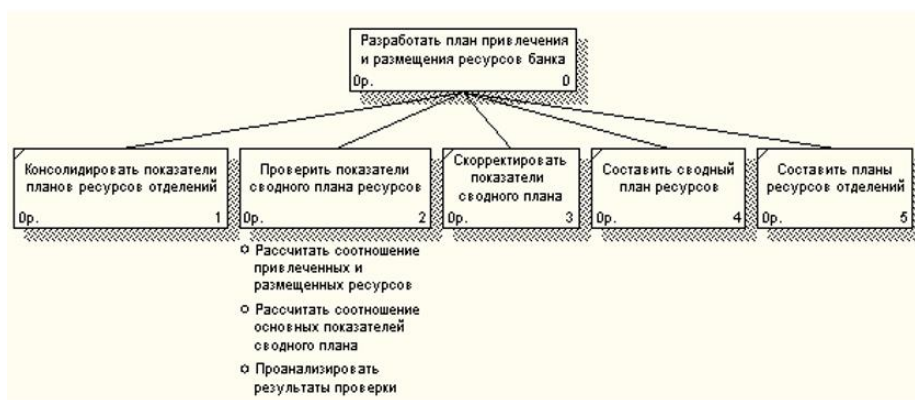


Рисунок 5.5 – Отчет Node Tree

Для описания логики взаимодействия информационных потоков более подходит IDEF3, называемая также WorkFlow Diagramming – методологией моделирования, использующая графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов. Диаграммы WorkFlow могут быть использованы в моделировании бизнес-процессов для анализа завершенности процедур обработки информации. С их помощью можно описывать сценарии действий сотрудников организации, например, последовательность обработки заказа события, которые необходимо обработать за конечное время. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования каждой функции.

IDEF3 – это метод, имеющий основной целью дать возможность аналитикам описать ситуацию, когда процессы выполня-

ются в определенной последовательности, а также описать объекты, участвующие совместно в одном процессе.

Техника описания набора данных IDEF3 является частью структурного анализа. В отличие от некоторых методик описаний процессов, IDEF3 не ограничивает аналитика чрезмерно жесткими рамками синтаксиса, что может привести к созданию неполных или противоречивых моделей.

IDEF3 может быть также использован как метод создания процессов. IDEF3 дополняет IDEF0 и содержит все необходимое для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа.

Каждая работа в IDEF3 описывает какой-либо сценарий бизнес-процесса и может являться составляющей другой работы. Поскольку сценарий описывает цель и рамки модели, важно, чтобы работы именовались отглагольным существительным, обозначающим процесс действия, или фразой, содержащей такое существительное.

Точка зрения на модель должна быть задокументирована. Обычно это точка зрения человека, ответственного за работу в целом. Также необходимо задокументировать цель модели – те вопросы, на которые призвана ответить модель.

Единицы работы – Unit of Work (UOW). UOW, также называемые работами (activity), являются центральными компонентами модели. В IDEF3 работы изображаются прямоугольниками с прямыми углами и имеют имя, выраженное отглагольным существительным, обозначающим процесс действия, одиночным или в составе фразы, и номер (идентификатор); другое имя существительное в составе той же фразы обычно отображает основной выход (результат) работы (например, «Изготовление изделия»). Часто имя существительное в имени работы меняется в процессе моделирования, поскольку модель может уточняться и редактироваться. Идентификатор работы присваивается при создании и не меняется никогда. Даже если работа будет удалена, ее идентификатор не будет вновь использоваться для других работ. Обычно номер работы состоит из номера родительской работы и порядкового номера на текущей диаграмме.

Работа в IDEF3 требует более подробного описания, чем работа в IDEF0. Каждая UOW должна иметь ассоциированный до-

кумент, который включает текстовое описание компонентов работы: объектов (Objects) и фактов (Facts), связанных с работой, ограничений (Constraints), накладываемых на работу, и дополнительное описание работы (Description). Эта информация заносится во вкладку UOW диалога Activity Properties.

Связи. Связи показывают взаимоотношения работ. Все связи в IDEF3 однонаправленны и могут быть направлены куда угодно, но обычно диаграммы IDEF3 стараются построить так, чтобы связи были направлены слева направо. В IDEF3 различают три типа стрелок, изображающих связи, стиль которых устанавливается во вкладке Style диалога Arrow Properties (пункт контекстного меню Style).


Старшая (Precedence) стрелка – сплошная линия, связывающая единицы работ (UOW). Рисуеться слева направо или сверху вниз. Показывает, что работа-источник должна закончиться прежде, чем работа-цель начнется.

Стрелка отношения (Relational Link) – пунктирная линия, используемая для изображения связей между единицами работ (UOW), а также между единицами работ и объектами ссылок.

Потоки объектов (Object Flow) – стрелка с двумя наконечниками, применяется для описания того факта, что объект используется в двух или более единицах работы, например, когда объект порождается в одной работе и используется в другой.

Старшая связь и поток объектов. Старшая связь показывает, что работа-источник заканчивается ранее, чем начинается работа-цель. Часто результатом работы-источника становится объект, необходимый для запуска работы-цели. В этом случае стрелку, обозначающую объект, изображают с двойным наконечником. Имя стрелки должно ясно идентифицировать отображаемый объект. Поток объектов имеет ту же семантику, что и старшая стрелка.

Отношение показывает, что стрелка является альтернативой старшей стрелке или потоку объектов в смысле задания последовательности выполнения работ – работа-источник не обязательно должна закончиться прежде, чем работа-цель начнется. Более того, работа-цель может закончиться прежде, чем закончится работа-источник.

Перекрестки (Junction). Окончание одной работы может служить сигналом к началу нескольких работ, или же одна работа для своего запуска может ожидать окончания нескольких работ. Перекрестки используются для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Различают перекрестки слияния (Fan-in Junction) и разветвления (Fan-out Junction) стрелок. Перекресток не может использоваться одновременно для слияния и для ветвления. Для внесения перекрестка служит кнопка  (добавить на диаграмму перекресток – Junction) в палитре инструментов. В диалоге Junction Type Editor необходимо указать тип перекрестка. Смысл каждого типа приведен в таблице 5.1.

Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс J. Можно редактировать свойства перекрестка при помощи диалога Junction Properties (вызывается из контекстного меню). В отличие от IDEF0 и DFD в IDEF3 стрелки могут сливаться и разветвляться только через перекрестки.




Объект ссылки. Объект ссылки в IDEF3 выражает некую идею, концепцию или данные, которые нельзя связать со стрелкой, перекрестком или работой. Для внесения объекта ссылки служит кнопка  (добавить в диаграмму объект ссылки – Referent) в палитре инструментов.

Таблица 5.1 – Типы перекрестков

Обозначение	Наименование	Смысл в случае слияния стрелок	Смысл в случае разветвления стрелок
	Асинхронное «И»	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
	Синхронное «И»	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно

Продолжение таблицы 5.1

Обозначение	Наименование	Смысл в случае слияния стрелок	Смысл в случае разветвления стрелок
	Асинхронное «ИЛИ»	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
	Синхронное «ИЛИ»	Один или несколько предшествующих процессов завершены одновременно	Один или несколько следующих процессов запускаются одновременно
	Исключающее «ИЛИ»	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Объект ссылки изображается в виде прямоугольника, похожего на прямоугольник работы. Имя объекта ссылки задается в диалоге Referent Properties (пункт контекстного меню Name), в качестве имени можно использовать имя какой-либо стрелки с других диаграмм или имя сущности из модели данных. Объекты ссылки должны быть связаны с единицами работ или перекрестками пунктирными линиями. Официальная спецификация IDEF3 различает три стиля объектов ссылок – безусловные (unconditional), синхронные (synchronous) и асинхронные (asynchronous). ВРwin поддерживает только безусловные объекты ссылок. Синхронные и асинхронные объекты ссылок, используемые в диаграммах переходов состояний объектов, не поддерживаются.

При внесении объектов ссылок помимо имени следует указывать тип объекта ссылки. Типы объектов ссылок приведены в таблице 5.2.



Таблица 5.2 – Типы объектов ссылок

Тип объекта ссылки	Цель описания
ОБЪЕСТ	Описывает участие важного объекта
GOTO	Инструмент циклического перехода (в повторяющейся последовательности работ), возможно на текущей диаграмме, но не обязательно. Если все работы цикла присутствуют на текущей диаграмме, цикл может также изображаться и стрелкой, возвращающейся на стартовую работу. GOTO может ссылаться на перекресток
UOB (Unit of Behavior)	Применяется, когда необходимо подчеркнуть множественное использование какой-либо работы, но без цикла. Например, работа «Контроль качества» может быть использована в процессе «Изготовление изделия» несколько раз, после каждой единичной операции. Обычно этот тип ссылки не используется для моделирования автоматически запускающихся работ
NOTE	Используется для документирования важной информации, относящейся к каким-либо графическим объектам на диаграмме. Является альтернативой внесению текстового объекта на диаграмму
ELAB (Elaboration)	Используется для усовершенствования графиков или их более детального описания. Обычно употребляется для детального описания разветвления и слияния стрелок на перекрестках

### 5.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Данное практическое занятие предполагает выполнение следующих этапов:

- изучить методические указания;
- создать функциональную модель и сценарий выбранного процесса;
- ответить на контрольные вопросы.

## 5.4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое бизнес-процесс?
2. Каковы основные компоненты функциональной модели?
3. Что представляют собой методологии функционального моделирования?
4. Что такое сценарии?
5. Какие виды сценариев Вы знаете?
6. В чем отличие серверных элементов управления от клиентских?
7. Какие технологии программирования серверных сценариев Вы знаете? В чем их отличие?
8. Для чего строится диаграмма IDEF3?
9. Чем диаграмма IDEF3 отличается от диаграммы IDEF0?
10. Как графически обозначается работа в диаграмме IDEF3?
11. С какой целью между работами устанавливают перекресток?
12. Какие типы перекрестков Вам знакомы?

## 6 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА И ОФОРМЛЕНИЕ ПРЕДЛОЖЕНИЙ ПО РАСШИРЕНИЮ ФУНКЦИОНАЛЬНОСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

### 6.1 ЦЕЛЬ РАБОТЫ

Цель работы – описать и проанализировать информационную систему, распределить роли в группе разработчиков.

### 6.2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Проблемы управления программными проектами впервые проявились в 60-х – начале 70-х годов, когда провалились многие большие проекты по разработке программных продуктов. Были зафиксированы задержки в создании программного обеспечения (ПО), оно было ненадежным, затраты на разработку в несколько раз превосходили первоначальные оценки, созданные программные системы часто имели низкие показатели производительности. Причины провалов коренились в тех подходах, которые использовались в управлении проектами. Применяемая методика была основана на опыте управления техническими проектами и оказалась неэффективной при разработке программного обеспечения.

Важно понимать разницу между профессиональной разработкой ПО и любительским программированием. Необходимость управления программными проектами вытекает из того факта, что процесс создания профессионального ПО всегда является субъектом бюджетной политики организации, где оно разрабатывается, и имеет временные ограничения. Работа руководителя программного проекта по большому счету заключается в том, чтобы гарантировать выполнение этих бюджетных и временных ограничений с учетом бизнес-целей организации относительно разрабатываемого ПО.

Руководители проектов призваны спланировать все этапы разработки программного продукта. Они также должны контролировать ход выполнения работ и соблюдения всех требуемых стандартов. Постоянный контроль за ходом выполнения работ необходим для того, чтобы процесс разработки не выходил за временные и бюджетные ограничения. Хорошее управление не

гарантирует успешного завершения проекта, но плохое управление обязательно приведет к его провалу. Это может выразиться в задержке сроков сдачи готового ПО, в превышении сметной стоимости проекта и в несоответствии готового ПО спецификации требований.

Процесс разработки ПО существенно отличается от процессов реализации технических проектов, что порождает определенные сложности в управлении программными проектами:

Программный продукт нематериален. Программное обеспечение нематериально, его нельзя увидеть или потрогать. Руководитель программного проекта не видит процесс «роста» разрабатываемого ПО. Он может полагаться только на документацию, которая фиксирует процесс разработки программного продукта.

Не существует стандартных процессов разработки ПО. На сегодняшний день не существует четкой зависимости между процессом создания ПО и типом создаваемого программного продукта. Другие технические дисциплины имеют длительную историю, процессы разработки технических изделий многократно опробованы и проверены. Процессы создания большинства технических систем хорошо изучены. Изучением же процессов создания ПО специалисты занимаются только последнее время. Поэтому пока нельзя точно предсказать, на каком этапе процесса разработки ПО могут возникнуть проблемы, угрожающие всему программному проекту.

Большие программные проекты – это часто «одноразовые» проекты. Большие программные проекты, как правило, значительно отличаются от проектов, реализованных ранее. Поэтому, чтобы уменьшить неопределенность в планировании проекта, руководители проектов должны обладать очень большим практическим опытом. Но постоянные технологические изменения в компьютерной технике и коммуникационном оборудовании обесценивают предыдущий опыт. Знания и навыки, накопленные опытом, могут не востребоваться в новом проекте.

Перечисленные отличия могут привести к тому, что реализация проекта выйдет из временного графика или превысит бюджетные ассигнования. Программные системы зачастую оказываются новинками, как в «идеологическом», так и в техническом плане. Поэтому, предвидя возможные проблемы в реализации

программного проекта, следует всегда помнить, что многим из них свойственно выходить за рамки временных и бюджетных ограничений.

#### 6.2.1 Процесс управления разработкой программного обеспечения

Невозможно описать и стандартизировать все работы, выполняемые в проекте по созданию ПО. Эти работы весьма существенно зависят от организации, где выполняется разработка ПО, и от типа создаваемого программного продукта. Но всегда можно выделить следующие:

1. Написание предложений по созданию ПО.
2. Планирование и составление графика работ по созданию ПО.
3. Оценивание стоимости проекта.
4. Подбор персонала.
5. Контроль за ходом выполнения работ.
6. Написание отчетов и представлений.

Первая стадия программного проекта может состоять из написания предложений по реализации этого проекта. Предложения должны содержать описание целей проектов и способов их достижения. Они также обычно включают в себя оценки финансовых и временных затрат на выполнение проекта. При необходимости здесь могут приводиться обоснования для передачи проекта на выполнение сторонней организации или команде разработчиков.

Написание предложений – очень ответственная работа, так как для многих организаций вопрос о том, будет ли проект выполняться самой организацией или разрабатываться по контракту сторонней компанией, является критическим. Не существует каких-либо рекомендаций по написанию предложений, многое здесь зависит от опыта.

На этапе планирования проекта определяются процессы, этапы и полученные на каждом из них результаты, которые должны привести к выполнению проекта. Реализация этого плана приведет к достижению целей проекта. Определение стоимости

проекта напрямую связано с его планированием, поскольку здесь оцениваются ресурсы, требующиеся для выполнения плана.

Контроль за ходом выполнения работ (мониторинг проекта) – это непрерывный процесс, продолжающийся в течение всего срока реализации проекта. Руководитель должен постоянно отслеживать ход реализации проекта и сравнивать фактические и плановые показатели выполнения работ с их стоимостью. Хотя многие организации имеют механизмы формального мониторинга работ, опытный руководитель может составить ясную картину о стадии развития проекта просто путем неформального общения с разработчиками.

Неформальный мониторинг часто помогает обнаружить потенциальные проблемы, которые в явном виде могут обнаружиться позднее. Например, ежедневное обсуждение хода выполнения работ может выявить отдельные недоработки в создаваемом программном продукте. Вместо ожидания отчетов, в которых будет отражен факт «пробуксовки» графика работ, можно обсудить со специалистами намечающиеся программистские проблемы и не допустить срыва графика работ.

В течение реализации проекта обычно происходит несколько формальных контрольных проверок хода выполнения работ по созданию ПО. Такие проверки должны дать общую картину хода реализации проекта в целом и показать, насколько уже разработанная часть ПО соответствует целям проекта.

Время выполнения больших программных проектов может занимать несколько лет. В течение этого времени цели и намерения организации, заказавшей программный проект, могут существенно измениться. Может оказаться, что разрабатываемый программный продукт стал уже ненужным либо исходные требования к создаваемому ПО просто устарели и их необходимо кардинально менять. В такой ситуации руководство организации-разработчика может принять решение о прекращении разработки ПО или об изменении проекта в целом с тем, чтобы учесть изменившиеся цели и намерения организации-заказчика.

Руководители проектов обычно обязаны сами подбирать исполнителей для своих проектов. В идеальном случае профессиональный уровень исполнителей должен соответствовать той работе, которую они будут выполнять в ходе реализации проекта.

Однако во многих случаях руководители должны полагаться на команду разработчиков, которая далека от идеальной. Такая ситуация может быть вызвана следующими причинами:

Бюджет проекта не позволяет привлечь высококвалифицированный персонал. В таком случае за меньшую плату привлекаются менее квалифицированные специалисты.

Бывают ситуации, когда невозможно найти специалистов необходимой квалификации, как в самой организации-разработчике, так и вне ее. Например, в организации «лучшие люди» могут быть уже заняты в других проектах.

Организация хочет повысить профессиональный уровень своих работников. В этом случае она может привлечь к участию в проекте неопытных или недостаточно квалифицированных работников, чтобы они приобрели необходимый опыт и поучились у более опытных специалистов.

Таким образом, почти всегда подбор специалистов для выполнения проекта имеет определенные ограничения и не является свободным. Вместе с тем необходимо, чтобы хотя бы несколько членов группы разработчиков имели квалификацию и опыт, достаточные для работы над данным проектом. В противном случае невозможно избежать ошибок в разработке ПО.

Руководитель проекта обычно обязан посылать отчеты о ходе его выполнения, как заказчику, так и подрядным организациям. Это должны быть краткие документы, основанные на информации, извлекаемой из подробных отчетов о проекте. В этих отчетах должна быть та информация, которая позволяет четко оценить степень готовности создаваемого программного продукта.

Выделим следующие роли в группе по разработке ПО:

1. Руководитель – общее руководство проектом, написание документации, общение с заказчиком ПО.
2. Системный аналитик – разработка требований (составление технического задания, проекта программного обеспечения).
3. Тестер – составление плана тестирования и аттестации готового ПО (продукта), составление сценария тестирования, базовый пример, проведение мероприятий по плану тестирования.
4. Разработчик – моделирование компонент программного обеспечения, кодирование.

### 6.2.2 Планирование проекта разработки программного обеспечения

Эффективное управление программным проектом напрямую зависит от правильного планирования работ, необходимых для его выполнения. План помогает руководителю предвидеть проблемы, которые могут возникнуть на каких-либо этапах создания ПО, и разработать превентивные меры для их предупреждения или решения. План, разработанный на начальном этапе проекта, рассматривается всеми его участниками как руководящий документ, выполнение которого должно привести к успешному завершению проекта. Этот первоначальный план должен максимально подробно описывать все этапы реализации проекта.

Процесс планирования начинается, исходя из описания системы, с определения проектных ограничений (временные ограничения, возможности наличного персонала, бюджетные ограничения и т.д.). Эти ограничения должны определяться параллельно с оцениванием проектных параметров, таких как структура и размер проекта, а также распределением функций среди исполнителей. Затем определяются этапы разработки и то, какие результаты документация, прототипы, подсистемы или версии программного продукта) должны быть получены по окончании этих этапов. Далее начинается циклическая часть планирования. Сначала разрабатывается график работ по выполнению проекта или дается разрешение на продолжение использования ранее созданного графика. После этого проводится контроль выполнения работ и отмечаются расхождения между реальным и плановым ходом работ.

Далее, по мере поступления новой информации о ходе выполнения проекта, возможен пересмотр первоначальных оценок параметров проекта. Это, в свою очередь, может привести к изменению графика работ. Если в результате этих изменений нарушаются сроки завершения проекта, должны быть пересмотрены (и согласованы с заказчиком ПО) проектные ограничения.

Конечно, большинство руководителей проектов не думают, что реализация их проектов пройдет гладко, без всяких проблем. Желательно описать возможные проблемы еще до того, как они проявят себя в ходе выполнения проекта. Поэтому лучше состав-



лять «пессимистические» графики работ, чем «оптимистические». Но, конечно, невозможно построить план, учитывающий все, в том числе случайные, проблемы и задержки выполнения проекта, поэтому и возникает необходимость периодического пересмотра проектных ограничений и этапов создания программного продукта.

План проекта должен четко показать ресурсы, необходимые для реализации проекта, разделение работ на этапы и временной график выполнения этих этапов. В некоторых организациях план проекта составляется как единый документ, содержащий все виды планов, описанных выше. В других случаях план проекта описывает только технологический процесс создания ПО. В таком плане обязательно присутствуют ссылки на планы других видов, но они разрабатываются отдельно от плана проекта.

Детализация планов проектов очень разнится в зависимости от типа разрабатываемого программного продукта и организации-разработчика. Но в любом случае большинство планов содержат следующие разделы.

1. Введение. Краткое описание целей проекта и проектных ограничений (бюджетных, временных и т.д.), которые важны для управления проектом.

2. Организация выполнения проекта. Описание способа подбора команды разработчиков и распределение обязанностей между членами команды.

3. Анализ рисков. Описание возможных проектных рисков, вероятности их проявления и стратегий, направленных на их уменьшение.

4. Аппаратные и программные ресурсы, необходимые для реализации проекта. Перечень аппаратных средств и программного обеспечения, необходимого для разработки программного продукта. Если аппаратные средства требуется закупать, приводится их стоимость совместно с графиком закупки и поставки.

5. Разбиение работ на этапы. Процесс реализации проекта разбивается на отдельные процессы, определяются этапы выполнения проекта, приводится описание результатов («выходов») каждого этапа и контрольные отметки.

6. График работ. В этом графике отображаются зависимости между отдельными процессами (этапами) разработки ПО,

оценки времени их выполнения и распределение членов команды разработчиков по отдельным этапам.

7. Механизмы мониторинга и контроля за ходом выполнения проекта. Описываются предоставляемые руководителем отчеты о ходе выполнения работ, сроки их предоставления, а также механизмы мониторинга всего проекта.

План должен регулярно пересматриваться в процессе реализации проекта. Одни части плана, например, график работ, изменяются часто, другие более стабильны. Для внесения изменений в план требуется специальная организация документопотока, позволяющая отслеживать эти изменения.

### 6.2.3 Общие сведения о требованиях к информационным системам

Проблемы, которые приходится решать специалистам в процессе создания программного обеспечения, очень сложны. Природа этих проблем не всегда ясна, особенно если разрабатываемая программная система инновационная. В частности, трудно чётко описать те действия, которые должна выполнять система. Описание функциональных возможностей и ограничений, накладываемых на систему, называется требованиями к этой системе, а сам процесс формирования, анализа, документирования и проверки этих функциональных возможностей и ограничений – разработкой требований.

Требования подразделяются на пользовательские и системные. Пользовательские требования – это описание на естественном языке (плюс поясняющие диаграммы) функций, выполняемых системой, и ограничений, накладываемых на неё. Системные требования – это описание особенностей системы (архитектура системы, требования к параметрам оборудования и т.д.), необходимых для эффективной реализации требований пользователя.

Первые шаги по разработке требований к информационным системам – анализ осуществимости.

Разработка требований – это процесс, включающий мероприятия, необходимые для создания и утверждения документа, содержащего спецификацию системных требований. Для новых программных систем процесс разработки требований должен

начинаться с анализа осуществимости. Началом такого анализа является общее описание системы и ее назначения, а результатом анализа – отчет, в котором должна быть четкая рекомендация, продолжать или нет процесс разработки требований проектируемой системы. Другими словами, анализ осуществимости должен осветить следующие вопросы.

1. Отвечает ли система общим и бизнес-целям организации-заказчика и организации-разработчика?

2. Можно ли реализовать систему, используя существующие на данный момент технологии и не выходя за пределы заданной стоимости?

3. Можно ли объединить систему с другими системами, которые уже эксплуатируются?

Критическим является вопрос, будет ли система соответствовать целям организации. Если система не соответствует этим целям, она не представляет никакой ценности для организации. В то же время многие организации разрабатывают системы, не соответствующие их целям, либо не совсем ясно понимая эти цели, либо под влиянием политических или общественных факторов.

Выполнение анализа осуществимости включает сбор и анализ информации о будущей системе и написание соответствующего отчета. Сначала следует определить, какая именно информация необходима, чтобы ответить на поставленные выше вопросы. Например, эту информацию можно получить, ответив на следующее:

4. Что произойдет с организацией, если система не будет введена в эксплуатацию?

5. Какие текущие проблемы существуют в организации и как новая система поможет их решить?

6. Каким образом система будет способствовать целям бизнеса?

7. Требуется ли разработка системы технологии, которая до этого не использовалась в организации?

Далее необходимо определить источники информации. Это могут быть менеджеры отделов, где система будет использоваться, разработчики программного обеспечения, знакомые с типом будущей системы, технологи, конечные пользователи и т.д.

После обработки собранной информации готовится отчет по анализу осуществимости создания системы. В нем должны быть даны рекомендации относительно продолжения разработки системы. Могут быть предложены изменения бюджета и графика работ по созданию системы или предъявлены более высокие требования к системе.

### 6.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Данное практическое занятие предполагает выполнение следующих этапов:

- изучить методические указания;
- составить подробное описание информационной системы;
- провести анализ осуществимости;
- распределить роли в группе;
- заполнить разделы плана;
- ответить на контрольные вопросы.

### 6.4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите сложности при управлении программными проектами.
2. В чем заключается этап написания предложений?
3. Поясните этап мониторинг проекта.
4. Для чего используется неформальный мониторинг?
5. Поясните анализ осуществимости.
6. Что понимается под пользовательскими требованиями?
7. Что понимается под системными требованиями?

## 7 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА ПЕРЕЧНЯ ОБУЧАЮЩЕЙ ДОКУМЕНТАЦИИ НА ИНФОРМАЦИОННУЮ СИСТЕМУ

### 7.1 ЦЕЛЬ РАБОТЫ

Цель работы – изучить вопросы, которые охватывает план документирования и перечень необходимой документации.

### 7.2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Начальным этапом проекта внедрения компьютерно-интегрированных систем (КИС) является подготовка. В контексте данной фазы формулируются цели и задачи, а также готовятся шаблоны документов и укрупненный план-график проекта.

Основным документом этапа служит устав, определяющий цели проекта, а также содержащий его функциональный, организационный, технический и методологический объемы. Кроме того, документ описывает участников проекта и задает порядок согласования соответствующей документации.

Подготавливается концепция обучения проектной группы, включающая предлагаемый подход к обучению команды внедрения КИС заказчика. Шаблоны документов, используемые для подготовки документации на последующих этапах проекта, формируются здесь же.

Документатор должен подготовить план документирования, в котором должны быть определены задания, выполняемые при создании конкретной документации. Данный план должен быть официально согласован заказчиком, что подтверждает полный учет в этом плане всех требований заказчика.

Обычно план документирования должен охватывать весь комплект документации, например руководства пользователя, диалоговую документацию, справочные тексты и краткие справочные карты.

В плане документирования должны быть четко описаны область применения и ограничения по использованию планируемой документации, а также основные решения по анализу и проектированию этой документации. В плане также должны быть опре-

делены процессы и проверки, выполняемые при разработке документации.

План документирования должен охватывать следующие вопросы (но не ограничиваться ими):

1) Рабочее наименование, назначение, область применения и ограничения по использованию планируемой документации.

2) Спецификацию стиля.

3) Определение аудитории пользователей.

4) Обоснование причин использования документации данной аудиторией и ее целевое назначение.

5) Содержание (план-проспект) документации, с оценкой ее постраничного объема, и соответствующие уточнения для других машинных носителей документации.

6) Номенклатуру поставки – число печатных копий, наличие электронных копий, форматы дисков и файлов (включая версии программных средств) и откуда они могут быть поставлены.

7) Установление собственника авторских прав на документацию и любых других прав собственности. Вопрос прав собственности является сложным. Во всех договорах на документацию должны быть указаны собственники соответствующих прав. При этом может быть указана последующая возможность передачи авторских прав от документатора к заказчику. Передача авторских прав целесообразна при определении места и способа тиражирования документации.

8) Обеспечение перевода документации на другие языки.

9) Уровни (грифы) секретности и конфиденциальности (при необходимости).

10) Процедуры и проверки, могущие влиять на процесс разработки документации, включая, при необходимости, хранение, поиск, резервирование, передачу и оценку качества.

11) Методы и средства производства (тиражирования) и используемые версии данных средств.

12) Структуру коллектива разработчиков документации и, возможно, плана выбора данной структуры. Конкретные лица привлекаются на различных этапах написания и производства (тиражирования) документации в зависимости от уровня своего

опыта и знаний. Например, может быть необходимым хорошее знание автором документируемой системы и опыт в написании документации; для редактора может потребоваться только опыт редактирования, но не знание системы; от компоновщика (оформителя) может не требоваться других знаний кроме знания средств оформления.

13) Взаимосвязи (подчиненности) проекта.

14) Почасовую загрузку и зарплату персонала.

15) Требования к проектным ресурсам, включая информационные и прочие ресурсы, представляемые заказчиком, и срокам их представления.

16) Метод передачи документатору информации об изменениях программного средства в процессе его разработки.

17) Планы контроля изменений и сопровождения документации.

18) Планы проверки документации после ее создания.

19) Календарное планирование (графики) по контрольным точкам (milestones), включая (при необходимости):

- утверждение плана документирования,
- подготовку, проверку и корректировку проекта каждого документа,
- тестирование на практичность,
- подготовку оригиналов фотошаблонов,
- распечатку, переплетение и распространение документации.

При необходимости каждый из пунктов плана должен быть описан для каждого элемента документации.

Полезно также включить в план документирования образцы аналогичной документации, выпускаемой документатором или другими сторонами, чтобы показать ее стили и компоновку.

План документирования должен быть подготовлен и утвержден до начала разработки документации, чтобы гарантировать согласование всеми сторонами поставленных задач и используемых методов. После утверждения плана он должен быть доведен до всех заинтересованных сторон, включая персонал разработчиков документации, а также заказчика и субподрядчиков (например, печатников, наборщиков, переводчиков).

План документирования должен включать определение аудитории(й) пользователей документации, уровня образования, способностей, подготовки, опыта данных пользователей и других характеристик, связанных с содержанием, структурой и использованием документации.

Обычно имеется ряд различных групп пользователей, преследующих различные цели при использовании конкретной документации. Каждый тип пользователей, включая их характеристики и задачи, решаемые ими при помощи документации, должен быть определен отдельно.

Данные об определении аудитории пользователей могут быть получены из:

- результатов изучения аудитории, проведенного заказчиком или документатором;
- описаний, представляемых заказчиком;
- определений аудитории, полученных из других источников.

По возможности персонал разработчиков документации должен организовать встречи с типовыми пользователями в их рабочей обстановке и обследовать их работу.

Создание такого плана требует выполнения следующих задач:

- идентификация конечных пользователей и определение с помощью матрицы Job Role необходимого каждому отдельному пользователю объема обучения;
- определение требований к пользовательской документации;
- распределение ответственности за подготовку материалов и составление графика.

Все эти действия выполняются на основе плана документации, подготовленного на этапе концептуального планирования.

Для облегчения этой задачи в SAP предусмотрены шаблоны процедур для конечных пользователей, которые подготавливаются на основе более шестисот заранее стандартных ВРР (см. раздел «Подготовка плана обучения пользователей и пользовательской документации»).

Обучение конечных пользователей имеет огромное значение для успешного запуска системы, причем деятельность по подго-



товке обучения в первую очередь связана с созданием пользовательской документации, для чего необходимо выполнить следующие задачи:

- идентифицировать методику обучения и необходимые материалы;
- распределить ответственность и утвердить график составления обучающих материалов;
- распределить ответственность и утвердить график составления материалов для инструкторов, которые будут проводить обучение;
- распределить ответственность и утвердить график проведения обучения;
- организовать оповещение конечных пользователей об обучении и их регистрация.

Обучение в зависимости будущих обязанностей пользователя можно организовать, используя Информационную базу данных (Information Database, InfoDB), в которой содержится список курсов и соответствующие роли пользователей системы.

Подготовка материалов для обучения включает в себя создание обучающих слайдов, распечатку экранов, подготовку упражнений и буклетов и т. д.

Для конечных пользователей необходимо подготовить полный комплект дополнительных материалов, подсказок, сборников часто задаваемых вопросов и т. д.

### 7.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Данное практическое занятие предполагает выполнение следующих этапов:

- изучить методические указания;
- ответить на контрольные вопросы.

### 7.4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что включает план документации для конечных пользователей?
2. Какие задачи решаются при создании пользовательской документации?
3. Типовые этапы реализации проектов по внедрению ИС.

4. Что могут включать материалы для обучения?
5. Какие вопросы должен охватывать план документирования?

## 8 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА РУКОВОДСТВА ОПЕРАТОРА

### 8.1 ЦЕЛЬ РАБОТЫ

Цель работы – изучение структуры разделов руководства оператора по ГОСТ 19.505-79.

### 8.2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

В 2004-2005 годах был опубликован минимально необходимый набор «учебно-тренировочных» документов на программы, включающий техническое задание на программу по ГОСТ 19.201-78, программу и методику испытаний по ГОСТ 19.301-79, руководство оператора по ГОСТ 19.505-79. Этого достаточно для разработки программы, проведения испытаний и сдачи ее заказчику.

Рассмотрим структуру разделов руководства оператора по ГОСТ 19.505-79.

Структуру и оформление документа устанавливают в соответствии с ГОСТ 19.105-78.

Составление информационной части (аннотации и содержания) является обязательным

В аннотации целесообразно привести следующую фразу: «Настоящее руководство распространяется исключительно на программу и не заменяет учебную, справочную литературу, руководства от производителя ОС и прочие источники информации, освещающие работу с графическим пользовательским интерфейсом операционной системы». Допустимо создание подраздела «Назначение руководства» или «Рекомендации по освоению».

Руководство оператора должно содержать следующие разделы:

- назначение программы;
- условия выполнения программы;
- выполнение программы;
- сообщения оператору.

В зависимости от особенностей документы допускается объединять отдельные разделы или вводить новые.

Последняя фраза предоставляет разработчикам программной документации пространство для маневра.

В разделе «Назначение программы» должны быть указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации «...должны быть указаны сведения о назначении программы». Сведения о назначении программы изложены в основополагающем документе – в техническом задании.

Функциональным назначением программы является предоставление пользователю возможности работы с текстовыми документами в формате rtf.

Эксплуатационное назначение. Программа должна эксплуатироваться в профильных подразделениях на объектах заказчика.

Пользователями программы должны являться сотрудники профильных подразделений объектов заказчика.

Состав функций. Программа обеспечивает возможность выполнения перечисленных ниже функций:

- функции создания нового (пустого) файла;
- функции открытия (загрузки) существующего файла;
- функции редактирования открытого (далее – текущего) файла путем ввода, замены, удаления содержимого файла с применением стандартных устройств ввода;
- функции редактирования текущего файла с применением буфера обмена операционной системы;
- функции сохранения файла с исходным именем;
- функции сохранения файла с именем, отличным от исходного;
- функции отправки содержимого текущего файла электронной почтой с помощью внешней клиентской почтовой программы;
- функции вывода оперативных справок в строковом формате (подсказок);
- функции интерактивной справочной системы;
- функции отображения названия программы, версии программы, копирайта и комментариев разработчика.

Условия выполнения программы. В разделе «Условия выполнения программы» должны быть указаны условия, необходи-

мые для выполнения программы (минимальный и (или) максимальный состав аппаратурных и программных средств и т.п.).

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

Минимальный состав технических средств. В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий в себя:

- процессор Pentium-1000 с тактовой частотой, ГГц – 10, не менее;
- материнскую плату с FSB, ГГц – 5, не менее;
- оперативную память объемом, Тб – 10, не менее;
- и т. д.

Минимальный состав программных средств. Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы. Допускается использование пакета обновления.

Требования к персоналу (пользователю). Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц – системный администратор и пользователь программы – оператор.

Системный администратор должен иметь высшее профильное образование и сертификаты компании-производителя операционной системы. В перечень задач, выполняемых системным администратором, должны входить:

- задача поддержания работоспособности технических средств;
- задачи установки (инсталляции) и поддержания работоспособности системных программных средств – операционной системы;
- задача установки (инсталляции) программы.

Пользователь программы (оператор) должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

Персонал должен быть аттестован на II квалификационную группу по электробезопасности (для работы с конторским оборудованием).

Выполнение программы. В разделе «Выполнение программы» должна быть указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузки и управляет выполнением программы, а также ответы программы на эти команды.

Сообщения оператору. В данном разделе должны быть приведены тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия оператора (действия оператора в случае сбоя, возможности повторного запуска программы и т.п.).

Завершение работы программы. Ключевая фраза подраздела «Требования к количеству и квалификации персонала» технического задания – «пользователь программы (оператор) должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы» снимает с автора обязанность подробно расписывать способы загрузки и запуска программы... Не обязан разработчик разжевывать оператору приемы работы с графическим пользовательским интерфейсом операционной системы. За исключением случаев применения в программе элементов интерфейса, не свойственных операционной системе.

### 8.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Данное практическое занятие предполагает выполнение следующих этапов:

- изучить методические указания;
- ответить на контрольные вопросы.

### 8.4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какой стандарт устанавливает требования к содержанию и оформлению руководства оператора?
2. Перечислите разделы руководства оператора.
3. Что указывается в разделе «Назначение программы»?
4. Что указывается в разделе «Условия выполнения программы»?

5. Что указывается в разделе «Выполнение программы»?
6. Что указывается в разделе «Сообщения оператору»?

## 9 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. РАЗРАБОТКА МОДЕЛЕЙ ИНТЕРФЕЙСОВ ПОЛЬЗОВАТЕЛЕЙ

### 9.1 ЦЕЛЬ РАБОТЫ

Цель работы – изучить процесс создания интерфейса пользователя.

### 9.2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Интерфейс пользователя является одним из важнейших элементов программы, это та часть программы, которая находится у всех на виду. Недочеты в пользовательском интерфейсе могут серьезно испортить впечатление даже о самых многофункциональных программах. Именно поэтому разработке и проектированию пользовательского интерфейса нужно уделять особое внимание.

Некоторые программисты склонны оставлять дизайн интерфейса пользователя на потом, считая, что реальное достоинство приложения его программный код, который и требует большего внимания. Однако часто возникает недовольство пользователей из-за неудачно подобранных шрифтов, непонятного содержимого экрана и скорости его прорисовывания, поэтому работу над интерфейсом также нужно воспринимать серьезно.

Для чего нужна разработка пользовательского интерфейса? Как минимум, для этого есть две причины:

- чтобы пользователи работали более продуктивно, программа должна быть простой в использовании;
- хороший интерфейс может стать преимуществом против конкурентов, плохой послужить причиной неудачи всего проекта.

Процесс создания интерфейса начинается с определения целей проекта, а также внутренних и внешних обстоятельств, которые вы должны принять во внимание. Для того чтобы правильно расставить приоритеты, необходимо учитывать:

- опыт работы пользователей с компьютером, типовые ситуации использования;
- какая информация необходима и когда, какие результаты должны быть получены;



– технологию разработки и платформа, на которой будут работать пользователи.

### 9.2.1 Начальная фаза разработки. Концептуальный дизайн

В этой фазе разработки Вы должны решить, какой интерфейс лучше всего будет подходить для достижения ваших целей текстовый, графический или мультимедиа. Затем необходимо выбрать структуру взаимодействия, которая обеспечивает разные степени гибкости для пользователей. Обычно, чем гибче структура, тем больше она требует от пользователя обучения, понимания, и времени на работу с окнами (открыть, закрыть, разместить и т.д.).

Для выполнения начальной фазы разработки необходимо погрузиться целиком в задачи пользователей и создать бумажный прототип навигационной модели. Навигационная модель показывает, как необходимо распределять функции или задачи между окнами вашей программы, она определяет, как пользователи смогут перемещаться как между различными задачами, так и внутри отдельной задачи.

Для того чтобы создать хороший интерфейс, на каждой стадии разработки необходима обратная связь от пользователей. Чтобы оценить концептуальную модель программы, необходимо показать ее схему пользователям и попросить объяснить ее вам. Если у пользователя возникнут трудности, значит, вы еще не достигли точки зрения пользователя в понимании проблемы.

### 9.2.2 Визуальный дизайн. Использование компонентов

Хорошо выполненный дизайн выглядит чистым, простым и аккуратным. Его можно понять одним взглядом. Пользователь должен сразу распознавать, какие данные можно редактировать, какие нет; по каким объектам можно щелкать мышью и какие объекты можно перетаскивать.

Работа с несколькими формами. Если интерфейс пользователя должен содержать несколько форм, вам предстоит принять самое важное решение: какой использовать вид интерфейса одно документный (SDI) или многодокументный (MDI).

В SDI-приложениях окна форм появляются совершенно независимо друг от друга (примером такого интерфейса может служить программа «Блокнот (Notepad)» или графический редактор MS Paint), в MDI-приложениях вы можете одновременно работать с несколькими объектами (примером такого интерфейса может служить текстовый процессор MS Word). Однако не имеет значения какой тип интерфейса SDI или MDI выбран; взаимодействие пользователя с формами происходит одинаково посредством обработки событий, поступающих от элементов управления формы. Поэтому, если в вашем приложении предусмотрено несколько форм, программу необходимо написать так, чтобы у пользователей не было возможности нарушить предписанные ход ее выполнения (например, у пользователя не должно быть средств вывести форму, для которой еще не готова информация).

### 9.2.3 Эффективные меню

Еще одна важная часть разработки форм создание содержательных и эффективных меню. Приведем некоторые важные рекомендации:

- Следуйте стандартным соглашениям о расположении пунктов меню принятым в Windows File, Edit, View, и т.д.
- Группируйте пункты меню в логическом порядке и по содержанию.
- Для группировки пунктов в раскрывающихся меню используйте разделительные линии.
- Избегайте избыточных меню.
- Избегайте пунктов меню верхнего уровня, не содержащих раскрывающихся меню.
- Не забывайте использовать символ троеточия для обозначения пунктов меню, активизирующих диалоговые окна.
- Обязательно используйте клавиатурные эквиваленты команд и «горячие» клавиши.
- Помещайте на панель инструментов часто используемые команды меню.

Когда есть видимость работы приложения, пользователи более легко переносят длительное ожидание в работе программы.

Один из способов информирования пользователя о ходе выполнения работы использовать в форме индикатор процесса.

### 9.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Данное практическое занятие предполагает выполнение следующих этапов:

- изучить методические указания;
- ответить на контрольные вопросы.

### 9.4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Почему разработке пользовательского интерфейса уделяется особое внимание?
2. В чем заключается начальная стадия разработки интерфейса?
3. Рекомендации по созданию эффективного меню.
4. Особенности работы с несколькими формами.
5. Виды интерфейсов пользователей.

## 10 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. НАСТРОЙКА ДОСТУПА К СЕТЕВЫМ УСТРОЙСТВАМ

### 10.1 ЦЕЛЬ РАБОТЫ

Цель работы – изучение настроек доступа к сетевым устройствам.

### 10.2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Раньше для удаленной настройки сетевых устройств в основном применялся протокол Telnet. Однако он не обеспечивает шифрование информации, передаваемой между клиентом и сервером, что позволяет анализаторам сетевых пакетов перехватывать пароли и данные конфигурации.

Secure Shell (SSH) — это сетевой протокол, устанавливающий безопасное подключение с эмуляцией терминала к маршрутизатору или иному сетевому устройству. Протокол SSH шифрует все сведения, которые поступают по сетевому каналу, и предусматривает аутентификацию удаленного компьютера.

Протокол SSH все больше заменяет Telnet — именно его выбирают сетевые специалисты в качестве средства удаленного входа в систему. Чаще всего протокол SSH применяется для входа в удаленное устройство и выполнения команд, но может также передавать файлы по связанным протоколам SFTP или SCP.

Чтобы протокол SSH мог работать, на сетевых устройствах, взаимодействующих между собой, должна быть настроена поддержка SSH.

В локальной сети подключение обычно устанавливается с помощью Ethernet и IP.

Доступ к сетевым устройствам по протоколу SSH.

Настройка основных параметров устройств. Требуется настроить топологию сети и основные параметры, такие как IP-адреса интерфейсов, доступ к устройствам и пароли на маршрутизаторе.

Шаг 1: Создайте сеть согласно топологии.

Шаг 2: Выполните инициализацию и перезагрузку маршрутизатора и коммутатора.

Шаг 3: Настройте маршрутизатор.

a. Подключитесь к маршрутизатору с помощью консоли и активируйте привилегированный режим EXEC.

b. Войдите в режим конфигурации.

c. Отключите поиск DNS, чтобы предотвратить попытки маршрутизатора неверно преобразовывать введенные команды таким образом, как будто они являются именами узлов.

d. Назначьте class в качестве зашифрованного пароля привилегированного режима EXEC.

e. Назначьте cisco в качестве пароля консоли и включите режим входа в систему по паролю.

f. Назначьте cisco в качестве пароля VTU и включите вход по паролю.

g. Зашифруйте открытые пароли.

h. Создайте баннер, который предупреждает о запрете несанкционированного доступа.

i. Настройте и активируйте на маршрутизаторе интерфейс G0/1, используя информацию, приведенную в таблице адресации.

j. Сохраните текущую конфигурацию в файл загрузочной конфигурации.

Шаг 4: Настройте компьютер PC-A.

a. Настройте для PC-A IP-адрес и маску подсети.

b. Настройте для PC-A шлюз по умолчанию.

Шаг 5: Проверьте подключение к сети. Пошлите с PC-A команду Ping на маршрутизатор R1. Если эхо-запрос с помощью команды ping не проходит, найдите и устраните неполадки подключения.

Настройка маршрутизатора для доступа по протоколу SSH. Подключение к сетевым устройствам по протоколу Telnet сопряжено с риском для безопасности, поскольку вся информация передается в виде открытого текста. Протокол SSH шифрует данные сеанса и обеспечивает аутентификацию устройств, поэтому для удаленных подключений рекомендуется использовать именно этот протокол.

Шаг 1: Настройте аутентификацию устройств.

При генерации ключа шифрования в качестве его части используются имя устройства и домен. Поэтому эти имена необходимо указать перед вводом команды crypto key.

a. Задайте имя устройства.

```
Router(config)# hostname R1
```

б. Задайте домен для устройства.

```
R1(config)# ip domain-name ccna-lab.com
```

Шаг 2: Создайте ключ шифрования с указанием его длины.

```
R1(config)# crypto key generate rsa modulus 1024
```

The name for the keys will be: R1.ccna-lab.com

% The key modulus size is 1024 bits

% Generating 1024 bit RSA keys, keys will be non-exportable...

[OK] (elapsed time was 1 seconds)

```
R1(config)#
```

\*Jan 28 21:09:29.867: %SSH-5-ENABLED: SSH 1.99 has been enabled

Шаг 3: Создайте имя пользователя в локальной базе учетных записей.

```
R1(config)# username admin privilege 15 secret adminpass
```

Уровень привилегий 15 дает пользователю права администратора.

Шаг 4: Активируйте протокол SSH на линиях VTY.

а. Активируйте протоколы Telnet и SSH на входящих линиях VTY с помощью команды transport input.

```
R1(config)# line vty 0 4
```

```
R1(config-line)# transport input telnet ssh
```

б. Измените способ входа в систему таким образом, чтобы использовалась проверка пользователей по локальной базе учетных записей.

```
R1(config-line)# login local
```

```
R1(config-line)# end
```

```
R1#
```

Шаг 5: Сохраните текущую конфигурацию в файл загрузочной конфигурации.

```
R1# copy running-config startup-config
```

Destination filename [startup-config]?

Building configuration...

[OK]

```
R1#
```

Шаг 6: Установите соединение с маршрутизатором по протоколу SSH.

а. Запустите Tera Term с PC-A.

b. Установите SSH-подключение к R1. Используйте имя пользователя `admin` и пароль `adminpass`. У вас должно получиться установить SSH-подключение к R1.

Настройка коммутатора для доступа по протоколу SSH. Настроить коммутатор в топологии для приема подключений по протоколу SSH, а затем установить SSH-подключение с помощью программы Tera Term.

Шаг 1: Настройте основные параметры коммутатора.

a. Подключитесь к коммутатору с помощью консольного подключения и активируйте привилегированный режим EXEC.

b. Войдите в режим конфигурации.

c. Отключите поиск DNS, чтобы предотвратить попытки маршрутизатора неверно преобразовывать введенные команды таким образом, как будто они являются именами узлов.

d. Назначьте `class` в качестве зашифрованного пароля привилегированного режима EXEC.

e. Назначьте `cisco` в качестве пароля консоли и включите режим входа в систему по паролю.

f. Назначьте `cisco` в качестве пароля VTY и включите вход по паролю.

g. Зашифруйте открытые пароли.

h. Создайте баннер, который предупреждает о запрете не-санкционированного доступа.

i. Настройте и активируйте на коммутаторе интерфейс VLAN 1.

j. Сохраните текущую конфигурацию в файл загрузочной конфигурации.

Шаг 2: Настройте коммутатор для соединения по протоколу SSH.

Для настройки протокола SSH на коммутаторе используйте те же команды, которые применялись для аналогичной настройки маршрутизатора.

a. Настройте имя устройства, как указано в таблице адресации.

b. Задайте домен для устройства.

`S1(config)# ip domain-name csna-lab.com`

c. Создайте ключ шифрования с указанием его длины.

`S1(config)# crypto key generate rsa modulus 1024`

d. Создайте имя пользователя в локальной базе учетных записей.

```
S1(config)# username admin privilege 15 secret adminpass
```

e. Активируйте протоколы Telnet и SSH на линиях VTY.

```
S1(config)# line vty 0 15
```

```
S1(config-line)# transport input telnet ssh
```

f. Измените способ входа в систему таким образом, чтобы использовалась проверка пользователей по локальной базе учетных записей.

```
S1(config-line)# login local
```

```
S1(config-line)# end
```

Шаг 3: Установите соединение с коммутатором по протоколу SSH.

Запустите программу Tera Term на PC-A, затем установите подключение по протоколу SSH к интерфейсу SVI коммутатора S1.

Настройка протокола SSH с использованием интерфейса командной строки (CLI) коммутатора Клиент SSH встроен в операционную систему Cisco IOS и может запускаться из интерфейса командной строки. Установить соединение с маршрутизатором по протоколу SSH, используя интерфейс командной строки коммутатора.

Шаг 1: Посмотрите доступные параметры для клиента SSH в Cisco IOS.

Используйте вопросительный знак (?), чтобы отобразить варианты параметров для команды ssh.

```
S1# ssh ?
```

```
-c Select encryption algorithm
```

```
-l Log in using this user name
```

```
-m Select HMAC algorithm
```

```
-o Specify options
```

```
-p Connect to this port
```

```
-v Specify SSH Protocol Version
```

```
-vrf Specify vrf name
```

```
WORD IP address or hostname of a remote system
```

Шаг 2: Установите с коммутатора S1 соединение с маршрутизатором R1 по протоколу SSH.



а. Чтобы подключиться к маршрутизатору R1 по протоколу SSH, введите команду – l admin. Это позволит вам войти в систему под именем admin. При появлении приглашения введите в качестве пароля adminpass.

S1# ssh - l admin 192.168.1.1

Password:

Да.

\*\*\*\*\*

Warning: Unauthorized Access is Prohibited!

\*\*\*\*\*

R1#

б. Чтобы вернуться к коммутатору S1, не закрывая сеанс SSH с маршрутизатором R1, нажмите комбинацию клавиш Ctrl+Shift+6. Отпустите клавиши Ctrl+Shift+6 и нажмите х. Отображается приглашение привилегированного режима EXEC коммутатора.

R1#

S1#

с. Чтобы вернуться к сеансу SSH на R1, нажмите клавишу Enter в пустой строке интерфейса командной строки. Чтобы увидеть окно командной строки маршрутизатора, нажмите клавишу Enter еще раз.

S1#

[Resuming connection 1 to 192.168.1.1 ... ]

R1#

д. Чтобы завершить сеанс SSH на маршрутизаторе R1, введите в командной строке маршрутизатора команду exit.

R1# exit

[Connection to 192.168.1.1 closed by foreign host]

S1#

### 10.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Данное практическое занятие предполагает выполнение следующих этапов:

- изучить методические указания;
- ответить на контрольные вопросы.

## 10.4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите решаемые задачи при настройке доступа к сетевым устройствам.
2. Для чего используется протокол SSH?
3. Перечислите необходимые ресурсы.
4. Основные шаги при настройке параметров устройств.
5. Последовательность настройки маршрутизатора.
6. Настройка маршрутизатора для доступа по протоколу SSH.
7. Создание ключа шифрования.
8. Как предоставить доступ к сетевому устройству нескольким пользователям, у каждого из которых есть собственное имя пользователя?
9. Как настроить доступ по SSH и создать несколько пользователей при помощи команды `username`?
10. Настройка коммутатора для доступа по протоколу SSH.

## 11 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. НАСТРОЙКА ПОЛИТИКИ БЕЗОПАСНОСТИ

### 11.1 ЦЕЛЬ РАБОТЫ

Цель работы – приобретение необходимого объема знаний и практических навыков в области политики безопасности.

### 11.2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Политика безопасности системы является одной из важнейших составляющих в обеспечении надежной и защищенной работы операционной системы. Существует несколько способов для получения доступа к политикам безопасности локального компьютера.

Необходимо учитывать, что локальная политика безопасности используется только в старших версиях операционной системы Windows: корпоративной, профессиональной и для образовательных учреждений.

Для выполнения настройки локальных политик безопасности необходимо войти в Windows с учетной записью администратора или у вас должны быть права администратора.

Далее приведены способы получения доступа к локальной политике безопасности в Windows 10 или Windows 11.

1. Открытие окна локальной политики безопасности через поиск Windows.

Панель поиска Windows помогает найти нужные файлы, папки и приложения на устройстве, в том числе и системные инструменты. Для открытия окна локальной политики безопасности требуется выполнить следующие действия:

- щелкните по значку поиска на панели задач, чтобы открыть поиск Windows;
- в строке поиска введите выражение «локальная политика безопасности»;
- откройте локальную политику безопасности.
- на экране компьютера откроется редактор локальной политики безопасности (рисунок 11.1), в котором вносятся необходимые изменения в параметрах безопасности компьютера.

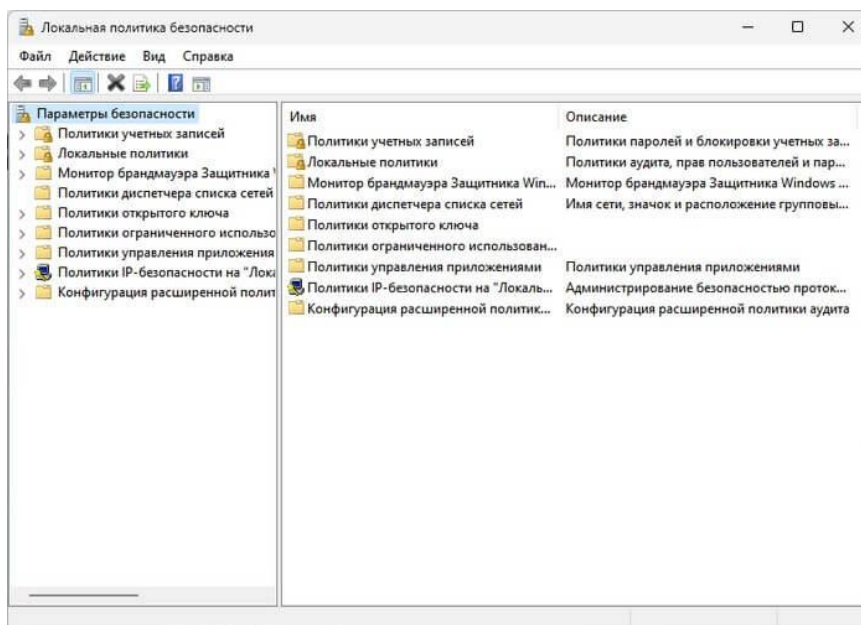


Рисунок 11.1 – Окно «Локальная политика безопасности»

2. Запуск локальной политики безопасности с помощью команды «Выполнить».

С помощью выполнения команд в окне «Выполнить» открываются программы или системные средства. Вызывает приложение «Локальная политика безопасности» команда «secpol.msc»:

- нажмите на клавиатуре «Win» + «R»;
- в диалоговое окно «Выполнить» введите команду «msc» (рисунок 11.2), а затем нажмите кнопку диалогового окна «ОК» или на клавиатуре «Enter».

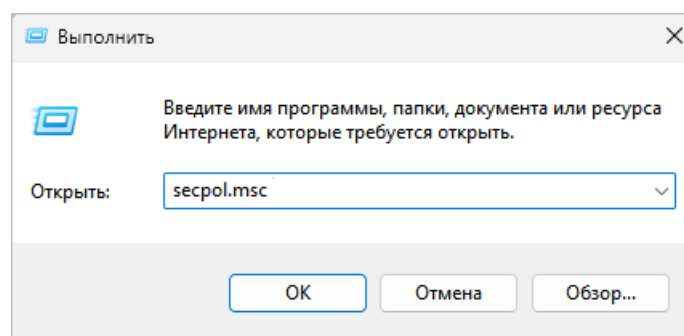


Рисунок 11.2 – Окно «Выполнить»

3. Открытие локальной политики безопасности с помощью меню «Пуск»

Локальная политика безопасности Windows 11 запускается следующим образом:

- откройте меню «Пуск», нажмите на кнопку «Все приложения» в правом верхнем углу;
- прокрутите страницу вниз, а затем щелкните по «Инструменты Windows»;
- в окне «Инструменты Windows» (рисунок 11.3) найдите «Локальная политика безопасности» и дважды щелкните по ней.

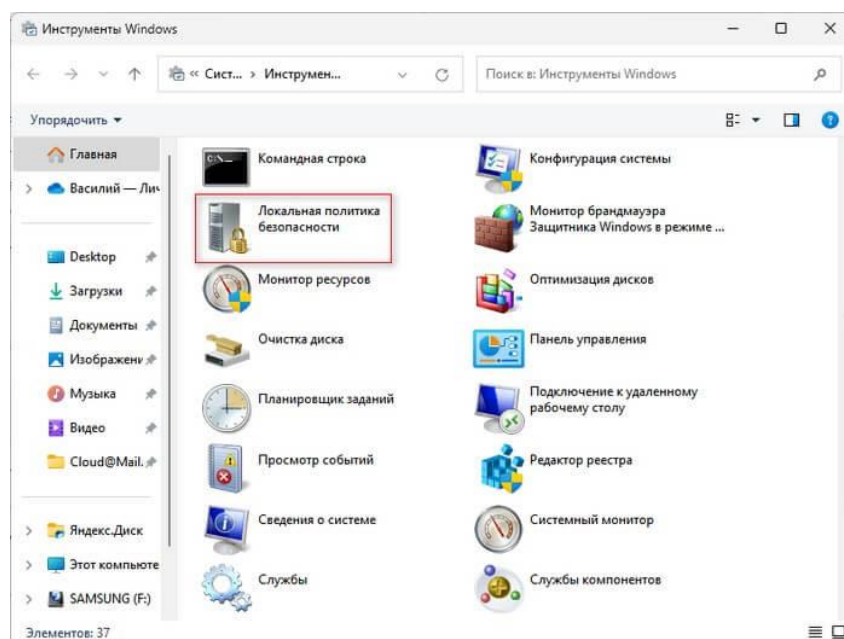


Рисунок 11.3 – Окно «Инструменты Windows»

Локальная политика безопасности Windows 10 открывается из меню «Пуск» таким способом:

- войдите в меню «Пуск»;
- в списке программ найдите папку «Средства администрирования Windows» (рисунок 11.4);
- щелкните по значку «Локальная политика безопасности» (рисунок 11.4).

4. Вход в локальную политику безопасности через «Панель управления».

Панель управления Windows обеспечивает доступ к различным системным инструментам и настройкам операционной системы.

Для входа в локальную политику безопасности следует выполнить следующие шаги:

- нажмите на клавиши «Win» + «R»;

- введите «control panel» в диалоговом окне «Выполнить» и нажмите кнопку «ОК»;
- в окне «Все элементы панели управления» выберите представление для просмотра крупными или мелкими значками;
- нажмите на «Инструменты Windows»;
- в окне «Инструменты Windows» дважды щелкните по значку «Локальная политика безопасности».

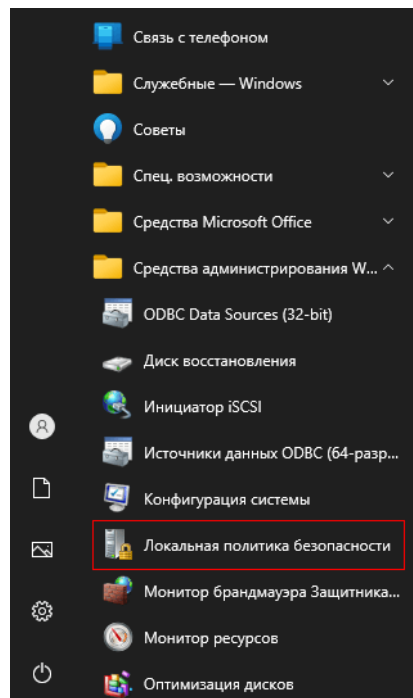


Рисунок 11.4 - Меню кнопки «Пуск»

5. Открытие локальной политики безопасности с помощью «Диспетчера задач».

Диспетчер задач Windows предоставляет сведения об активных процессах и программах, запущенных на компьютере, другую полезную информацию. Этот инструмент можно использовать не только для завершения или остановки процессов и служб, но и для запуска программ на вашем компьютере.

Выполните следующие действия:

- щелкните правой кнопкой мыши по меню «Пуск»;
- в контекстном меню выберите «Диспетчер задач»;
- во вкладке «Процессы» нажмите «Запустить новую задачу» на верхней панели в Windows, в Windows 10 откройте меню «Файл», а там кликните «Запустить новую задачу»;

- в открывшемся окне, в поле «Открыть:» введите «secpol.msc»;

- нажмите «ОК».

6. Переход к локальной политике безопасности из адресной строки Проводника

Проводник Windows имеет адресную строку, которую можно использовать для доступа к локальной политике безопасности.

Для перехода к локальной политике безопасности следуйте инструкции:

- откройте Проводник Windows;
- щелкните по адресной строке, введите «secpol.msc»;
- нажмите «Enter» или на стрелку справа от адресной строки.

7. Запуск локальной политики безопасности из системной папки Windows.

Имеется возможность запустить локальную политика безопасности непосредственно из папки «Windows», в которой располагается операционная система на компьютере. Для этого нужно выполнить следующие действия:

- откройте окно «Проводника»;
- перейдите в следующую папку «C:\Windows\System32»;
- найдите и дважды щелкните по файлу «secpol» для запуска системного средства.

8. Запуск утилиты локальной политики безопасности с помощью командной строки или PowerShell.

Можно применить методы командной строки для запуска локальной политики безопасности на своем ПК. С этой целью можно использовать командную строку или Windows PowerShell, в обоих средствах выполняется одинаковая команда.

Алгоритм работы:

- в поле поиска Windows введите «cmd» или «powershell»;
- откройте командную строку или Windows PowerShell;
- в окне системного средства введите «secpol», а затем нажмите на «Enter».

На экране вашего компьютера появится окно «Локальная политика безопасности».

9. Запуск локальной политики безопасности с помощью редактора локальной групповой политики

Можно использовать редактор локальной групповой политики для запуска локальной политики безопасности. Для этого требуется выполнить следующий набор шагов:

- в поле поиска Windows введите «gpedit.msc»;
- откройте окно «Редактор локальной групповой политики»;
- перейдите по пути «Конфигурация компьютера – Конфигурация Windows – Параметры безопасности»;
- здесь находятся все модули локальной политики безопасности.

10. Как закрепить локальную политику безопасности на панели задач

Можно открыть локальную политику безопасности, используя любой из перечисленных выше методов. Намного проще получить доступ к политикам безопасности прямо из панели задач. Для этого значок инструмента локальной политики безопасности необходимо закрепить на панели задач:

- нажмите на клавиши «Win» + «S»;
- в поле поиска введите «локальная политика безопасности»;
- в результатах поиска отобразится нужное приложение, выберите «Закрепить на панели задач».

Теперь можно запускать локальную политику безопасности непосредственно с панели задач Windows.

После запуска программы Назначение прав пользователя появится окно Локальные параметры безопасности (рисунок 11.5).



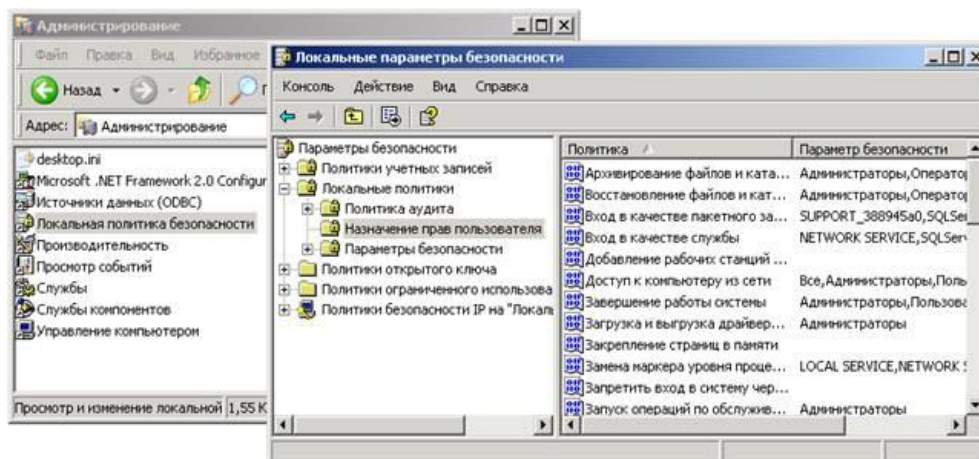


Рисунок 11.5 – Окно «Локальные параметры безопасности»

Основные пункты политики безопасности:

1. Пункт «Доступ к компьютеру из сети» – определяет, какие именно пользователи и группы пользователей могут получать доступ к данному компьютеру по компьютерной сети. Если компьютер не подключен к локальной сети, рекомендуется запретить доступ пользователей извне, это позволит избежать атак взломщиков и их проникновение в систему при работе в Интернете. Для запрета доступа сетевых пользователей к компьютеру следует:

- в окне «Политика программы – Назначение прав пользователя» щелчком мыши выбрать политику Доступ к компьютеру из сети;
- появится окно «Параметр локальной безопасности – Доступ к компьютеру из сети» (рисунок 11.6.);
- выделить всех пользователей (или лишних пользователей) при помощи указателя мыши и клавиши «Shift»;
- сделать щелчок по кнопке «Удалить»;
- нажать кнопку «ОК».

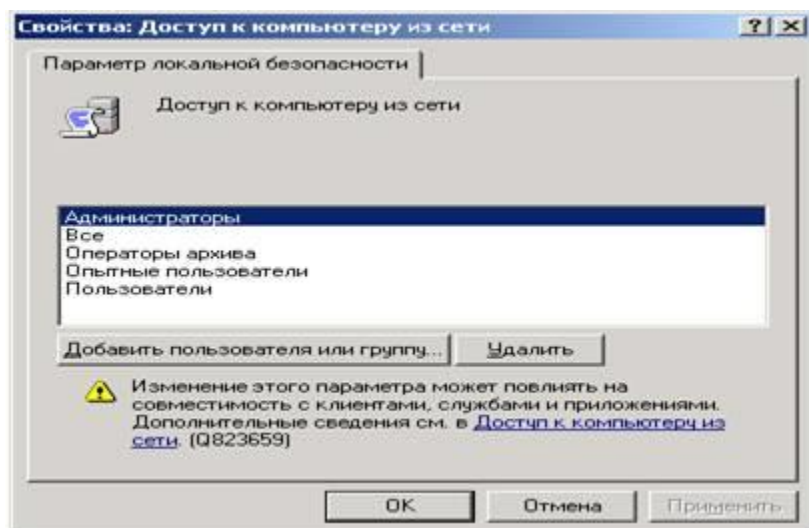


Рисунок 11.6 – Окно «Параметр локальной безопасности»

Пользователи, которым разрешен доступ к компьютеру, должны быть отображены в данном пункте политики безопасности, иначе они не смогут войти в систему. Если пользователи в списке окна отсутствуют, то их следует добавить при помощи кнопки «Добавить пользователя или группу». Для этого следует:

- сделать щелчок по кнопке «Добавить пользователя или группу»;
- в появившемся диалоговом окне сделать щелчок по кнопке «Дополнительно» (рисунок 11.7);
- в окне «Пользователи или группы» нажать кнопку «Поиск»;
- в нижней части окна появится список всех пользователей и групп;
- щелчком выбрать нужную строку нажать кнопку «ОК»;
- в появившемся диалоговом окне в поле «Введите имена выбираемых объектов», появится выбранный пользователь (группа), нажать кнопку «ОК»;
- выбранный пользователь (группа) будет отображен в окне «Доступ к компьютеру из сети».

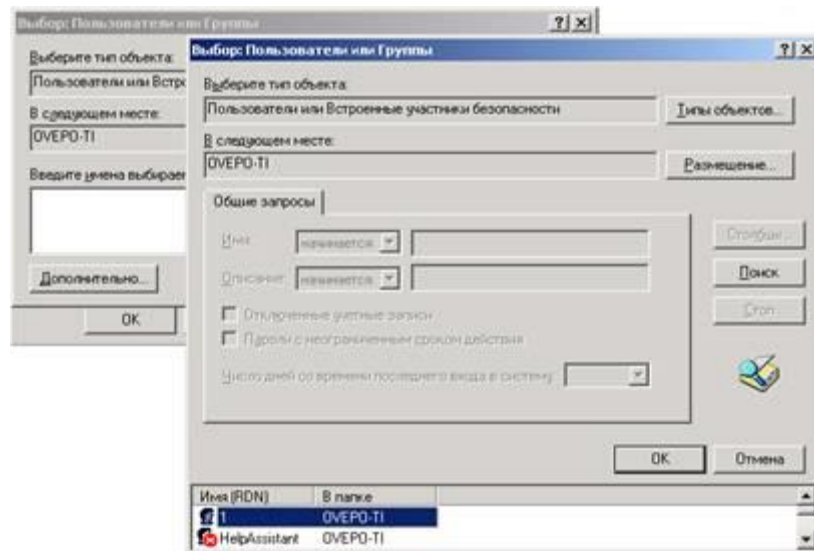


Рисунок 11.7 – Добавление пользователей или групп

2. Пункт «Разрешать вход в систему через службу терминалов» является аналогичным предыдущему, но вход пользователей в систему осуществляется в качестве клиентов терминал-сервера. Если данный сервис не используется, то рекомендуется аналогичным методом запретить вход в систему всех пользователей, убрав их из значения данного пункта как клиентов терминал-сервера. В случае необходимости всегда можно добавить нужных пользователей и их группы при помощи кнопки «Добавить пользователя или группу» (рисунок 11.8).

3. Пункт «Изменение системного времени», позволяющий пользователям, перечисленным в нем, менять системное время, а также просматривать календарь, появляющийся на экране при двойном щелчке по текущему времени на панели задач. По умолчанию данной возможностью обычные пользователи не смогут воспользоваться. Для разрешения пользователям выполнять такое действие следует их внести в список данного пункта политики безопасности при помощи кнопки «Добавить пользователя или группу» (рисунок 11.9).

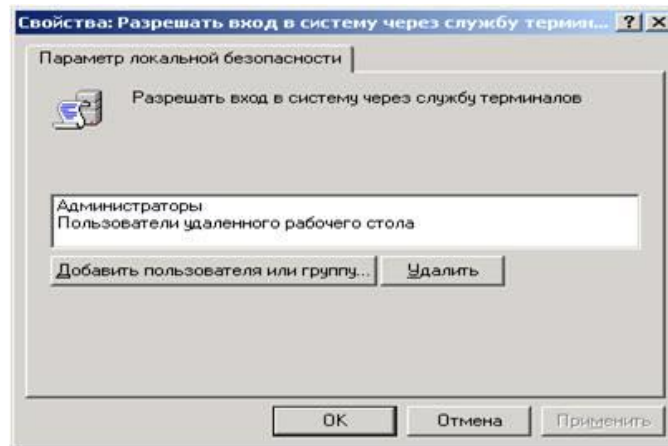


Рисунок 11.8 – Окно «Разрешить вход в систему через службу терминалов»

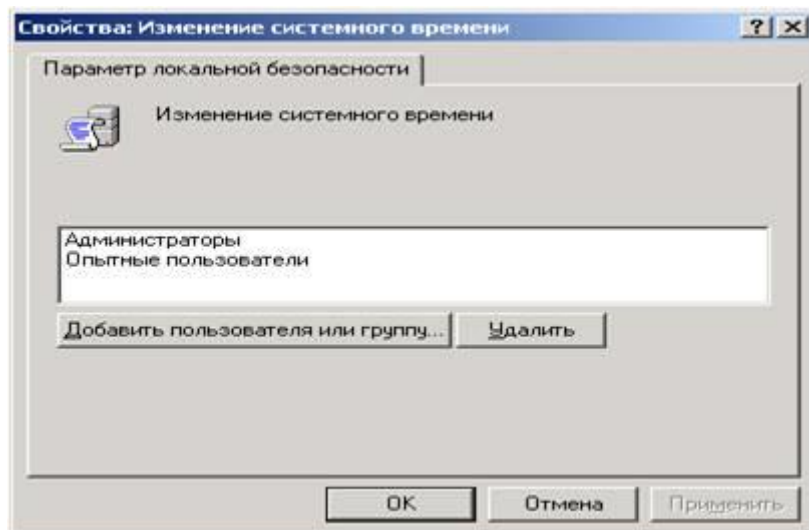


Рисунок 11.9 – Окно «Изменение системного времени»

4. Пункт «Отладка программ» позволяет указать пользователей, которые смогут подсоединять свой отладчик к процессам и производить их отладку. Следует включать в этот пункт только тех пользователей, которым это действительно нужно, например, системный администратор и системные программисты. Не следует давать это право другим пользователям, так как этой возможностью могут воспользоваться вирусы для заражения системы, запущенные под одной из пользовательских записей, имеющей право на отладку процессов.

5. Пункт «Отказ в доступе к компьютеру из сети» содержит пользователей и их группы, которым запрещен вход в систему по компьютерной сети. При необходимости можно добавить пользо-

вателей, которым запрещен доступ к компьютеру с помощью кнопки «Добавить пользователя или группу» (рисунок 11.10).

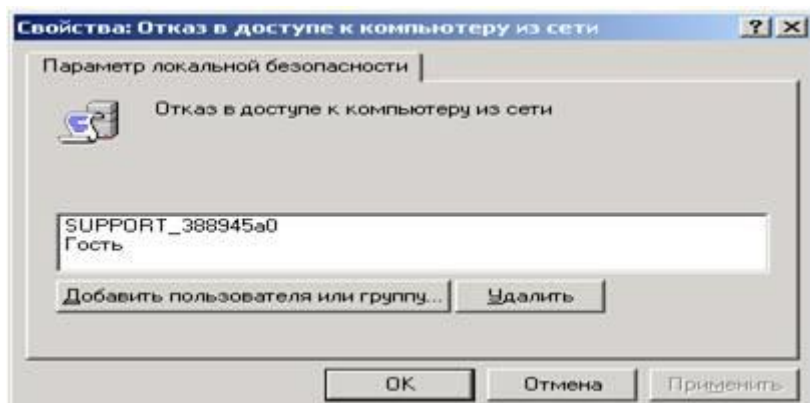


Рисунок 11.10 – Окно «Отказ в доступе к компьютеру из сети»

6. Пункт «Отклонить локальный вход» содержит пользователей и их группы, которым запрещен локальный вход в систему. При необходимости можно добавить пользователей, которым запрещен доступ к компьютеру с помощью кнопки «Добавить пользователя или группу» (рисунок 11.11).

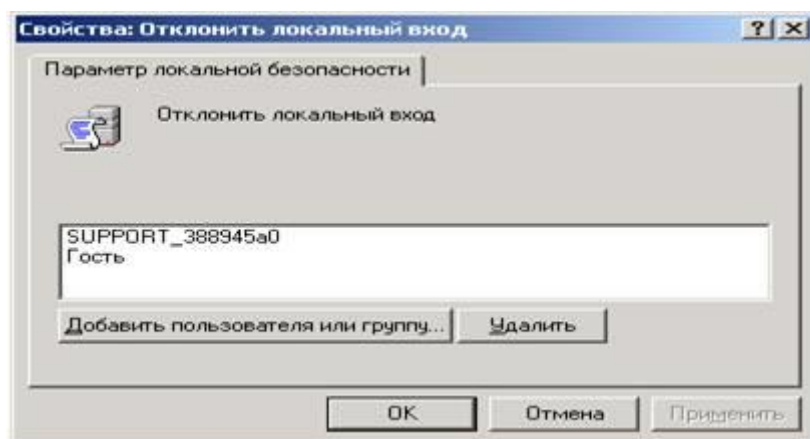


Рисунок 11.11 – Окно «Отклонить локальный вход»

7. Пункт «Запретить вход через службу терминалов» также содержит пользователей и их группы, которым запрещен вход в систему как клиентов терминал-сервера. При необходимости можно добавить пользователей, которым запрещен доступ к компьютеру с помощью кнопки «Добавить пользователя или группу» (рисунок 11.12).

С помощью трех перечисленных выше опций локальной политики безопасности можно запретить пользователям, которые по структуре организации не должны получать доступа, вход в

систему. Этим можно предотвратить внутренние коллизии организации и защитить данные от их искажения или разрушения пользователями, которые удаленно пытаются ими воспользоваться.

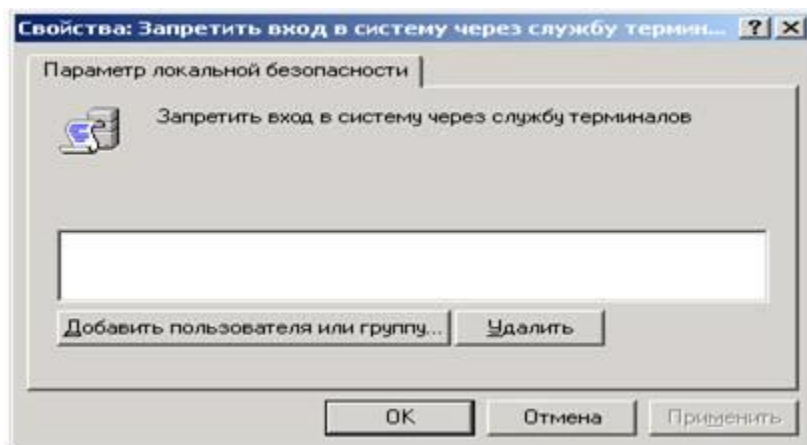


Рисунок 11.12 – Окно «Запретить вход в систему через службу терминалов»

8. Пункт «Принудительное удаленное завершение» является очень важным в настройке локальной политики безопасности, так как если его не настроить соответствующим образом, то система может получить команду на выключение или перезагрузку от удаленно пользователя. Поэтому в данном пункте следует указывать только пользователей, которым действительно может потребоваться с машин, находящихся в локальной сети, выключить или перезапустить систему (рисунок 11.13).

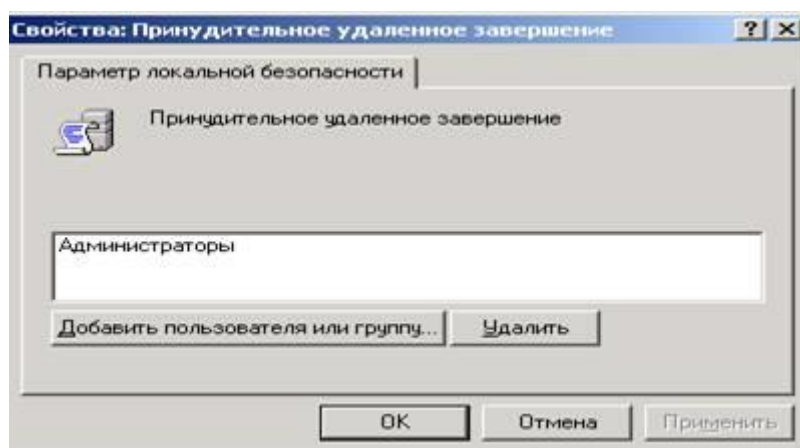


Рисунок 11.13 – Окно «Принудительное удаленное завершение»

9. Пункт «Загрузка и выгрузка драйверов устройств» позволяет указать, кто из пользователей может динамически установ-

ливать и выгружать драйвера устройств. Это право необходимо для установки драйверов устройств, имеющих спецификацию Plug and Play.

10. Пункт «Локальный вход в систему» является очень важным и определяет, какие пользователи и их группы могут локально входить в систему.

11. Пункт «Управление аудитом и журналом безопасности» относится к механизму аудита системы и определяет, какие пользователи и их группы могут устанавливать аудит доступа к определенным объектам, таким как файлы, ключи реестра и пр. По умолчанию в данном пункте перечислена лишь одна группа локальных системных администраторов.

12. Пункт «Изменение параметров среды оборудования» определяет пользователей, которые будут иметь право менять значения системных переменных. По умолчанию на это имеют право только пользователи, принадлежащие локальной группе администраторов.

13. Пункт «Запуск операций по обслуживанию тома» позволяет указать пользователей и их группы, которые будут иметь право выполнять задачи по поддержанию работы накопителей, такие как очистка диска или его дефрагментация. Выполнение данных задач, по умолчанию, доверяется только пользователям из группы системных администраторов.

14. Пункт «Восстановление файлов и каталогов» позволяет указывать пользователей и их группы, которые могут выполнять операцию восстановления файлов и директорий из сохраненных копий, а также ставить им необходимые права доступа. По умолчанию в системе такими пользователями являются члены группы системных администраторов, а также операторы сохранения данных.

15. Пункт «Завершение работы системы» указывает, кто из локальных пользователей, имеющих учетные записи в системе, имеет право на ее выключение или перезагрузку. По умолчанию на это имеют право все пользователи. Однако, в ряде случаев, может потребоваться запретить выполнять данные функции некоторым пользователям. Например, если нужно, чтобы компьютеры работали в то время, когда некоторые пользователи их пытаются отключить. В этом случае нужно убрать этих пользовате-



лей из данного пункта. Особенно это может быть полезно, если определенные пользователи пытаются выключить компьютер, на котором находится информация, используемая удаленно другими пользователями.

16. Пункт «Овладение файлами или иными объектами» отвечает за возможность пользователей, перечисленных в нем, брать на себя право становиться владельцами файлов и объектов. Этими объектами могут быть структуры Active Directory, ключи реестра, принтеры и процессы. По умолчанию на это имеют право только пользователи группы системных администраторов. Добавление к этому пункту пользователей означает предоставление им всех прав по доступу к различным объектам.

Глобальные параметры безопасности устанавливаются в разделе Локальной политики безопасности – Параметры безопасности (рисунк 11.14).

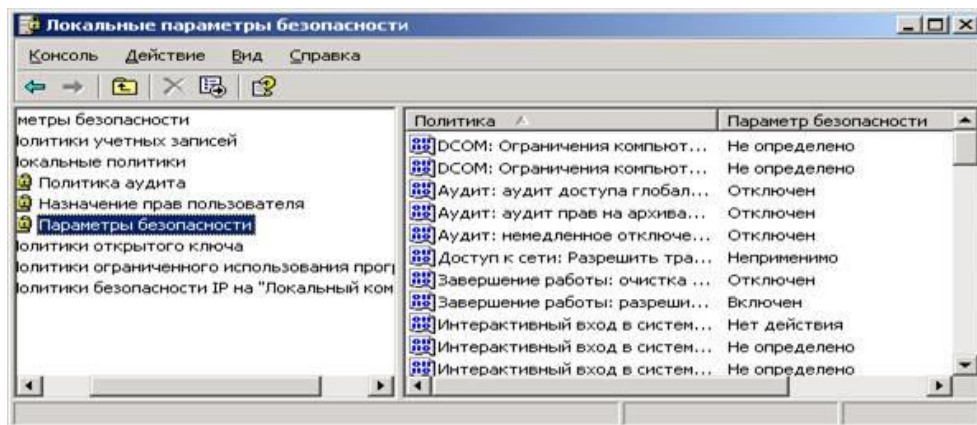


Рисунок 11.14 – Окно «Параметры безопасности»

Рассмотрим наиболее важные пункты.

1. Пункт «Учетные записи: Состояние учетной записи Администратор» предоставляет возможность выбора: будет ли учетная запись администратора системы включена или отключена, при нормальном функционировании системы. В случае использования системы в безопасном режиме запись администратора будет включена, независимо от значения данного пункта. Для изменения значения этого пункта следует его выбрать двойным щелчком мыши и в появившемся окне поставить флажок в соответствующем режиме (рисунк 11.15)



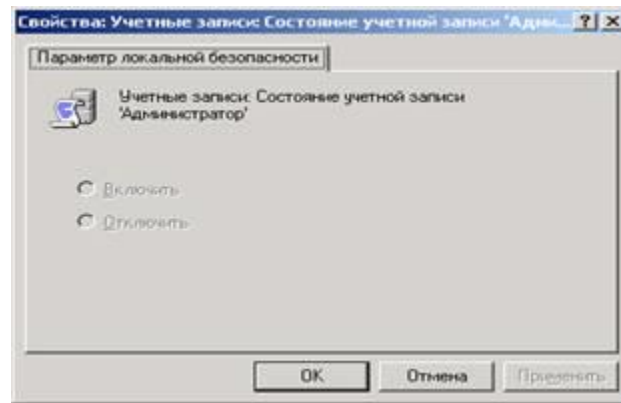


Рисунок 11.15 – Окно «Состояние учетной записи Администратор»

Отключение учетной записи системного администратора может быть полезно, т.к. это дает гарантированную защиту от атак взломщиков на эту учетную запись. Если необходимо включить учетную запись системного администратора, то это можно сделать под учетной записью другого пользователя, принадлежащего к группе системных администраторов, или в защищенном режиме работы операционной системы.

Пункт «Учетные записи: Состояние учетной записи Гость» позволяет отключать учетную запись гостя, т.к. для входа под данной учетной записью не требуется пароль, что может нарушить политику прав доступа пользователями. Учетная запись Гость по умолчанию отключена.

Пункт «Accounts: Limit local account use of blank passwords to console logon only», в случае включения позволяет ограничить доступ к незащищенным паролями консольным учетным записям локальных пользователей со стороны различных сетевых сервисов, например: терминал-сервера, Telnet и FTP. По умолчанию, в целях защиты системы от сетевых атак, данный пункт включен.

Пункт «Accounts: Rename administrator account» позволяет переименовать встроенную учетную запись администратора системы. Это делается в целях защиты от атаки методом подбора паролей. Чтобы изменить имя учетной записи администратора, нужно дважды щелкнуть мышью по имени этого пункта и в появившемся окне ввести новое имя этой учетной записи.

Пункт «Audit: Audit the use of Backup and Restore privilege» позволяет контролировать выполнение всех операций сохранения и восстановления данных. Система будет сохранять сообщения

обо всех резервируемых и восстанавливаемых файлах и папках. Это очень удобно для проведения контроля за операциями резервирования и восстановления данных. Для работы данного пункта необходимо включение в политике аудита опции Аудит использования привилегий. По умолчанию данный пункт локальной политики безопасности отключен.

Пункт «Audit: Shut down system immediately if unable to log security audits» является очень полезным и позволяет после своего включения, в случае обнаружения операционной системой невозможности производить запись событий аудита, произвести автоматическое выключение системы. Невозможность записи аудита событий системы обычно связана с переполнением хранилища этих сообщений. Для продолжения нормальной работы системы необходимо войти в нее под учетной записью администратора и произвести в программе «Администрирование – Просмотр событий» очистку всех этих сообщений, возможно, предварительно их сохранив. Это является гарантией того, что все действия системы или пользователей будут контролироваться администратором.

Пункт «Devices: Prevent users from installing printer drivers» – позволяет запретить пользователям устанавливать драйвера принтеров под их учетными записями.

Пункт «Devices: Unsigned driver installation behavior» позволяет указать поведение системе, при попытке пользователей установить драйвер, не прошедший процедуру сертификации Microsoft. Он может иметь три значения:

- Silent succeed – происходит инсталляция этого драйвера и никаких сообщений не выдается;
- Warn but allow installation – происходит предупреждение пользователя о том, что драйвер не прошел сертификацию, но инсталляция продолжается. Данный пункт используется по умолчанию;
- Do not allow installation – накладывается запрет на установку драйверов системы, не прошедших сертификацию.

Пункт «Interactive logon: Do not display last user name», в случае своего включения, запрещает показ системе имени пользователя, который в ней работал последним. Это удобно в тех случаях, когда нужно избежать подбора паролей взломщиками к

учетным записям пользователей системы, т.к. если у них не будет не только пароля, но и имени учетной записи пользователя, то их задача может стать в два раза сложнее. Данный пункт работает только в том случае, если отключен экран приветствия системы.

Пункт «Interactive logon: Do not require CTRL+ALT+DEL», в случае его выключения, производит отображение на экране таблички, требующей от пользователя нажатия комбинации клавиш «CTRL»+ «ALT»+ «DEL» для входа в систему. В случае включения этого пункта данное сообщение системы появляться не будет. Данный пункт работает только в том случае, если отключен экран приветствия. Смысл ввода этой комбинации клавиш для входа в систему заключается в том, что она обрабатывается только системой. И это гарантирует то, что в операционную систему входит человек, а не программа по подбору паролей пользователей. Таким образом, данное сообщение может быть дополнительным барьером, охраняющим систему от взломщиков.

Пункт «Interactive logon: Prompt user to change password before expiration» устанавливает количество дней до конца срока действия пароля пользователя, когда система будет предупреждать пользователя об этом. Данный пункт имеет смысл только в том случае, если пароли пользователей имеют определенный срок действия.

Пункт «Recovery console: Allow automatic administrative logon» устанавливает автоматический вход системного администратора в консоль восстановления системы. Это удобно тем, что не требует ввода пароля администратора, но по той же причине, создает большие проблемы с безопасностью, так как консолью восстановления с администраторскими правами сможет воспользоваться любой желающий.

Пункт «Recovery console: Allow floppy copy and access to all drives and all folders», в случае его включения, позволяет вам использовать команду SET консоли восстановления, которая может помочь установить следующие значения переменных:

- AllowWildCards – переменная включает поддержку масок у команд, например, DEL;
- AllowAllPath – переменная позволяет получить доступ ко всем файлам и папкам системы.

- AllowRemovableMedia – переменная позволяет копировать файлы на сменные носители, например, гибкие диски;
- NoCopyPrompt – переменная запрещает системе выдавать дополнительные сообщения при перезаписи существующего файла.

С помощью данного пункта можно скопировать или удалить информацию с жесткого диска системы. Поэтому не рекомендуется совмещать его использование с включенным предыдущим пунктом, позволяющим вход в консоль восстановления системы без администраторского пароля, т.к. можно лишиться всей информации.

Пункт «Shutdown: Allow system to be shut down without having to log on» позволяет, в случае его включения, производить выключение операционной системы до непосредственного входа в нее пользователями. Если вы не хотите, чтобы пользователи, не имеющие на это прав, выключали систему, установите данный пункт в положение Отключен.

Пункт «Shutdown: Clear virtual memory pagefile» является чрезвычайно важным в обеспечении безопасности вашей системы. При выключении системы в ее файле подкачки остаются данные, которые использовались в работе различными пользовательскими приложениями. Среди этих данных могут быть, частично или полностью, ваши документы, с которыми вы работали в течение сеанса работы с системой. Впоследствии, во время вашего отсутствия, эти данные могут быть кем-либо извлечены из файла подкачки. Таким образом, возможна утечка информации. И чтобы этого не случилось, системе может потребоваться очищать свой файл подкачки. Это можно сделать, включив данный пункт. Однако нужно учесть, время очистки файла займет некоторое дополнительное время, и система будет выключаться чуть дольше.

#### Политика обновления.

Любое программное обеспечение содержит ошибки (баги), т.к. на этапе проектирования приложений и систем невозможно все предусмотреть. Поэтому в любом приложении появляются места кода, которые работают не так, как рассчитывали разработчики, что может привести к нештатной работе программного

обеспечения, а также появлению новых ошибок или уязвимостей при его работе.

Для выявления ошибок все компании-разработчики стараются тестировать свое программное обеспечение, т.е. проверять работу программного обеспечения в шоковых для него условиях, когда его ограничивают в размере доступной памяти, дискового пространства, скорости работы центрального процессора и пр. На этом этапе вылавливаются ошибки и вносятся исправления в код программного обеспечения, улучшающие его стабильность (робастность) или отказоустойчивость. Однако эти меры лишь частично позволяют избавиться от наиболее явных ошибок, которые проявили себя в тестировании. В настоящее время не существует аппаратных или математических методов, позволяющих избавиться от ошибок в программном обеспечении на этапе его разработки.

После долгих поисков компании-разработчики нашли простой метод, который позволяет практически со стопроцентной вероятностью избавиться от ошибок в конечных продуктах, находящихся у пользователей. Этим методом является периодическое обновление программных продуктов. Компании разработчики решили, что в идеале программное обеспечение должно работать двадцать четыре часа в сутки, семь дней в неделю и в его работе не должно быть никаких нештатных ситуаций, вызванных ошибками. Это можно достигнуть с помощью грамотной политики обновления, которая используется практически в любом современном программном продукте, т.е. система периодически выходит в Интернет и проверяет сайт компании-разработчика на появление обновлений к программному обеспечению.

Компания-разработчик для программного продукта периодически помещает на своем сайте исправления, обновления и дополнения. Исправления – это специальные заплатки для программного обеспечения, которые исправляют существующие в нем ошибки, замеченные пользователями или специалистами компании. Обновления включают различные обновления программного продукта и заплатки от обнаруженных в нем ошибок. Дополнения добавляют программному продукту определенную функциональность.

Обновления для пользователей позволяют не только избежать ошибок ОС, проявляющихся при ее использовании, но и практически гарантированно защитить ее от взломщиков и вирусов, т.к. исправляются все замеченные ошибки в системе безопасности. Алгоритм работы системы обновления настроен на периодическую проверку сайта компании Microsoft на наличие различных обновлений и скачивание или предупреждение пользователя, в зависимости от его настроек. Все настройки политики безопасности находятся в программе «Пуск – Настройка – Панель управления – Система».

Все операции настройки политики обновления ОС, а также выполнения процедуры обновления и получения от нее различных сообщений возможны только под учетной записью администратора системы. После запуска программы появится окно, в котором нужно выбрать закладку «Автоматическое обновление (Automatic Updates)» (рисунок 11.16). На закладке можно установить один из четырех параметров, которые будут определять частоту обновления системы:

1. Автоматически (рекомендуется), Automatic (recommended) – параметр устанавливается операционной системой по умолчанию и означает регулярное обновление системы, заданное в двух нижерасположенных параметрах: частоты обновления и времени обновления. По умолчанию система будет обновляться каждый день в три часа. Скачивание обновлений операционной системы может происходить параллельно с работой в Интернете, т.к. операционной системой резервируется двадцать процентов пропускной способности канала связи с Интернетом, что позволяет быстро и незаметно скачивать системные обновления с сайта Microsoft.

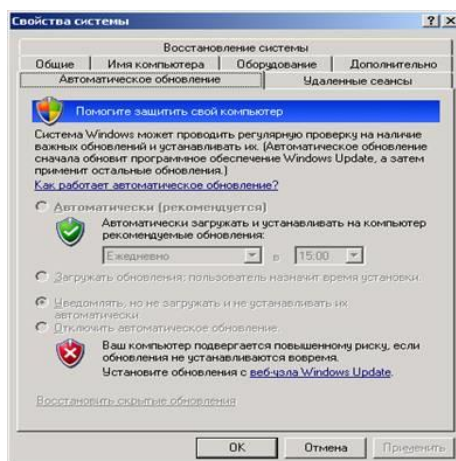


Рисунок 11.16 – Закладка Автоматическое обновление программы Система

Если пользователь работает на домашнем компьютере, достаточно производить обновления системы раз в неделю. Если система является корпоративной или часто находится в Интернете, рекомендуется проводить каждодневное обновление, чтобы надежно защититься от сетевых взломщиков. Если система скачает обновления, и они будут готовы к инсталляции, система сообщит об этом. Если же за время выхода в Интернет система не успеет скачать все обновления, они будут скачаны в следующий раз. Все скачанные и установленные обновления можно удалить, воспользовавшись для этого программой Установка и удаление программ: Пуск\Панель управления\Установка и удаление программ.

2. Загружать обновления, пользователь назначит время установки, Download updates for me, but let me choose when to install them опция позволяет операционной системе самостоятельно скачивать обновления, но для их установки она должна спросить разрешения у пользователя.

3. Уведомлять, но не загружать и не устанавливать их автоматически, Notify me but don't automatically download or install them опция позволяет операционной системе проверять наличие обновлений, но запрещает их непосредственное скачивание или установку. Эта опция полезна, если пользователь сам устанавливает обновления, например с компакт-диска, также она позволяет сэкономить на интернет-трафике.

4. Отключить автоматическое обновление, Turn off Automatic Updates опция запрещает работу системы обновления.

### 11.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Данное практическое занятие предполагает выполнение следующих этапов:

- изучить методические указания;
- произвести настройку Политики безопасности и обновления;
- ответить на контрольные вопросы.

### 11.4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Определите назначение политики безопасности системы.
2. Где производится настройка политики безопасности системы?
3. Как вызвать диалоговое окно для управления политикой безопасности.
4. Как запретить доступ сетевых пользователей к компьютеру?
5. Как разрешить доступ сетевым пользователям, которым разрешено работать в системе к компьютеру?
6. Определите назначения пункта политики безопасности «Разрешать вход в систему через службу терминалов».
7. Как предоставить определенной группе пользователей вносить изменения в системное время?
8. Определите назначение пункта политики безопасности «Отладка программ».
9. Каким образом запретить вход определенной группе пользователей в систему по локальной сети?
10. Определите назначение пункта политики безопасности Принудительное удаленное завершение.
11. Как установить пользователей и их группы, которые могут локально входить в систему?
12. Как запретить определенной группе пользователей завершать работу системы, и в каких случаях это актуально?
13. В каком разделе производится настройка глобальных параметров безопасности?
14. Определите назначение политики обновления.
15. Как произвести настройку политики обновления?



## 12 ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. ВЫПОЛНЕНИЕ ЗАДАЧ ТЕСТИРОВАНИЯ В ПРОЦЕССЕ ВНЕДРЕНИЯ.

### 12.1 ЦЕЛЬ РАБОТЫ

Цель работы – изучить классификацию видов тестирования, практически закрепить эти знания путем генерации тестов различных видов, научиться планировать тестовые активности в зависимости от специфики, поставляемой на тестирование функциональности.

### 12.2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Тестирование – процесс, направленный на оценку корректности, полноты и качества разработанного программного обеспечения. Тестирование можно классифицировать по очень большому количеству признаков.

#### 1. Типы тестов по покрытию (по глубине):

Smoke test – тестирование системы для определения корректной работы базовых функций программы в целом, без углубления в детали. При проведении теста определяется пригодность сборки для дальнейшего тестирования.

Minimal Acceptance Test (MAT, Positive test) – тестирование системы или ее части только на валидных данных (валидные данные – данные, которые необходимо использовать для корректной работы модуля/функции). При тестировании проверяется правильная работа всех функций и модулей с валидными данными.

Для крупных и сложных приложений используется ограниченный набор сценариев и функций.

Acceptance Test (AT) – полное тестирование системы или ее части как на корректных, так и на некорректных данных / сценариях.

Вид теста, направленный на подтверждение того, что приложение может использоваться по назначению при любых условиях. Тест на этом уровне покрывает все возможные сценарии тестирования:

- проверку работоспособности модулей при вводе корректных значений;

- проверку при вводе некорректных значений; использование форматов данных отличных от тех, которые указаны в требованиях;
- проверку исключительных ситуаций, сообщений об ошибках;
- тестирование на различных комбинациях входных параметров;
- проверку всех классов эквивалентности;
- тестирование граничных значений интервалов;
- сценарии не предусмотренные спецификацией и т.д.

## 2. Тестовые активности (типы тестов по ширине):

**Defect Validation** – проверка результата исправления дефектов. Включает в себя проверку на воспроизводимость дефектов, которые были исправлены в новой сборке продукта, а также проверку того, что исправление не повлияло на ранее работавшую функциональность.

**New Feature Test (NFT, AT of NF)** – определение качества поставленной на тестирование новой функциональности, которая ранее не тестировалась. Данный тип тестирования включает в себя: проведение полного теста (АТ) непосредственно новой функциональности; тестирование новой функциональности на соответствие документации; проверку всевозможных взаимодействий ранее реализованной функциональности с новыми модулями и функциями.

**Regression testing (регрессионное тестирование)** – проводится с целью оценки качества ранее реализованной функциональности. Включает в себя проверку стабильности ранее реализованной функциональности после внесения изменений, например добавления новой функциональности, исправление дефектов, оптимизация кода, разворачивание приложения на новом окружении.

Регрессионное тестирование может быть проведено на уровне Smoke, MAT или АТ.

## 3. Типы тестов по знанию коду.

**Черный ящик** – тестирование системы, функциональное или нефункциональное, без знания внутренней структуры и компонентов системы. У тестировщика нет доступа к внутренней

структуре и коду приложения либо в процессе тестирования он не обращается к ним.

Белый ящик – тестирование основанное на анализе внутренней структуры компонентов или системы. У тестировщика есть доступ к внутренней структуре и коду приложения.

Серый ящик – комбинация методов белого и черного ящика, состоящая в том, что к части кода архитектуры у тестировщика есть, а к части кода – нет.

#### 4. Типы тестов по степени автоматизации.

Ручное – тестирование, в котором тест-кейсы выполняются тестировщиком вручную без использования средств автоматизации.

Автоматизированное – набор техник, подходов и инструментальных средств, позволяющий исключить человека из выполнения некоторых задач в процессе тестирования. Тест-кейсы частично или полностью выполняет специальное инструментальное средство.

#### 5. Типы тестов по изолированности компонентов.

Unit component (модульное) – тестирование отдельных компонентов (модулей) программного обеспечения.

Integration (интеграционное) – тестируется взаимодействие между интегрированными компонентами или системами.

System (системное) – тестируется работоспособность системы в целом с целью проверки того, что она соответствует установленным требованиям.

#### 6. Типы тестов по подготовленности.

Интуитивное тестирование выполняется без подготовки к тестам, без определения ожидаемых результатов, проектирования тестовых сценариев.

Исследовательское тестирование – метод проектирования тестовых сценариев во время выполнения этих сценариев. Тестировщик совершает проверки, продумывает их, придумывает новые проверки, часто использует для этого полученную информацию.

Тестирование по документации – тестирование по подготовленным тестовым сценариям, руководству по осуществлению тестов.

#### 7. Типы тестов по месту и времени проведения.

User Acceptance Testing (UAT) (приемочное тестирование) – формальное тестирование по отношению к потребностям, требованиям и бизнес процессам пользователя, проводимое с целью определения соответствия системы критериям приёмки и дать возможность пользователям, заказчикам или иным авторизованным лицам определить, принимать систему.

Alpha Testing (альфа-тестирование) – моделируемое или действительное функциональное тестирование, выполняется в организации, разрабатывающей продукт, но не проектной командой (это может быть независимая команда тестировщиков, потенциальные пользователи, заказчики). Альфа тестирование часто применяется к коробочному программному обеспечению в качестве внутреннего приемочного тестирования.

Beta Testing (бета-тестирование) – эксплуатационное тестирование потенциальными или существующими клиентами/заказчиками на внешней стороне (в среде, где продукт будет использоваться) никак связанными с разработчиками, с целью определения действительно ли компонент или система удовлетворяет требованиям клиента/заказчика и вписывается в бизнес-процессы. Бета-тестирование часто проводится как форма внешнего приемочного тестирования готового программного обеспечения для того, чтобы получить отзывы рынка.

#### 8. Типы тестов по объекту тестирования.

Functional testing (функциональное тестирование) – это тестирование, основанное на анализе спецификации, функциональности компонента или системы. Функциональным можно назвать любой вид тестирования, который согласно требованиям проверяет правильную работу.

Safety testing (тестирование безопасности) – тестирование программного продукта с целью определить его безопасность (безопасность – способность программного продукта при использовании оговоренным образом оставаться в рамках приемлемого риска причинения вреда здоровью, бизнесу, программам, собственности или окружающей среде).

Security testing (тестирование защищенности) – это тестирование с целью оценить защищенность программного продукта. Тестирование защищенности проверяет фактическую реакцию защитных механизмов, встроенных в систему, на проникновение.

Compatibility testing (тестирование совместимости) – процесс тестирования для определения возможности взаимодействия программного продукта, проверка работоспособности приложения в различных средах (браузеры и их версии, операционные системы, их типа, версии и разрядность).

Виды тестов:

- кросс-браузерное тестирование (различные браузеры или версии браузеров);
- кросс-платформенное тестирование (различные операционные системы или версии операционных систем).

Нефункциональное тестирование – это проверка характеристик программы. Иначе говоря, когда проверяется не именно правильность работы, а какие-либо свойства (внешний вид и удобство пользования, скорость работы и т.п.).

1. Тестирование пользовательского интерфейса (GUI) – тестирование, выполняемое путем взаимодействия с системой через графический интерфейс пользователя.

- навигация;
- цвета, графика, оформление;
- содержание выводимой информации;
- поведение курсора и горячие клавиши;
- отображение различного количества данных (нет данных, минимальное и максимальное количество);
- изменение размеров окна или разрешения экрана.

2. Тестирование удобства использования (Usability Testing) – тестирование с целью определения степени понятности, легкости в изучении и использовании, привлекательности программного продукта для пользователя при условии использования в заданных условиях эксплуатации.

- визуальное оформление;
- навигация;
- логичность.

3. Тестирование доступности (Accessibility testing) – тестирование, определяет степень легкости, с которой пользователи с ограниченными способностями могут использовать систему или ее компоненты.

4. Тестирование интернационализации – тестирование способности продукта работать в локализованных средах (способность изменять элементы интерфейса в зависимости от длины и направления текста, менять сортировки/форматы под различные локали и т.д.).

Интернационализация – это процесс, упрощающий дальнейшую адаптацию продукта к языковым и культурным особенностям региона, отличного от того, в котором разрабатывался продукт. Это адаптация продукта для потенциального использования практически в любом месте. Интернационализация производится на начальных этапах разработки, в то время как локализация – для каждого целевого языка.

5. Тестирование локализации (Localization testing) – тестирование, проводимое с целью проверить качество перевода продукта с одного языка на другой.

6. Тестирование производительности или нагрузочное тестирование – процесс тестирования с целью определения производительности программного продукта.

Виды тестов:

- нагрузочное тестирование (Performance and Load testing) – вид тестирования производительности, проводимый с целью оценки поведения компонента или системы при возрастающей нагрузке, например количестве параллельных пользователей и/или операций, а также определения какую нагрузку может выдержать компонент или система;

- объемное тестирование (Volume testing) – позволяет получить оценку производительности при увеличении объемов данных в базе данных приложения;

- тестирование стабильности и надежности (Stability / Reliability testing) – позволяет проверять работоспособность приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки.

- стрессовое тестирование (Stress testing) – вид тестирования производительности, оценивающий систему или компонент на граничных значениях рабочих нагрузок или за их пределами, или же в состоянии ограниченных ресурсов, таких как память или доступ к серверу.

7. Тестирование требований (Requirements testing) – проверка требований на соответствие основным характеристикам качества.

8. Тестирование прототипа (Prototype testing) – метод выявления структурных, логических ошибок и ошибок проектирования на ранней стадии развития продукта до начала фактической разработки.

9. Тестирование установки (Installability testing) и лицензирования – процесс тестирования устанавливаемости программного продукта.

Виды тестов:

- формальный тест программы установки приложения (проверка пользовательского интерфейса, навигации, удобства пользования, соответствия общепринятым стандартам оформления);

- функциональный тест программы установки;

- тестирование механизма лицензирования и функций защиты от пиратства;

- проверка стабильности приложения после установки.

10. Тестирование на отказ и восстановление (Failover and Recovery Testing) – тестирование при помощи эмуляции отказов системы или реально вызываемых отказов в управляемом окружении.

Тестирование программного продукта включает следующие этапы:

1. Изучение и анализ предмета тестирования.

2. Планирование тестирования.

3. Исполнение тестирования.

Изучение и анализ предмета тестирования начинается еще до утверждения спецификации и продолжается на стадии разработки (кодирования) программного обеспечения. Конечной целью этапа изучение и анализ предмета тестирования является получение ответов на два вопроса:

- какие функциональности предстоит протестировать;

- как эти функциональности работают.

Планирование тестирования происходит на стадии разработки (кодирования) программного обеспечения. На стадии планирования тестирования перед тестировщиком стоит задача по-

иска компромисса между объемом тестирования, который возможен в теории, и объемом тестирования, который возможен на практике. На данной стадии необходимо ответить на вопрос: как будем тестировать? Результатом планирования тестирования является тестовая документация.

Выполнение тестирования происходит на стадии тестирования и представляет собой практический поиск дефектов с использованием тестовой документации, составленной ранее.

Для всех программных продуктов выполняют следующие типы тестов и их композиции.

Для первого рекомендуется проводить Smoke+AT готовой функциональности:

- поверхностное тестирование (Smoke Test) выполняется для определения пригодности сборки для дальнейшего тестирования;
- полное тестирование системы или ее части как на корректных, так и на некорректных данных/сценариях (Acceptance Test, AT) позволяет обнаружить дефекты и внести запись о них в багтрекинг-систему.

Для последующих – композиции тестов могут быть следующими:

- Если не была добавлена новая функциональность, то DV + MAT. Выполняется проверка исправления дефектов программистом (Defect Validation, DV), а также проверка работоспособности остальной функциональности после исправления дефектов на позитивных сценариях (Minimal Acceptance Test, MAT).

– Если была добавлена новая функциональность, то Smoke + DV + NFT + Regression Test. В частности, выполняется поверхностное тестирование (Smoke Test), проверка исправления дефектов программистом (Defect Validation, DV), тестирование новых функциональностей (New Feature Testing, NFT), проверка старых функциональностей, т.е. регрессионное тестирование (Regression Test).

- Если была добавлена новая функциональность, то возможен также вариант DV + NFT + Regression test, т.е. без выполнения Smoke Test.



В зависимости от типа и специфики приложения (web, desktop, mobile) выполняют специализированные тесты (например, кроссбраузерное или кроссплатформенное тестирование, тестирование локализации и интернационализации и др.).

### 12.3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Данное практическое занятие предполагает выполнение следующих этапов:

- изучить методические указания;
- ответить на контрольные вопросы.

### 12.4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое тестирование?
2. Какие существуют типы тестов по покрытию?
3. Какие существуют тестовые активности?
4. Какие существуют типы тестов знанию кода?
5. Какие существуют типы тестов по степени автоматизации?
6. Какие существуют типы тестов по изолированности компонентов?
7. Какие существуют типы тестов по подготовленности?
8. Типы тестов по месту и времени проведения.
9. Типы тестов по объекту тестирования?
10. Какие существуют типы функциональных тестов?
11. Типы нефункциональных тестов.
12. Какие этапы составляют процесс тестирования?
13. Что происходит на этапе изучения и анализа предмета тестирования?
14. Что происходит на этапе планирования тестирования?
15. Что происходит на этапе исполнения тестирования?
16. Какие типы тестов выполняют для первой поставки программного продукта?
17. Какие типы тестов выполняют для последующих поставок программного продукта?

## 13 САМОСТОЯТЕЛЬНАЯ РАБОТА. ОСНОВНЫЕ ЭТАПЫ И МЕТОДОЛОГИИ В ПРОЕКТИРОВАНИИ И ВНЕДРЕНИИ ИНФОРМАЦИОННЫХ СИСТЕМ

### 13.1 ЦЕЛЬ РАБОТЫ

Цель работы – изучить методологию и стандарты по внедрения информационных систем.

### 13.2 ОСНОВНЫЕ РАЗДЕЛЫ ИЗУЧАЕМОГО МАТЕРИАЛА

Внедрение информационных систем в жизнь компании сопровождается значительными инвестициями и должно быть основано на серьезной и всесторонней оценке ее уникальных особенностей и требований. Чтобы выбрать информационную систему, наиболее полно отвечающую потребностям вашей компании, необходимо выполнить следующие работы:

- провести документирование и экспресс-анализ основных бизнес-процессов. На основании этого анализа сформулировать требования к информационной системе;
- провести документирование и анализ функций и технических параметров информационных систем и решений, уже работающих в компании, анализ возможностей их объединения в единую корпоративную систему;
- провести аргументированный выбор программных решений и их компонентов, а также принять решение о том, какие из процессов управления предприятием и в каком объеме будут поддерживаться системой автоматизации.

Для успешного внедрения информационной системы специалистам компании также необходимо выполнить полный спектр проектных работ, включающий:

- разработку концепции внедрения;
- внедрение ERP системы, документирование всех настроек, создание системы поддержки и адекватное обучение конечных пользователей;
- дальнейшее непрерывное совершенствование процессов компании.

В качестве начального этапа должно быть проведено документирование существующих бизнес-процессов. Далее можно использовать один из нескольких подходов:

- провести серьезную реорганизацию процессов управления до внедрения информационной системы;
- провести небольшую подстройку процессов под внедряемую информационную систему, внедрить ее, а затем заниматься оптимизацией процессов и соответствующей информационной системы;
- ничего не делать с процессами, а сразу начинать внедрение информационных систем.

Каждый из подходов имеет свои плюсы и минусы, и для выработки той или иной конкретной стратегии необходимо изучить специфику предприятия.

Методология – это учение об организации деятельности.

Информационная система (ИС) – вся инфраструктура предприятия, задействованная в процессе управления всеми информационно-документальными потоками.

Инициация – стадия управления проектом, результатом которой является санкционирование начала работы проекта.

Современные сети разрабатываются на основе стандартов, что позволяет обеспечить, во-первых, их высокую эффективность и, во-вторых, возможность их взаимодействия между собой. Все стандарты на информационные системы (как и на любые системы вообще) можно разбить на следующие два основных класса:

- Функциональные стандарты, определяющие порядок функционирования системы в интересах достижения цели, поставленной перед нею ее создателями.
- Стандарты жизненного цикла, определяющие то, как создается, развертывается, применяется и ликвидируется система.

Модели, определяемые стандартами этих двух классов, конечно же взаимосвязаны, однако решают совершенно разные задачи и характеризуются принципиально различными подходами к их построению.

Жизненный цикл информационной системы охватывает все стадии и этапы ее создания, сопровождения и развития:

- предпроектный анализ (включая формирование функциональной и информационной моделей объекта, для которого предназначена информационная система);
- проектирование системы (включая разработку технического задания, эскизного и технического проектов);
- разработку системы (в том числе программирование и тестирование прикладных программ на основании проектных спецификаций подсистем, выделенных на стадии проектирования);
- интеграцию и сборку системы, проведение ее испытаний;
- эксплуатацию системы и ее сопровождение;
- развитие системы.

Продолжительность жизненного цикла современных информационных систем составляет около 10 лет, что значительно превышает сроки морального и физического старения технических и системных программных средств, используемых при построении системы. Поэтому в течение жизненного цикла системы проводится модернизация ее технико-программной базы. При этом прикладное программное обеспечение системы должно быть сохранено и перенесено на обновляемые аппаратно-программные платформы. Эти проблемы привели к тому, что подавляющее большинство проектов информационных систем внедряется с нарушениями качества, сроков или сметы.

Из мировой практики известно, что затраты на сопровождение прикладного программного обеспечения информационных систем составляют не менее 70% его совокупной стоимости на протяжении жизненного цикла. Поэтому крайне важно еще на проектной стадии предусмотреть необходимые методы и средства сопровождения прикладного программного обеспечения, включая методы конфигурационного управления.

В России создание и испытания автоматизированных систем, к которым относятся и информационные системы, регламентированы рядом ГОСТов, прежде всего серии 34. Однако отдельные положения этих ГОСТов уже устарели, а ряд этапов жизненного цикла информационных систем предоставлены недостаточно полно. Поэтому более целесообразно рассматривать в качестве определяющего документа международный стандарт

ISO/IEC 12207. Данный стандарт определяет структуру жизненного цикла, содержащую процессы, которые должны быть выполнены во время создания программного обеспечения информационной системы.

Эти процессы подразделяются на три группы: основные (приобретение, поставка, разработка, эксплуатация и сопровождение), вспомогательные (документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, оценка, аудит и решение проблем) и организационные (управление проектами, создание инфраструктуры проекта, определение, оценка и улучшение самого жизненного цикла, обучение).

Стандарт ISO/IEC 12207 не предлагает конкретной модели жизненного цикла и методов разработки, его рекомендации являются общими для любых моделей жизненного цикла. Под моделью обычно понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла. Из существующих в настоящее время моделей наиболее распространены две: каскадная и спиральная. Они принципиально различаются самим подходом к информационной системе и ее программному обеспечению. Суть различий в том, что в каскадной модели информационная система является однородной и ее программное обеспечение определяется как единое (с ней) целое. Данный подход характерен для более ранних информационных систем (каскадный метод применяется с 1970 года), а также для систем, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования. При выполнении этих условий каскадный метод позволяет достичь хороших результатов.

Суть каскадного метода заключается в разбиении всей разработки на этапы, причем переход от предыдущего этапа к последующему осуществляется только после полного завершения работ предыдущего этапа. Соответственно на каждом этапе формируется законченный набор проектной документации, достаточной для того, чтобы разработка могла быть продолжена другой группой разработчиков. Другим положительным моментом каскадной модели является возможность планирования сроков завершения работ и затрат на их выполнение. Однако у каскадной модели есть один существенный недостаток – очень сложно

уложить реальный процесс создания программного обеспечения в такую жесткую схему и поэтому постоянно возникает необходимость возврата к предыдущим этапам с целью уточнения и пересмотра ранее принятых решений. Результатом такого конфликта стало появление модели с промежуточным контролем, которую представляют или как самостоятельную модель, или как вариант каскадной модели. Эта модель характеризуется межэтапными корректировками, удлиняющими период разработки изделия, но повышающими надежность.

Однако и каскадная модель, и модель с промежуточным контролем обладают серьезным недостатком – запаздыванием с получением результатов. Данное обстоятельство объясняется тем, что согласование результатов возможно только после завершения каждого этапа работ. На время же проведения каждого этапа требования жестко задаются в виде технического задания. Так что существует опасность, что из-за неточного изложения требований или их изменения за длительное время создания программного обеспечения конечный продукт окажется невостребованным. Для преодоления этого недостатка и была создана спиральная модель, ориентированная на активную работу с пользователями и представляющая разрабатываемую информационную систему как постоянно корректируемую во время разработки. В спиральной модели основной упор делается на этапы анализа и проектирования, на которых реализуемость технических решений проверяется путем создания прототипов. Спиральная модель позволяет начинать работу над следующим этапом, не дожидаясь завершения предыдущего. Спиральная модель имеет целью, как можно раньше ознакомить пользователей с работоспособным продуктом, корректируя при необходимости требования к разрабатываемому продукту и каждый «виток» спирали означает создание фрагмента или версии. Основная проблема спирального цикла – определение момента перехода на следующий этап, и возможным ее решением является принудительное ограничение по времени для каждого из этапов жизненного цикла. Наиболее полно достоинства такой модели проявляются при обслуживании программных средств.

Сравнивая эти модели, можно сказать, что каскадная модель более универсальна, т. е. она применима к производству разных

изделий, будь то отбойный молоток или графический редактор. Для разных изделий просто будут изменяться количество и название этапов модели. Спиральная же модель более ориентирована именно на информационные системы, особенно на программные продукты, поэтому при разработке информационных систем и их программного обеспечения она предпочтительнее каскадной.

Следующим шагом в вопросе поддержания жизненного цикла информационной системы, как, впрочем, и любого другого изделия, является его автоматизация. Однако автоматизация различных процессов, связанных с разработкой, производством и эксплуатацией как изделий промышленности, так и информационных систем наиболее эффективна в том случае, когда она охватывает все этапы жизненного цикла изделия. При этом необходимо преодоление следующих проблем: наличие множества различных систем, ориентированных на решение конкретных задач, относящихся к разным этапам жизненного цикла, приводит к трудностям обмена данными между смежными системами; участие в поддержке жизненного цикла изделия нескольких предприятий требует эффективного обмена информацией об изделии между партнерами; сложность изделия, наличие множества его модификаций, заимствование, стандартизация, унификация, требуют поддержки многоуровневых многовариантных сборочных моделей. Эти проблемы могут быть преодолены путем реализации концепции CALS.

Аббревиатура CALS расшифровывается как Continuous Acquisition and Life cycle Support – непрерывная информационная поддержка жизненного цикла продукта. Встречается также другой перевод, менее схожий с исходным названием, но более близкий по смыслу: обеспечение неразрывной связи между производством и прочими этапами жизненного цикла изделия. Данная технология, разработанная в 80-х годах в Министерстве обороны США, распространилась по всему миру и охватила практически все сферы мировой экономики. Она предназначена для повышения эффективности и качества бизнес-процессов, выполняемых на протяжении всего жизненного цикла продукта, за счет применения безбумажных технологий. Началом создания систе-

мы CALS-технологий явилась разработка системы стандартов описания процессов на всех этапах жизненного цикла продукции.

В международных стандартах серии ISO 9004 (управление качеством продукции) введено понятие «жизненный цикл изделия». Данное понятие включает в себя следующие этапы жизненного цикла изделия: маркетинг, поиск и изучение рынка; проектирование и/или разработка технических требований к создаваемой продукции; материально-техническое снабжение; подготовка и разработка технологических процессов; производство; контроль, проведение испытаний и обследований; упаковка и хранение; реализация и/или распределение продукции; монтаж, эксплуатация; техническая помощь в обслуживании; утилизация после завершения использования продукции.

Для развития методологии CALS в США были созданы Управляющая промышленная группа по вопросам CALS (ISG) и ее исполнительный консультативный комитет. В настоящий момент в мире действует более 25 национальных организаций (комитетов или советов по развитию CALS), в том числе в США, Японии, Канаде, Великобритании, Германии, Швеции, Норвегии, Австралии и других странах, а также в НАТО.

Основные усилия этих и подобных организаций были направлены на создание разного уровня нормативной документации. За последние несколько лет разработаны следующие документы: ISO 10303 (Industrial automation systems and integration – Product data representation and exchange), ISO 13584 (Part Library), Def Stan 00-60 (Integrated Logistic Support), MIL-STD-2549 (Configuration Management. Data Interface), MIL-HDBK-61 (Configuration Management. Guidance), AECMA Specification 2000M (International Specification for Materiel Management Integrated Data Processing for Military Equipment), AECMA Specification 1000D (International Specification for Technical Data Publications, Utilising a Common Source Data Base) и т. д.

Стандарты, разработанные ISO для CALS-технологий, можно разбить на три группы: представление информации о продукте, представление текстовой и графической информации и общего назначения.



К первой группе относятся: ISO/IEC 10303 Standard for the Exchange of Product Model Data (STEP) и ISO 13584 Industrial Automation – Parts Library.

Во вторую группу входят: ISO 8879 Information Processing – Text and Office System – Standard Generalized Markup Language (SGML); ISO/IEC 10179 Document Style Semantics and Specification Language (DSSSL); ISO/IEC IS 10744 Information Technology – Hypermedia/Time Based Document Structuring Language (HyTime); ISO/IEC 8632 Information Processing Systems – Computer Graphics – Metafile; ISO/IEC 10918 Coding of Digital Continuous Tone Still Picture Images (JPEG); ISO 11172 MPEG2 Motion Picture Experts Group (MPEG); Coding of Motion Pictures and associated Audio for Digital Storage Media и ISO/IECS 13522 Information Technology – Coding of Multimedia and Hypermedia Information (MHEG).

Третья группа: ISO 11179 Information Technology – Basic Data Element Attributes; ISO 3166 Information Processing – Country Name Representations; ISO 31 Information Processing Representation of Quantities and Units; ISO 4217 Information Processing – Currencies and Funds; ISO 639 Information Processing Coded Representation of Names of Languages и ISO 8601 Information Processing – Date/Time Representations.

Кроме международных стандартов, разработанных ISO, стандарты CALS широко представлены стандартами с индексами MIL и FIPS, которые лишний раз подчеркивают приоритетность разработки технологии CALS Соединенными Штатами и их военным ведомством изначально (самая многочисленная группа стандартов CALS имеет индекс MIL – стандартный индекс для документов, разработанных в МО США). Аббревиатура FIPS означает федеральный стандарт обработки информации (Federal Information Processing Standard).

Стандарты CALS военного ведомства США, имеющие индекс MIL, также можно разбить на три группы: общих принципов электронного обмена и управления данными; представления текстовой и графической информации; электронных технических руководств.

Стандарты FIPS не так многочисленны, как ISO и MIL, и делятся всего на две группы: описания процессов и безопасности информации.

Госстандарт РФ готовит набор ГОСТов, отражающих, в частности, требования CALS-ориентированных стандартов серии ISO 10303 (Системы автоматизации производства и их интеграция; представление данных об изделии и обмен этими данными). Однако внедрение CALS-подхода в России имеет специфические сложности: часто для использования CALS-решений требуется предварительное проведение реинжиниринга бизнес-процессов; невысок уровень компьютеризации предприятий; отсутствует нормативная база; не хватает специалистов; нет рынка CALS-продуктов и услуг; нет денег на внедрение CALS технологий.

Понятно, что первоочередной задачей для развития CALS-технологий в России является создание соответствующей нормативной базы. Поэтому Госстандартом России и Минэкономки России было принято решение о совместном финансировании разработки ряда первоочередных стандартов, которые открывают путь к внедрению CALS-технологий в отечественной промышленности. В настоящее время уже утверждены первые стандарты в области CALS. Создан и уже действует Технический комитет № 431 при Госстандарте России, организованный на базе научно-исследовательского центра CALS, основная задача которого – разработка стандартов в области CALS.

К настоящему времени приняты следующие стандарты серии «Системы автоматизации производства и их интеграция»:

ГОСТ Р ИСО 10303-1-99. Методы описания. Общий обзор и основополагающие принципы;

ГОСТ Р ИСО 10303 – 21-99. Представление и обмен данными об изделии. Методы реализации. Текстовый обменный файл;

ГОСТ Р ИСО 10303 – 41-99. Представление и обмен данными об изделии. Интегрированные родовые ресурсы. Принципы описания продукта.

### 13.3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что необходимо для успешного внедрения информационной системы?

2. Подходы к документированию.
3. Понятие методологии.
4. Понятие информационной системы.
5. Понятие инициации.
6. Этапы жизненного цикла информационной системы.
7. Понятие CALS-технологий
8. Стандарты, разработанные ISO для CALS-технологий.

## 14 САМОСТОЯТЕЛЬНАЯ РАБОТА. ОРГАНИЗАЦИЯ И ДОКУМЕНТАЦИЯ ПРОЦЕССА ВНЕДРЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

### 14.1 ЦЕЛЬ РАБОТЫ

Цель работы – изучить организацию и документацию процесса внедрения информационных систем.

### 14.2 ОСНОВНЫЕ РАЗДЕЛЫ ИЗУЧАЕМОГО МАТЕРИАЛА

Обзор проектных документов при внедрении ERP-систем  
Уровни и этапы имплементации

Имплементация – реализация международных обязательств на внутригосударственном уровне путём включения международно-правовых норм в национальную правовую систему (законы и подзаконные нормативные акты).

Имплементация корпоративных информационных систем ведется на основе различных подходов, методологий и способов. К уровням имплементации, группирующим набор проектных задач, относят:

1) Основные, обеспечивающие создания содержания проекта:

- процесс;
- данные;
- приложения;
- техника;

2) поддерживающие, обеспечивающие доставку содержания:

- проект;
- изменения.

Типовые этапы внедрения задаются следующими фазами:

- анализа, обеспечивающая сбор требований;
- проектирования, создающая концепцию будущей системы;
- реализации, включающая настройку, разработку и функциональный тест;

- тестирования, содержащая интеграционное и приемочное испытание системы;
- перехода, включая активности обучения;
- продуктивной поддержки.

#### Список проектных документов

Таблица 14.1 содержит список проектных документов, выделенных для этапа и уровня внедрения, что обеспечивает наглядность и объем выполняемых работ.

Таблица 14.1 - Список проектных документов в разрезе фаз внедрения ERP-систем

№	Номер этапа	Название этапа	Уровень	Документ	Вышестоящий документ
1	1	Подготовка к проекту	Проект	Устав проекта	План управления проектом
2	1	Подготовка к проекту	Проект	План-график проекта	План управления проектом
3	1	Подготовка к проекту	Проект	Ресурсный план	План управления проектом
4	1	Подготовка к проекту	Проект	План затрат и бюджет проекта	План управления проектом
5	1	Подготовка к проекту	Проект	RACI-матрица документов и работ	План управления проектом
6	1	Подготовка к проекту	Проект	Список проектной команды и стейкхолдеров	План управления проектом
7	1	Подготовка к	Проект	План ком-	План управ-

№	Номер этапа	Название этапа	Уровень	Документ	Вышестоящий документ
		проекту		муникаций	ления проектом
8	1	Подготовка к проекту	Проект	Реестр рисков	План управления проектом

Продолжение таблицы 14.1

№	Номер этапа	Название этапа	Уровень	Документ	Вышестоящий документ
9	1	Подготовка к проекту	Проект	Список задач и проблем	План управления проектом
10	1	Подготовка к проекту	Проект	План по управлению качеством проекта	План управления проектом
11	1	Подготовка к проекту	Проект	План закупок проекта	План управления проектом
12	1	Подготовка к проекту	Процессы, данные, приложения, техника	Стратегия управления содержанием проекта	Стратегии доставки содержания и изменений
13	1	Подготовка к проекту	Процессы, данные, приложения, техника	Стратегия анализа	Стратегии доставки содержания и изменений
14	1	Подготовка к проекту	Процессы, данные, приложения	Стратегия проектирования	Стратегии доставки содержания и

Продолжение таблицы 14.1

№	Номер этапа	Название эта- па	Уровень	Документ	Вышестоящий документ
			ния		изменений
15	1	Подготовка к проекту	Процессы, данные, приложе- ния	Стратегия ролей и полномочий	Стратегии до- ставки содер- жания и изме- нений
16	1	Подготовка к проекту	Приложе- ния	Стратегия настройки	Стратегии до- ставки содер- жания и изме- нений
17	1	Подготовка к проекту	Данные	Стратегия миграции	Стратегии до- ставки содер- жания и изме- нений
18	1	Подготовка к проекту	Процессы, данные, приложе- ния	Стратегия тестирова- ния	Стратегии до- ставки содер- жания и изме- нений
19	1	Подготовка к проекту	Техника	Стратегия технической подготовки систем	Стратегии до- ставки содер- жания и изме- нений
20	1	Подготовка к проекту	Процессы, данные, приложе- ния, изме- нения	Стратегия обучения	Стратегии до- ставки содер- жания и изме- нений
21	1	Подготовка к проекту	Изменения	Стратегия бизнес пе- рехода	Стратегии до- ставки содер- жания и изме- нений
22	1	Подготовка к	Приложе-	Стратегия	Стратегии до-

Продолжение таблицы 14.1

№	Номер этапа	Название этапа	Уровень	Документ	Вышестоящий документ
		проекту	ния, изменения	внедрения	ставки содержания и изменений
23	1	Подготовка к проекту	Изменения	Стратегия поддержки продуктивного запуска	Стратегии доставки содержания и изменений
24	1	Подготовка к проекту	Изменения	Стратегия изменений	Стратегии доставки содержания и изменений
25	1	Подготовка к проекту	Процессы, данные, приложения, техника	Шаблон плана-трекера (различных активностей)	Шаблоны документов
26	1	Подготовка к проекту	Процессы, данные, приложения, техника	Шаблон матрицы отслеживания требований	Шаблоны документов
27	1	Подготовка к проекту	Процессы, данные, приложения	Шаблон проектного решения	Шаблоны документов
28	1	Подготовка к проекту	Приложения	Шаблон документа настройки	Шаблоны документов
29	1	Подготовка к	Приложе-	Шаблон	Шаблоны до-



Продолжение таблицы 14.1

№	Номер этапа	Название эта- па	Уровень	Документ	Вышестоящий документ
		проекту	ния	специфика- ции на раз- работку	кументов
30	1	Подготовка к проекту	Приложе- ния	Шаблон матрицы ролей и полномочий	Шаблоны до- кументов
31	1	Подготовка к проекту	Данные	Шаблон ре- естра объ- ектов ми- грации	Шаблоны до- кументов
32	1	Подготовка к проекту	Техника	Шаблон плана тех- нического катовера	Шаблоны до- кументов
33	1	Подготовка к проекту	Приложе- ния	Шаблон технической специфика- ции	Шаблоны до- кументов
34	1	Подготовка к проекту	Приложе- ния	Шаблон протокола настроек	Шаблоны до- кументов
35	1	Подготовка к проекту	Изменения	Шаблон сценария интеграци- онного те- стирования	Шаблоны до- кументов
36	1	Подготовка к проекту	Изменения	Шаблон сценария приемочно- го тестиро-	Шаблоны до- кументов

Продолжение таблицы 14.1

№	Номер этапа	Название эта- па	Уровень	Документ	Вышестоящий документ
				вания	
37	1	Подготовка к проекту	Изменения	Шаблон ре- естра де- фектов	Шаблоны до- кументов
38	1	Подготовка к проекту	Изменения	Шаблон обучающего материала	Шаблоны до- кументов
39	1	Подготовка к проекту	Приложе- ния	Шаблон плана Б	Шаблоны до- кументов
40	1	Подготовка к проекту	Изменения	Шаблон плана обес- печения не- прерывно- сти бизнеса	Шаблоны до- кументов
41	1	Подготовка к проекту	Техника, изменения	Шаблон плана отка- та	Шаблоны до- кументов
42	2	Анализ	Проект	План-трекер по проекту в целом	Шаблон пла- на-трекера
43	2	Анализ	Процессы, данные, приложе- ния, тех- ника	Концепция управления содержани- ем проекта	Стратегия управления содержанием проекта
44	2	Анализ	Процессы, данные, приложе- ния, тех- ника	Концепция анализа	Стратегия анализа
45	2	Анализ	Процессы,	План-трекер	Шаблон пла-

Продолжение таблицы 14.1

№	Номер этапа	Название этапа	Уровень	Документ	Вышестоящий документ
			данные, приложения, техника	по сессиям сбора требований	на-трекера
46	2	Анализ	Процессы, данные, приложения, техника	Матрица отслеживания требований	Шаблон матрицы отслеживания требований
47	3	Проектирование	Процессы, данные, приложения	Концепция проектирования	Стратегия проектирования
48	3	Проектирование	Процессы, данные, приложения	Концепция ролей и полномочий	Стратегия ролей и полномочий
49	3	Проектирование	Техника	Концепция технической подготовки систем	Стратегия технической подготовки систем
50	3	Проектирование	Процессы, данные, приложения, техника	План-трекер подготовки проектных документов	Шаблон плана-трекера
51	3	Проектирование	Процессы, данные, приложения, техника	План-трекер тестовых и продуктивного технического катавера	Шаблон плана-трекера

Продолжение таблицы 14.1

№	Номер этапа	Название этапа	Уровень	Документ	Вышестоящий документ
52	3	Проектирование	Процессы, данные, приложения	Проектное решение	Шаблон проектного решения
53	3	Проектирование	Приложения	Документ настройки	Шаблон документа настройки
54	3	Проектирование	Приложения	Спецификация на разработку	Шаблон спецификации на разработку
55	3	Проектирование	Данные	Реестр объектов миграции	Шаблон реестра объектов миграции
56	4	Реализация	Приложения	Концепция настройки и разработки	Стратегия настройки и разработки
57	4	Реализация	Данные	Концепция миграции	Стратегия миграции
58	4	Реализация	Процессы, данные, приложения	Концепция тестирования	Стратегия тестирования
59	4	Реализация	Приложения	План-трекер разработки, настройки и функционального теста	Шаблон плана-трекера
60	4	Реализация	Данные	План-трекер тестовых и продуктив-	Шаблон плана-трекера

Продолжение таблицы 14.1

№	Номер этапа	Название эта- па	Уровень	Документ	Вышестоящий документ
				ной мигра- ции	
61	4	Реализация	Приложе- ния	Техниче- ская специ- фикация	Шаблон тех- нической спе- цификации
62	4	Реализация	Приложе- ния	Протокол настроек	Шаблон про- токола настроек
63	5	Тестирование	Изменения	Сценарий интеграци- онного те- стирования	Шаблон сце- нария инте- грационного тестирования
64	5	Тестирование	Изменения	План-трекер интеграци- онного те- стирования	Шаблон пла- на-трекера
65	5	Тестирование	Изменения	Сценарий приемочно- го тестиро- вания	Шаблон сце- нария прие- мочного те- стирования
66	5	Тестирование	Изменения	План-трекер приемочно- го тестиро- вания	Шаблон пла- на-трекера
67	5	Тестирование	Изменения	Реестр де- фектов	Шаблон ре- естра дефек- тов
68	6	Переход	Процессы, данные, приложе- ния, изме-	Концепция обучения	Стратегия обучения

Продолжение таблицы 14.1

№	Номер этапа	Название эта- па	Уровень	Документ	Вышестоящий документ
			нений		
69	6	Переход	Изменения	Концепция бизнес пе- рехода	Стратегия бизнес пере- хода
70	6	Переход	Приложе- ния, изме- нения	Концепция внедрения	Стратегия внедрения
71	6	Переход	Изменения	Концепция поддержки продуктив- ного запус- ка	Стратегия поддержки продуктивно- го запуска
72	6	Переход	Изменения	Концепция изменений	Стратегия из- менений
73	6	Переход	Изменения	План-трекер обучения конечных пользовате- лей	Шаблон пла- на-трекера
74	6	Переход	Изменения	Обучающий материал	Шаблон обу- чающего ма- териала
75	6	Переход	Изменения	План-трекер бизнес ка- товера	Шаблон пла- на-трекера
76	6	Переход	Приложе- ния	План Б	Шаблон плана Б
77	6	Переход	Изменения	План обес- печения не- прерывно- сти бизнеса	Шаблон плана обеспечения непрерывно- сти бизнеса

Продолжение таблицы 14.1

№	Номер этапа	Название эта- па	Уровень	Документ	Вышестоящий документ
78	6	Переход	Техника, изменения	План отката	Шаблон плана отката

### 14.3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое имплементация?
2. Назовите уровни имплементации.
3. Какие документы относятся к этапу «Подготовка к проекту»?
4. Какие документы относятся к этапу «Анализ»?
5. Перечислите основные уровни проектных документов.

## 15 САМОСТОЯТЕЛЬНАЯ РАБОТА. ИНСТРУМЕНТЫ И ТЕХНОЛОГИИ ВНЕДРЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

### 15.1 ЦЕЛЬ РАБОТЫ

Цель работы – познакомиться с инструментами и технологиями внедрения информационных систем.

### 15.2 ОСНОВНЫЕ РАЗДЕЛЫ ИЗУЧАЕМОГО МАТЕРИАЛА

Сложность современных информационных систем требует применения современных технологий и инструментальных средств, обеспечивающих автоматизацию процессов жизненного цикла информационных систем, называемых CASE-средствами. Использование таких инструментальных средств позволяет существенно сократить длительность и стоимость разработки систем при одновременном повышении качества процесса разработки и, как следствие, качества разработанного продукта.

CASE (Computer Aided Software / System Engineering) – набор инструментов и методов программной инженерии для проектирования программного обеспечения, который помогает обеспечить высокое качество, отсутствие ошибок и простоту в обслуживании программных продуктов.

CASE-технология – совокупность методологий разработки и сопровождения программных средств и сложных систем, поддерживаемая комплексом взаимосвязанных средств автоматизации.

Основные цели использования CASE-технологий при разработке программного обеспечения – отделить анализ и проектирование от программирования и последующих работ процесса разработки, предоставив разработчику соответствующие методологии визуального анализа и проектирования.

CASE-средства – программные средства, поддерживающие процессы создания и (или) сопровождения полного жизненного цикла систем.

CASE-средства предусматривают:

- анализ и формулировку требований;



- проектирование баз данных и приложений;
- генерацию кода;
- тестирование;
- обеспечение качества;
- управление конфигурацией и проектом.

Использование инструментальных средств позволяет не только ускорить работы и повысить качество их выполнения, но и дает инструменты для организации коллективного труда проектной группы.

Фактически CASE-средство – совокупность графически ориентированных инструментальных средств, поддерживающих процессы или отдельные этапы процессов жизненного цикла программного обеспечения.

К CASE-средствам может быть отнесено любое программное средство, обеспечивающее автоматическую помощь при разработке программных систем, их сопровождении или управлении проектом, базирующееся на следующих основополагающих принципах:

- наличие мощных графических средств для описания и документирования системы, обеспечивающих удобный интерфейс с разработчиком и использующих его творческие возможности;
- интеграция отдельных компонентов CASE-средств, обеспечивающая управление процессом разработки программного обеспечения;
- использование специальным образом организованного хранилища проектных метаданных – репозитория.

CASE-средства охватывают обширную область поддержки многочисленных технологий разработки и программных систем – от простых средств анализа и документирования до полномасштабных средств автоматизации.

CASE-средства содержат инструменты различного функционального назначения, поддерживающие различные этапы основных, вспомогательных и организационных процессов жизненного цикла программного обеспечения. В состав CASE-средств входят четыре основных компонента.

1. Средства централизованного хранения всей информации о проекте – репозиторий. Предназначен для хранения информа-

ции о разрабатываемом программном обеспечении или системе в течение всего жизненного цикла разработки.

2. Средства ввода. Служат для ввода данных в репозиторий, организации взаимодействия участников проекта с CASE-средством. Должны поддерживать различные методологии анализа, проектирования, тестирования, контроля. Предназначены для использования в течение жизненного цикла программного обеспечения или системы различными категориями участников проекта (системными аналитиками, проектировщиками, программистами, тестировщиками, менеджерами, специалистами по качеству и т.д.).

3. Средства анализа и разработки. Предназначены для анализа различных видов графических и текстовых описаний и их преобразований в процессе разработки.

4. Средства вывода. Служат для кодогенерации, создания различного вида документов, управления проектом.

Все компоненты CASE-средств в совокупности обладают следующими функциональными возможностями:

- поддержка графических моделей;
- контроль ошибок;
- поддержка репозитория;
- поддержка процессов жизненного цикла информационных систем.

Поддержка графических моделей в CASE-средствах обеспечивает создание и редактирование комплекса взаимосвязанных диаграмм, образующих модели системы. Графические средства моделирования предметной области позволяют в наглядном виде изучать существующую информационную систему, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями. На разных уровнях проектирования могут использоваться различные виды и нотации графического представления программного обеспечения. Разработка диаграмм осуществляется с помощью специальных графических редакторов, основными функциями которых являются создание и редактирование иерархически связанных диаграмм, их объектов и связей между объектами, а также автоматический контроль ошибок.

Контроль ошибок в CASE-средствах предполагает автоматическую верификацию и контроль проекта на полноту и состоя-

тельность уже на ранних этапах разработки, что влияет на успех разработки и внедрения системы в целом.

Поддержка репозитория в CASE-средствах обеспечивает хранение версий проекта, корректную работу проектной группы, контроль метаданных на полноту и непротиворечивость. Репозиторий является базой для стандартизации документации по проекту и контроля проектных спецификаций.

Поддержка процессов жизненного цикла информационных систем включает в себя следующие возможности современных CASE-средств:

- поддержка прототипирования;
- поддержка современных методологий разработки систем;
- автоматическая генерация кода.

Использование CASE-средств позволяет улучшить качество создаваемых проектов, а при создании крупных корпоративных систем является практически неизбежным. Однако использование CASE-средств не освобождает проектировщика от понимания не только общей сущности, но и деталей логического проектирования.

На протяжении осуществления всех работ по внедрению информационной системы все создаваемые модели, планы, рабочие программные компоненты, документация помещаются в репозиторий проекта.

Репозиторий обеспечивает хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость. Содержимое репозитория включает в себя не только информационные объекты различных типов, но и правила использования или обработки этих компонентов. Репозиторий может хранить диаграммы, определения экранов и меню, проекты отчетов, описания данных, исходные коды и т.д. Каждый информационный объект в репозитории описывается перечислением его свойств (идентификатор, синоним, тип, текстовое описание, компоненты, область значений и др.). Кроме того, в нем хранятся правила формирования и редактирования объекта, а также контрольная информация о вре-

мени создания объекта, времени его последнего обновления, номере версии, возможности обновления и т.д.

Репозиторий является базой для стандартизации документации по проекту и контроля проектных спецификаций. Все отчеты строятся автоматически по содержимому репозитория. Документация всегда отражает текущее состояние дел, поскольку любые изменения в проекте автоматически отражаются в репозитории.

Важная часть репозитория внедрения – система документации, формируемая в рамках проекта. Репозиторий выступает базой для автоматической генерации документации по проекту. Основными типами отчетов являются:

- отчеты по содержимому – включают в себя информацию по потокам данных и их компонентов; списки функциональных блоков диаграмм и их входных и выходных потоков; списки всех информационных объектов и их атрибутов; историю изменений объектов; описания модулей и интерфейсов между ними; планы тестирования модулей и т.д.;

- отчеты по перекрестным ссылкам – содержат информацию по связям всех вызывающих и вызываемых модулей; списки объектов репозитория, к которым имеет доступ конкретный исполнитель проекта; информацию по связям между диаграммами и конкретными данными; маршруты движения данных от входа к выходу;

- отчеты по результатам анализа – включают в себя данные по взаимной корректности диаграмм; списки неопределенных информационных объектов; списки неполных диаграмм; данные по результатам анализа структуры проекта и т.д.;

- отчеты по декомпозиции объектов – включают в себя совокупности объектов, входящих в каждый объект, а также объекты, в состав которых входит каждый объект.

Посредством репозитория может осуществляться контроль безопасности (ограничения доступа, привилегии доступа), контроль версий, контроль изменений и др.

Oracle Desinger – продукт компании Oracle, который, возможно, наиболее полно поддерживает все этапы создания информационных систем, но практически поддерживает одну целевую СУБД – Oracle Server.

Oracle Designer обеспечивает графический интерфейс при разработке различных моделей (диаграмм) предметной области. В процессе построения моделей информация о них заносится в репозиторий.

Репозиторий Oracle Designer представляет собой хранилище всех проектных данных и может работать в многопользовательском режиме, обеспечивая параллельное обновление информации несколькими разработчиками.

В процессе проектирования автоматически поддерживаются перекрестные ссылки между объектами словаря и могут генерироваться более 70 стандартных отчетов о моделируемой предметной области. Физическая среда хранения репозитория – база данных Oracle.

Взаимодействие с другими средствами обеспечивается открытым интерфейсом приложений API (Application Programming Interface – интерфейс прикладного программирования). Кроме того, можно использовать средство Oracle CASE Exchange для экспорта/импорта объектов репозитория в целях обмена информацией с другими CASE-средствами.

AllFusion ERwin Data Modeler – CASE-средство, позволяющее создавать, документировать и сопровождать базы данных, хранилища и витрины данных.

Использование модели ERwin одновременно для логического и физического представления данных позволяет по окончании работы получить полностью документированную модель.

Документирование данных является очень важной частью моделирования, так как это дает возможность другим разработчикам или лицам, которые будут сопровождать систему, быстрее начать ориентироваться во внутренней структуре и понимать назначение компонентов.

Для управления групповой разработкой используется средство Model Mart, обеспечивающее многопользовательский доступ к моделям, созданным с помощью ERwin. Модели хранятся на центральном сервере и доступны для всех участников группы проектирования.

Model Mart удовлетворяет требованиям, предъявляемым к средствам управления разработкой крупных информационных систем:

- совместное моделирование – каждый участник проекта имеет инструмент поиска и доступа к интересующей его модели в любое время;
- создание библиотек решений, включающих в себя наиболее удачные фрагменты реализованных проектов, типовые модели, которые объединяются при необходимости в «сборки» больших систем;
- управление доступом – для каждого участника проекта определяются права доступа, в соответствии с которыми они получают возможность работать только с определенными моделями;
- взаимодействие с другими средствами – ERwin поддерживает взаимодействие с Rational Rose.

AllFusion Process Modeler BPwin – CASE-средство, позволяющее проводить моделирование, анализ, описание и последующую оптимизацию бизнес-процессов. С помощью BPwin можно создавать графические модели бизнес-процессов. Графическое изображение схемы выполнения работ, организации документооборота, обмена различными видами информации позволяет визуализировать существующую модель организации бизнеса. Это дает возможность использовать передовые инженерные технологии для решения задач управления организацией.

Для анализа работы организации в комплексе и построения больших моделей предусмотрена детализация. Модели могут быть разбиты на группы.

Следует отметить, что AllFusion Process Modeler имеет широкий набор средств документирования моделей и проектов.

Rational Rose – семейство объектно-ориентированных CASE-средств фирмы Rational Software Corporation, используемое для автоматизации процессов анализа и проектирования программного обеспечения, а также для генерации кодов на различных языках и формирования проектной документации. В основе Rational Rose лежит метод объектно-ориентированного анализа и проектирования, основанный на языке моделирования UML. В составе Rational Rose можно выделить следующие структурные компоненты: репозиторий, графический интерфейс пользователя, средства просмотра проекта (browser), средства контроля проекта, средства сбора статистики, генератор документов, генератор

кодов (индивидуальный для каждого языка) и анализатор для языков программирования, обеспечивающий реверсный инжиниринг.

Репозиторий представляет собой объектно-ориентированную базу данных. Средства просмотра обеспечивают навигацию по проекту. Средства контроля и сбора статистики дают возможность находить и устранять ошибки по мере развития проекта. Генератор отчетов формирует выходные документы на основе содержащейся в репозитории информации.

В результате разработки проекта с помощью CASE-средства Rational Rose формируются следующие документы:

- диаграммы UML, в совокупности представляющие собой модель разрабатываемой программной системы;
- спецификации классов, объектов, атрибутов и операций;
- заготовки текстов программ.

Для поддержки командной работы над проектом на каждой стадии жизненного цикла программного обеспечения имеется интегрированный набор продуктов Rational Suite.

Rational Rose пригодится при решении практически любых задач проектирования информационных систем: от анализа бизнес-процессов до кодогенерации на определенном языке программирования. Такой арсенал позволит не только спроектировать новую систему, но и доработать старую, произведя процесс обратного проектирования.

CALS-технология (Continuous Acquisition and Lifecycle Support – непрерывная информационная поддержка поставок и жизненного цикла изделий) – подход к проектированию и производству продукции, заключающийся в использовании информационных технологий на всех стадиях жизненного цикла изделия. Информационная поддержка реализуется в соответствии с требованиями системы международных стандартов, регламентирующих правила взаимодействия преимущественно посредством электронного обмена данными.

Методы CALS возникли в 1980-х гг. в военном ведомстве США для повышения эффективности управления и планирования в процессе заказа, разработки, организации производства, поставок и эксплуатации военной техники. В России принят аналог

CALS – информационная поддержка жизненного цикла изделий. В последнее время за рубежом наряду с CALS используется также термин PLM (Product Lifecycle Management – управление жизненным циклом продукта).

Главная цель внедрения CALS на предприятии – повышение качества изделия и его конкурентоспособности, минимизация затрат в ходе всего жизненного цикла изделия.

CALS-технологии базируются на наборе интегрированных информационных моделей систем и их производственной и эксплуатационной среды. Благодаря применению компьютерных сетей и стандартных форматов данных CALS-технологии дают возможность обмена информацией во время процессов, выполняемых в ходе жизненного цикла систем.

CALS-технологии расширяют области деятельности предприятий. Ввиду того что CALS-технологии применяют стандартное представление информации на разных стадиях и этапах жизненного цикла, они позволяют кооперироваться разным предприятиям.

В результате применения CALS-технологии, благодаря интегрированности и преемственности информации, повышается эффективность бизнес-процессов на предприятии. Это возможно за счет существенного сокращения сроков освоения производства новых изделий, улучшения качества этих изделий и технической документации, представляемой в электронном виде. Характеристики многих деталей создававшихся ранее изделий, описания систем, процессов, станков и оборудования, задействованных при его изготовлении, хранятся в унифицированном электронном виде и доступны любому пользователю независимо от его местонахождения.

Применение CALS-технологий дает возможность:

- существенно сократить объемы проектных работ;
- снизить затраты на эксплуатацию, обслуживание и ремонт изделий;
- уменьшить трудоемкость процессов технической подготовки и освоения производства новых изделий;
- сократить календарные сроки вывода новых видов продукции на рынок;



– уменьшить долю брака и затрат, связанных с внесением изменений в проект.

Основу современных CALS-технологий составляет разработка открытых распределенных автоматизированных систем для проектирования и управления производством. Необходимым условием их создания является обеспечение единообразного описания и интерпретации данных, независимо от места и времени их получения в общей системе. Структура проектной, технологической и эксплуатационной документации, языки ее представления должны быть стандартизированными. Тогда одна и та же конструкторская документация может быть использована многократно в разных проектах, а одна и та же технологическая документация адаптирована к разным производственным условиям. Это позволяет существенно сократить и удешевить общий цикл проектирования и производства продукта.

Федеральным агентством по техническому регулированию и метрологии (Росстандарт) разработана программа стандартизации в области CALS-технологий.

### 15.3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что относится к CASE-средствам?
2. Перечислите типовые функции инструментария для автоматизации процесса внедрения информационной системы.
3. Каковы основные цели использования CASE-технологий?
4. Для чего используется репозиторий проекта?
5. Какими функциональными возможностями обладают компоненты CASE-средств?
6. Какие отчеты формируют на базе репозитория?
7. Какими особенностями обладает продукт Oracle Desinger?
8. Какие документы можно сформировать в результате разработки проекта с помощью CASE-средства Rational Rose?
9. Какова цель внедрения CALS на предприятии?
10. Что составляет основу современных CALS-технологий?

## ЛИТЕРАТУРА

1. Перлова, О. Н. Соадминистрирование баз данных и серверов : учебник для студентов среднего профессионального образования по специальности 09.02.07 «Информационные системы и программирование» / О. Н. Перлова, О. П. Ляпина ; О. Н. Перлова, О. П. Ляпина. – Москва: Академия, 2020. – 304 с.
2. Федотова, Е. Л. Информационные технологии в профессиональной деятельности: Учебное пособие / Е. Л. Федотова. – Москва : НИЦ ИНФРА-М, 2024. – 367 с.
3. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : Учебное пособие / Г. Н. Федорова. – Москва: НИЦ ИНФРА-М, 2023. – 336 с.
4. Голицына, О. Л. Информационные системы и технологии : Учебное пособие / О. Л. Голицына, Н. В. Попов И. И. Максимов. – Москва: НИЦ ИНФРА-М, 2023. – 400 с.
5. Гвоздева, В. А. Информатика, автоматизированные информационные технологии и системы : Учебник / В. А. Гвоздева. – Москва: НИЦ ИНФРА-М, 2023. – 542 с.
6. Колдаев, В. Д. Архитектура ЭВМ : Учебное пособие / В. Д. Колдаев, С. А. Lupin. – Москва: НИЦ ИНФРА-М, 2023. – 383 с.